Inside Microsoft Networking
all the dirt on the SMB protocol

Andrew Tridgell
E-mail: tridge@samba.org

June 10, 1999

Abstract

The SMB protocol has a long and tortuous history which is unfortunately reflected in the structure of the protocol. In this talk I'll describe the internals of this protocol, which is now used as the central pillar of millions of networks worldwide.

1 Introduction

The SMB protocol is now one of the central pillars of local area networks worldwide. As the default file sharing and printing protocol for all Microsoft OSes it is hard to turn on a sniffer without being hit by barrages of SMB packets. Despite this, many system administrators know very little about how SMB works. This makes their job of fixing problems much harder.

2 The anatomy of SMB

The SMB protocol is rich protocol with many strange extensions and conventions. It is transport independent, with implementations existing for a wide range of common transports including NETBeui, IPX, DECNet and TCP/IP. I'll concentrate on the TCP/IP implementation because that is what I know best and it certainly dominates new installations of SMB.

SMB over TCP/IP works on 3 different ports. Strictly speaking much of the traffic on the first two ports isn't called SMB, it is more properly called NBT (Netbios over TCP/IP) but I hope you'll forgive me if I refer to the lot as SMB occasionally. They are all so intertwined that it really doesn't matter very much.

3 NBT on UDP/137

The first port is UDP/137. You can find a fairly good description of what happens on that port in RFC1001 but basically it is the name resolution port. Computers send name queries on this port and get replies from the owner of the name or from a system setup as a centralized name server. Name "claiming" is also done on this port. That fact that NBT has name claiming at all immediately differentiates it from traditional Unix networking protocols. By default no computer owns any names on the network and a computer, when switched on, must go through the laborious process of staking its claim in the netbios world by sending out name claim messages and seeing if any other computer disputes the name.

The security conscious among you will immediately see all sorts of nasty possibilities with a name resolution system of this sort, but really it isn't too bad. On a LAN it is no worse than the address resolution protocol (ARP) that ethernet LANs use and on a WAN you can block any nasty name resolution datagrams from coming in by blocking UDP/137. There are some consequences to this due to some amazingly silly hacks Microsoft have put into the protocol over the years but on the whole blocking UDP/137 from external networks is a very good idea. Those of you coming to the tutorial might like to ask about some of the silly things that are affected by blocking UDP/137, I don't really have room to describe them here.

Anyway, back to UDP/137. The two big things you need to know about name resolution in NBT is that it is a claim/defend system (and thus has a dynamic naming system) and that it can either operate as a broadcast based system or as a point-to-point "big central server" system. The former should be fairly obvious and is like ARP. The latter is called WINS and is rather similar to DNS except that names are dynamic.

4 WINS

The similarity with DNS isn't just cosmetic. It may surprise you to know that DNS and WINS (and broadcast based NBT for that matter) actually have the same packet format! The idea (as described in RFC1001/1002) was that NBT would eventually be folded in with DNS in servers so that a single daemon could serve both. That didn't happen (and is unlikely to ever happen) but I suppose it was a nice aim. Unfortunately it also led to some rather strange bit twiddling and name encoding in NBT. That is often the price you pay for compatibility.

The knowledge that DNS and NBT share the same packet format probably doesn't help you much really. In so many other ways they are very different. Unicast NBT (known as WINS) does use a central server but doesn't take advantage of the hierarchy system that DNS has. As originally deployed WINS was really used on small WANS with maybe a few hundred to a few thousand computers. These small scales are crucial to understanding a rather nasty design flaw in WINS, it has a flat name space! With so few systems it is easy to choose a different name for each computer on the network so the designers/implementors decided to use the domain field (the bit after the dot in the computers name) to divide administratively separate sets of computers in such a way that two computers with a different domain can't talk to each other at all. They just don't see each other on the network. They also called the field a "scope" instead of a domain to differentiate its meaning a bit. This allowed people to have two computers called FRED on the same network without their names colliding and without requiring users to navigate their way through a tree like structure. The downside is that WINS is totally hopeless for really large WANS (like the Internet). Imagine the fight over who gets the name WWW!

There is one other really big difference between WINS and DNS. WINS (and of course NBT) has the notion of "name types". This is really just the 16th (and last) character in the name but it plays a crucial role in how the whole system hangs together. The name type basically says what purpose the name has. For example type 0x1D is the "master browser" type and is used by a computer that collates names on a LAN so that everyone can quickly work out who is out there at any one time. Type 0x03 is the "user" type which is registered so that logged in users can be contacted by utilities such as WinPopUp. There are lots more but you probably get the idea already.

The fact that name types play such an important role in NBT is what makes it so hard to map DNS onto WINS. Lots of people suggest doing a simple DNS-¿WINS mapping (and Samba can in fact do this) but don't realize that this brings with it the problem of what you do with name types. Often there are two names which differ only by their name type, so how do you work out which is where using DNS?

Ok, so why do people talk so much about WINS? The big thing that WINS gives you that broadcast NBT doesn't have is the ability to operate somewhat reasonably across subnet boundaries. The fact that WINS uses point-to-point (I also call it unicast) communications means that IP routers can handle it and you don't have to bridge your entire network. That is a big win. Is there anything much else to WINS? Not really. In fact the first implementation of WINS I did for Samba was totally accidental! The NBT daemon in Samba (confusingly called nmbd) just didn't bother checking the bits in the packet that say whether it is a unicast or a broadcast. Then when WINS started becoming used on Samba networks people wrote to me to say that it worked great as a WINS server. At the time I hadn't even heard of the name WINS so I was a bit confused but after I saw some sniffs I realized what was happening. nmbd was just sending back reply packets to the originating computer and they were being accepted (perfectly correctly) as though the whole thing was happening with broadcasts.

Before leaving UDP/137 I should mention one final peculiarity. The computer names are "compressed", or at least that is what it is called in RFC1001/1002. The compression system isn't the most efficient that has ever been invented, in fact it doubles the length of all names! Maybe compression refers to the fact that it reduces the alphabet size from a 8 bit range to a 4 bit range. Each character is split into two 4 bit nibbles and is added to 'A' then the whole lot is concatenated. This certainly makes the names look strange in a sniff but it does mean that the "compressed" names are legal DNS names. I've never actually found a case where this has been important so maybe the designers took the notion of DNS compatibility a bit too far.

5 NBT on UDP/138

Enough about port 137, I'll now talk a bit about UDP/138, the browsing port. Port 138 is one of the weirdest beasts in the TCP/IP world. It actually has a mixed byte order! This comes from the fact that the headers are NBT, which uses the traditional network byte order in order to be compatible with DNS, but the body consists of SMB transaction packets which are in Intel byte order reflecting their heritage. Oh, what fun it is dealing with that in a bit of portable source code!

Port 138 is used mostly for browsing. As browsing is a over used word I should explain that in this case I am referring to the thing you do when you click on network neighborhood in a Windows95 box and it brings up a list of workgroups and computers in those workgroups. Or at least you hope it does. This "friendly face" of Windows networking is a source of endless frustration for system administrators fighting for some machine or other to magically appear in that list. I sometimes think there is a market for a "browsing problems 'R us" company or even a whole industry.

The heart of the problem with browsing is its distributed election based nature. It works by having a computer on each network (the local master browser or LMB) whose job it is to collate the list of computers on that network and make that list available to anyone that asks. This sounds simple enough until you realize that exactly which computer performs this task is decided by an election system where the computer with the biggest number wins. That computer might be your marvelous server with great uptimes or it might be Fred's computer with the dodgy ethernet cable, the one that loses half the packets when Fred leans back in his chair.

There are some heuristics that are used to try to make the election somewhat sane, such as preferring computers with later versions of the browsing software, and computers that have been up for longer or that have been marked as "preferred master browsers" but it still is a very hit and miss affair. You even find situations where two computers both think they won the election which leads to consequences not unlike those for the same situation in human politics.

Apart from elections there are other nasties lurking. After winning an election a computer starts listening for "announce" messages from other browse capable systems so it can collate them into a list. If that list gets a bit bigger than a pocket calculator could handle the browse master starts appointing deputies to try and spread the load. These deputies also collate browse lists and one of their addresses is given to any computer that comes along requesting a current list of computers. The idea is that the load will be spread so that the browse master doesn't get overloaded (hardly likely with current CPU/network ratios!) but the effect is that you now have not one point of failure but many. Even worse, the failure is intermittent because you don't know which of the deputies (called a backup master browser) you are going to be referred to. There are some good points to the system, such as smoother transitions when the master dies, but the deputy system certainly does make it a bit harder to debug.

Even if you cope with these problems you will someday be faced with the nastiest of all browsing setups - cross subnet browsing. This is where you have a single logical workgroup (group of computers) that are spread over multiple broadcast networks so they can communicate with unicast packets but not broadcasts. In that case each subnet elects its own master browser which in turn elects deputies and you need some way of getting all these separate lists together so users can "see" computers in other subnets in their browse lists.

The problem is that the browsing protocols were never really designed for this sort of cross-subnet setup. The broadcast election system won't work for multiple subnets, so how do you do it? This is where WINS and a thing called a domain master browser gets in on the act. WINS allows you to have a central name server for netbios names, so what happens is that a specially configured computer (the domain master browser, or DMB) registers the name WORKGROUP#1B with the WINS server and local master browsers contact the WINS server to find out who the DMB is. Then the LMBs can contact the DMB and arrange to exchange browse lists so that (eventually) all subnets know about all other subnets computers.

This sounds OK in theory but can be a real mess in practice. The first stumbling block is placed at your feet courtesy of Microsoft marketing decisions. Microsoft likes selling lots of copies of NT server so they made sure that it is needed. A Win95 box doesn't know how to contact a DMB (even as a client) so if it ends up as the LMB then that subnet is effectively cut off. Oops. It would have been real easy for Microsoft to include DMB client support in Win95 but then of course they would sell fewer copies of NT server. Chalk one up for marketing madness. The result is that you need either a NT server or a Samba server on each subnet to do cross subnet browsing (at least if you do it the way Microsoft intended, Samba has a few extra tricks that allow it to work without a server on each subnet).

Even if you do have a DMB-client capable machine on each subnet and make sure it becomes the LMB you are not out of the woods yet. You have to contend with the problem that the protocol only allows the LMB to contact the DMB for its own workgroup, not any other workgroups in your organizations. So if you have different workgroups for ACCOUNTING and ADMIN then the DMBs need some way to talk to each other or you will never see a workgroup list containing both workgroups. Unfortunately that bit of the protocol (which would have been very easy to do) has been left out completely. Oops again! The only way it can work is if there happens to be a subnet somewhere in your organization where both workgroups coexist. Then the two LMBs will notice each other the two workgroups will magically start to see each other.

For Samba we have added a simple extension to make this a slightly less hit and miss affair but as NT doesn't support our extension you would have to use Samba as the DMB for each workgroup.

By now you should be thoroughly confused, so I'll move onto the main course in the NBT suite.

6 SMB on TCP/139

The real action with the SMB protocol happens on TCP port 139. That is where file and print sharing happens and other large transfers (such as browse list synchronizations) are done.

Unlike that other major file sharing protocol for Unix systems, NFS, SMB on TCP/139 is a complex protocol. Whereas NFS has a dozen or so commands that can be sent over the network SMB has hundreds. In fact there are so many possible SMB commands and sub-commands that no one really knows the full extent of the protocol. This happens because the protocol includes a couple of "extension pack" subprotocols (called transactions) which can contain huge numbers of subcommands, most of which are undocumented and which grow every time you stare at them carefully. Unraveling the most important of these has been one of the more time consuming tasks the Samba Team has had to deal with.

The other notable thing about the file sharing part of SMB is all the PCisms that it has and the way it has been added onto with each new development in the Windows world. The protocol shows very little in the way of foresight, instead having things like 8.3 names mixed up with special exception cases to the semantics for any files ending in .exe or .sym. It is not a pretty protocol.

It does have some good features though. The idea of packet chaining, where you can paste together two requests (such as open and read) into one packet and get the second part to use the result of the first is really very good as it has the potential to save on lots of latency. It is just a pity that common implementations (ie. Windows clients) don't take advantage of this sort of thing much and instead use ten packets where one would suffice.

7 SMB authentication

Before I go off on too much of a SMB bashing expedition I should take a detour and actually tell you something that might come in handy. The thing that comes to mind is the difference between user and share level security in SMB and why neither is what you probably want, despite the fact that they are the only choices available.

When a client connects to a SMB server with the intention of accessing some files or perhaps a printer it first goes through a negotiation stage where the client and server agree on what SMB dialect they will talk and what security model they will use. The security model is the most interesting thing. There are basically two choices, "user level" security and "share level" security. The "modern" choice is user level security.

The choice (which is made by the server, not the client) determines whether the protocol has the notion of a username for authentication and when the client sends the password. If the server chooses share level security then it is telling the client that access to resources is controlled by password only and that the client need not send a username at all if it doesn't want to.

If the server chooses user level security then the server is saying that the client must first login to the server with a valid username/password before the server will allow access to any resources at all. Once the login (called a session setup) is done then the client can assume that it will have access to all resources without needing to send any further authentication information for individual resources.

The problems start when you look at when information is made available to the server with the two security models. With share level security the client never sends a username at all (unless it decides that it wants to) so interfacing with standard username/password based security systems is a bit tricky to say the least. With user level security the login is done *before* the client tells the server what resource (or resources) it wants to access so the server needs to send a yes/no answer without taking info account the desired resource.

So if you want users to authenticate themselves for their home directories but also want some users without server accounts to be able to get to guest resources then you are out of luck. The protocol will fight you all the way until you finally admit defeat and setup two virtual servers, one in share level security for guest access and the other in user level security for home directories.

Which brings us to another bit of ranting and raving. Early on the Samba Team realized that this guest/user type of access was an important thing to have working so we made it easy to setup virtual servers with different security parameters and with multiplexing based on the name of the server. Microsoft didn't have this facility so when they started a revamp of the protocol (which is still underway) they decided to leave out an essential piece of the protocol (the session request where the client sends the server name to the server). Our protests were largely ignored and it looks like this facility will be lost. So much for the CIFS effort to collaboratively develop this protocol.

8 The future

By now you will have probably guessed that I am not actually a great fan of this protocol despite (or because of!) having spent several years working with it as part of the Samba Team. The protocol is still developing, however, and some of the developments aren't too bad. You should really look at the archive of the CIFS list for details but I'll give a rough outline here.

- no-netbios operation. netbios, the bit of the protocol that gives us those silly compressed names and the flat name space, is being scrapped. This is definitely a good thing as long as we don't lose any important facilities (such as virtual servers) in the bargain. It is being replaced with LDAP which by all accounts seems to be a much better solution.
- kerberos authentication. The challenge response authentication in SMB is really not very good. Apart from gaping holes resulting from poor design choices and bad implementations it also is a one-off system that fits in only in the Microsoft world. It needs replacing. The CIFS solution is a kerberos based system and another negotiation section in SMB to handle it. Although this sounds good on the surface there are some downsides. First off, it sounds like there will be little bits of Microsoft proprietary data in the kerberos stream so it might be tricky to be completely compliant. We'll see. The real problem, however, is that it once again relies on a specialist back end security database. I pushed quite a bit for a public key based system which could interface to any existing security backend but without any success. Pity.
- signed and encrypted traffic. It looks like a few years will see SMB being packet signed and encrypted. That could be good if it is done well (that would be a first for a Microsoft designed security system!) but could also make things tricky for non-Microsoft servers.
- new port number. To support the netbios-less operation the protocol will move to a new port number. Microsoft pushed really hard for it to be a port number above 1024 which would have been a security nightmare but luckily sense finally prevailed and a low port number was chosen. Let's hope the new port number also heralds a new cleaner attitude to the development of this protocol.

9 Conclusion

This paper turned into a bit more of a rant and rave than I first intended, but maybe that is a good thing. I hope it gives a few system administrators some insight into how SMB works underneath and where some of the weaknesses are.

If you want more info then go to http://samba.org/ and follow your nose. You should be able to find lots of reading material from there.