1 Learning the Samba

If you are a typical system administrator, then you know what it means to be *swamped* with work. Your daily routine is filled with endless hardware incompatibility issues, system outages, data backup problems, and a steady stream of angry users. So adding another program to the mix of tools that you have to maintain may sound a bit perplexing. However, if you're determined to reduce the complexity of your work environment, as well as the workload of keeping it running smoothly, Samba may be the tool you've been waiting for.

A case in point: one of the authors of this book used to look after 70 Unix developers sharing 5 Unix servers. His neighbor administered 20 Windows 3.1 users and 5 OS/2 and Windows NT servers. To put it mildly, the Windows 3.1 administrator was swamped. When he finally left—and the domain controller melted—Samba was brought to the rescue. Our author quickly replaced the Windows NT and OS/2 servers with Samba running on a Unix server, and eventually bought PCs for most of the company developers. However, he did the latter without hiring a new PC administrator; the administrator now manages one centralized Unix application instead of fifty distributed PCs.

If you know you're facing a problem with your network and you're sure there is a better way, we encourage you to start reading this book. Or, if you've heard about Samba and you want to see what it can do for you, this is also the place to start. We'll get you started on the path to understanding Samba and its potential. Before long, you can provide Unix services to all your Windows machines—all without spending tons of extra time or money. Sound enticing? Great, then let's get started.

What is Samba?

Samba is a suite of Unix applications that speak the SMB (Server Message Block) protocol. Many operating systems, including Windows and OS/2, use SMB to perform client-server networking. By supporting this protocol, Samba allows Unix servers to get in on the action, communicating with the same networking protocol as Microsoft Windows products. Thus, a Samba-enabled Unix machine can masquerade as a server on your Microsoft network and offer the following services:

- Share one or more filesystems
- Share printers installed on both the server and its clients
- Assist clients with Network Neighborhood browsing
- Authenticate clients logging onto a Windows domain
- Provide or assist with WINS name server resolution

Samba is the brainchild of Andrew Tridgell, who currently heads the Samba development team from his home of Canberra, Australia. The project was born in 1991 when Andrew created a fileserver program for his local network that supported an odd DEC protocol from Digital Pathworks. Although he didn't know it at the time, that protocol later turned out to be SMB. A few years later, he expanded upon his custom-made SMB server and began distributing it as a product on the Internet under the name SMB Server. However, Andrew couldn't keep that name—it already belonged to another company's product—so he tried the following Unix renaming approach:

```
grep -i 's.*m.*b' /usr/dict/words
```

And the response was:

```
salmonberry samba sawtimber scramble
```

Thus, the name "Samba" was born.*

Today, the Samba suite revolves around a pair of Unix daemons that provide shared resources—or *shares*—to SMB clients on the network. (Shares are sometimes called s*ervices* as well.) These daemons are:

smbd
: A daemon that allows file and printer sharing on an SMB network and provides authentication and authorization for SMB clients.

nmbd
: A daemon that looks after the Windows Internet Name Service (WINS), and assists with browsing.

* Which is a good thing, because our marketing people highly doubt you would have picked up a book called "Using Salmonberry"!

Samba is currently maintained and extended by a group of volunteers under the active supervision of Andrew Tridgell. Like the Linux operating system, Samba is considered *Open Source software* (OSS) by its authors, and is distributed under the GNU General Public License (GPL). Since its inception, development of Samba has been sponsored in part by the Australian National University, where Andrew Tridgell earned his Ph.D.[*] In addition, some development has been sponsored by independent vendors such as Whistle and SGI. It is a true testament to Samba that both commercial and non-commercial entities are prepared to spend money to support an Open Source effort.

Microsoft has also contributed materially by putting forward its definition of SMB and the Internet-savvy Common Internet File System (CIFS), as a public Request for Comments (RFC), a standards document. The CIFS protocol is Microsoft's renaming of future versions of the SMB protocol that will be used in Windows products—the two terms can be used interchangeably in this book. Hence, you will often see the protocol written as "SMB/CIFS."

What Can Samba Do For Me?

As explained earlier, Samba can help Windows and Unix machines coexist in the same network. However, there are some specific reasons why you might want to set up a Samba server on your network:

- You don't want to pay for—or can't afford—a full-fledged Windows NT server, yet you still need the functionality that one provides.
- You want to provide a common area for data or user directories in order to transition from a Windows server to a Unix one, or vice versa.
- You want to be able to share printers across both Windows and Unix workstations.
- You want to be able to access NT files from a Unix server.

Let's take a quick tour of Samba in action. Assume that we have the following basic network configuration: a Samba-enabled Unix machine, to which we will assign the name `hydra`, and a pair of Windows clients, to which we will assign the names `phoenix` and `chimaera`, all connected via a local area network (LAN). Let's also assume that `hydra` also has a local inkjet printer connected to it, `lp`, and a disk share named `network`—both of which it can offer to the other two machines. A graphic of this network is shown in Figure 1-1.

[*] At the time of this printing, Andrew had completed his Ph.D. work and had joined San Francisco-based LinuxCare.

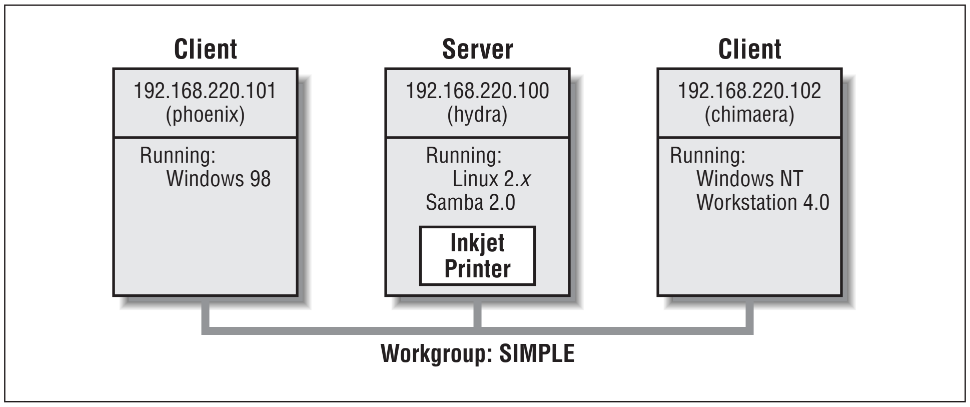
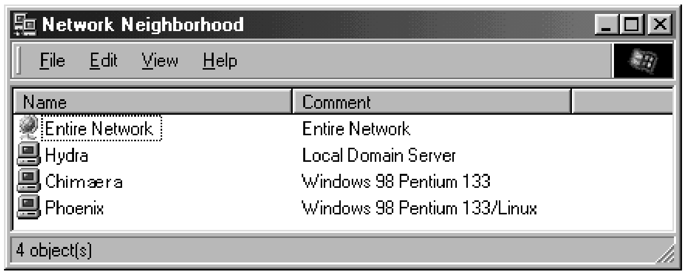


Figure 1-1. A simple network setup with a Samba server

In this network, each of the computers listed share the same *workgroup*. A workgroup is simply a group nametag that identifies an arbitrary collection of computers and their resources on an SMB network. There can be several workgroups on the network at any time, but for our basic network example, we'll have only one: the SIMPLE workgroup.

Sharing a Disk Service

If everything is properly configured, we should be able to see the Samba server, `hydra`, through the Network Neighborhood of the `phoenix` Windows desktop. In fact, Figure 1-2 shows the Network Neighborhood of the `phoenix` computer, including `hydra` and each of the computers that reside in the SIMPLE workgroup. Note the Entire Network icon at the top of the list. As we just mentioned, there can be more than one workgroup on an SMB network at any given time. If a user clicks on the Entire Network icon, he or she will see a list of all the workgroups that currently exist on the network.

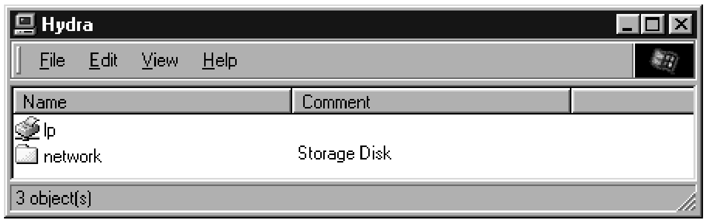


Name	Comment
Entire Network	Entire Network
Hydra	Local Domain Server
Chimaera	Windows 98 Pentium 133
Phoenix	Windows 98 Pentium 133/Linux

4 object(s)

Figure 1-2. The Network Neighborhood directory

We can take a closer look at the `hydra` server by double-clicking on its icon. This contacts `hydra` itself and requests a list of its *shares*—the file and printer resources—that the machine provides. In this case, there is a printer entitled `lp` and a disk share entitled `network` on the server, as shown in Figure 1-3. Note that the Windows display shows hostnames in mixed case (Hydra). Case is irrelevant in hostnames, so you may see hydra, Hydra, and HYDRA in various displays or command output, but they all refer to a single system. Thanks to Samba, Windows 98 sees the Unix server as a valid SMB server, and can access the `network` folder as if it were just another system folder.



Name	Comment
lp	
network	Storage Disk

3 object(s)

Figure 1-3. Shares available on the hydra sever as viewed from phoenix

One popular feature of Windows 95/98/NT is that you can map a letter-drive to a known network directory using the Map Network Drive option in the Windows Explorer.* Once you do so, your applications can access the folder across the network with a standard drive letter. Hence, you can store data on it, install and run programs from it, and even password-protect it against unwanted visitors. See Figure 1-4 for an example of mapping a letter-drive to a network directory.

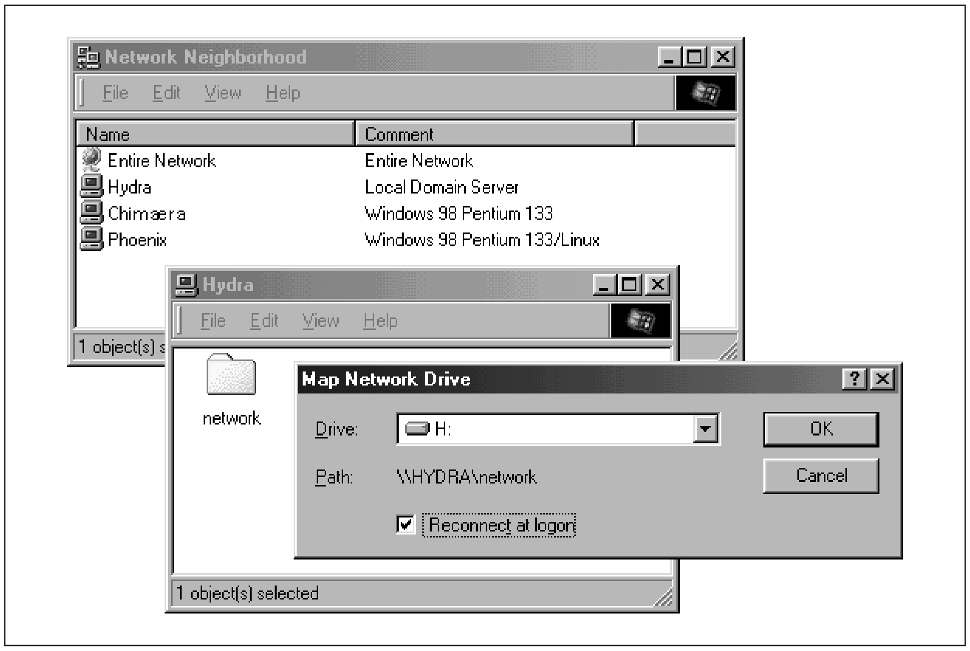
Take a look at the Path: entry in the dialog box of Figure 1-4. An equivalent way to represent a directory on a network machine is by using two backslashes, followed by the name of the networked machine, another backslash, and the networked directory of the machine, as shown below:

```
\\network-machine\directory
```

This is known as the *UNC* (Universal Naming Convention) in the Windows world. For example, the dialog box in Figure 1-4 represents the network directory on the `hydra` server as:

```
\\HYDRA\network
```

* You can also right-click on the shared resource in the Network Neighborhood, and then select the Map Network Drive menu item.



Name	Comment
Entire Network	Entire Network
Hydra	Local Domain Server
Chimaera	Windows 98 Pentium 133
Phoenix	Windows 98 Pentium 133/Linux

Drive: H:

Path: \\HYDRA\network

[✓] Reconnect at logon

Figure 1-4. Mapping a network drive to a Windows letter-drive

If this looks somewhat familiar to you, you're probably thinking of *uniform resource locators* (URLs), which are addresses that web browsers such as Netscape Navigator and Internet Explorer use to resolve machines across the Internet. Be sure not to confuse the two: web browsers typically use forward slashes instead of back slashes, and they precede the initial slashes with the data transfer protocol (i.e., ftp, http) and a colon (:). In reality, URLs and UNCs are two completely separate things.

Once the network drive is set up, Windows and its programs will behave as if the networked directory was a fixed disk. If you have any applications that support multiuser functionality on a network, you can install those programs on the network drive.* Figure 1-5 shows the resulting network drive as it would appear with other storage devices in the Windows 98 client. Note the pipeline attachment in the icon for the G: drive; this indicates that it is a network drive instead of a fixed drive.

From our Windows NT Workstation machine, `chimaera`, Samba looks almost identical to Windows 98. Figure 1-6 shows the same view of the `hydra` server from the Windows NT 4.0 Network Neighborhood. Setting up the network drive

* Be warned that many end-user license agreements forbid installing a program on a network such that multiple clients can access it. Check the legal agreements that accompany the product to be absolutely sure.

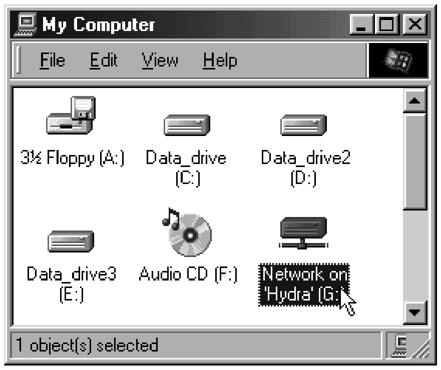


Figure 1-5. The Network directory mapped to the client letter-drive G

using the Map Network Drive option in Windows NT Workstation 4.0 would have identical results as well.

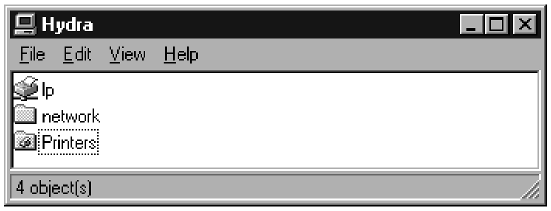


Figure 1-6. Shares available on hydra (viewed from chimaera)

Sharing a Printer

You probably noticed that the printer `lp` appeared under the available shares for `hydra` in Figure 1-3. This indicates that the Unix server has a printer that can be shared by the various SMB clients in the workgroup. Data sent to the printer from any of the clients will be spooled on the Unix server and printed in the order it is received.

Setting up a Samba-enabled printer on the Windows side is even easier than setting up a disk share. By double-clicking on the printer and identifying the manufacturer and model, you can install a driver for this printer on the Windows client. Windows can then properly format any information sent to the network printer and access it as if it were a local printer (we show you how to do this later in the chapter). Figure 1-7 shows the resulting network printer in the Printers window of Windows 98. Again, note the pipeline attachment below the printer, which identifies it as being on a network.

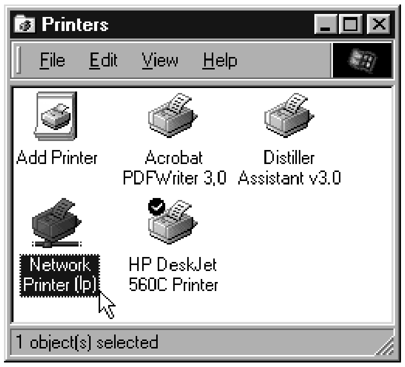


Figure 1-7. A network printer available on hydra (viewed from chimaera)

Seeing things from the Unix side

As mentioned earlier, Samba appears in Unix as a set of daemon programs. You can view them with the Unix `ps` and `netstat` commands, you can read any messages they generate through custom debug files or the Unix `syslog` (depending on how Samba is set up), and you can configure it from a single Samba properties file: *smb.conf*. In addition, if you want to get an idea of what each of the daemons are doing, Samba has a program called *smbstatus* that will lay it all on the line. Here is how it works:

```
# smbstatus
Samba version 2.0.4
Service      uid      gid      pid     machine
----------------------------------------------
network      davecb   davecb   7470   phoenix  (192.168.220.101) Sun May 16
network      davecb   davecb   7589   chimaera (192.168.220.102) Sun May 16

Locked files:
Pid    DenyMode   R/W        Oplock          Name
--------------------------------------------------
7589   DENY_NONE  RDONLY     EXCLUSIVE+BATCH /home/samba/quicken/inet/common/
system/help.bmp   Sun May 16 21:23:40 1999
7470   DENY_WRITE RDONLY     NONE           /home/samba/word/office/findfast.exe
Sun May 16 20:51:08 1999
7589   DENY_WRITE RDONLY     EXCLUSIVE+BATCH /home/samba/quicken/lfbmp70n.dll
Sun May 16 21:23:39 1999
7589   DENY_WRITE RDWR       EXCLUSIVE+BATCH /home/samba/quicken/inet/qdata/
runtime.dat   Sun May 16 21:23:41 1999
7470   DENY_WRITE RDONLY     EXCLUSIVE+BATCH /home/samba/word/office/osa.exe
Sun May 16 20:51:09 1999
7589   DENY_WRITE RDONLY     NONE            /home/samba/quicken/qversion.dll
Sun May 16 21:20:33 1999
```

```
7470   DENY_WRITE RDONLY     NONE                              /home/samba/quicken/
qversion.dll   Sun May 16 20:51:11 1999

Share mode memory usage (bytes):
   1043432(99%) free + 4312(0%) used + 832(0%) overhead = 1048576(100%) total
```

The Samba status from this output provides three sets of data, each divided into separate sections. The first section tells which systems have connected to the Samba server, identifying each client by its machine name (`phoenix` and `chimaera`) and IP address. The second section reports the name and status of the files that are currently in use on a share on the server, including the read/write status and any locks on the files. Finally, Samba reports the amount of memory it has currently allocated to the shares that it administers, including the amount actively used by the shares plus additional overhead. (Note that this is not the same as the total amount of memory that the *smbd* or *nmbd* processes are using.)

Don't worry if you don't understand these statistics; they will become easier to understand as you move through the book.

Getting Familiar with a SMB/CIFS Network

Now that you have had a brief tour of Samba, let's take some time to get familiar with Samba's adopted environment: an SMB/CIFS network. Networking with SMB is significantly different from working with a Unix TCP/IP network, because there are several new concepts to learn and a lot of information to cover. First, we will discuss the basic concepts behind an SMB network, followed by some Microsoft implementations of it, and finally we will show you where a Samba server can and cannot fit into the picture.

Understanding NetBIOS

To begin, let's step back in time. In 1984, IBM authored a simple application programming interface (API) for networking its computers called the *Network Basic Input/Output System* (NetBIOS). The NetBIOS API provided a rudimentary design for an application to connect and share data with other computers.

It's helpful to think of the NetBIOS API as networking extensions to the standard BIOS API calls. With BIOS, each low-level call is confined to the hardware of the local machine and doesn't need any help traveling to its destination. NetBIOS, however, originally had to exchange instructions with computers across IBM PC or Token Ring networks. It therefore required a low-level transport protocol to carry its requests from one computer to the next.

In late 1985, IBM released one such protocol, which it merged with the NetBIOS API to become the *NetBIOS Extended User Interface* (*NetBEUI*). NetBEUI was designed for small local area networks (LANs), and it let each machine claim a name (up to 15 characters) that wasn't already in use on the network. By a "small LAN," we mean fewer than 255 nodes on the network—which was considered a practical restriction in 1985!

The NetBEUI protocol was very popular with networking applications, including those running under Windows for Workgroups. Later, implementations of NetBIOS over Novell's IPX networking protocols also emerged, which competed with NetBEUI. However, the networking protocols of choice for the burgeoning Internet community were TCP/IP and UDP/IP, and implementing the NetBIOS APIs over those protocols soon became a necessity.

Recall that TCP/IP uses numbers to represent computer addresses, such as 192.168.220.100, while NetBIOS uses only names. This was a major issue when trying to mesh the two protocols together. In 1987, the Internet Engineering Task Force (IETF) published a series of standardization documents, titled RFC 1001 and 1002, that outlined how NetBIOS would work over a TCP/UDP network. This set of documents still governs each of the implementations that exist today, including those provided by Microsoft with their Windows operating systems as well as the Samba suite.

Since then, the standard this document governs has become known as *NetBIOS over TCP/IP*, or NBT for short. The NBT standard (RFC 1001/1002) currently outlines a trio of services on a network:

- A name service
- Two communication services:
  - Datagrams
  - Sessions

The name service solves the name-to-address problem mentioned earlier; it allows each computer to declare a specific name on the network that can be translated to a machine-readable IP address, much like today's DNS on the Internet. The datagram and session services are both secondary communication protocols used to transmit data back and forth from NetBIOS machines across the network.

Getting a Name

For a human being, getting a name is easy. However, for a machine on a NetBIOS network, it can be a little more complicated. Let's look at a few of the issues.

In the NetBIOS world, when each machine comes online, it wants to claim a name for itself; this is called *name registration*. However, no two machines in the same workgroup should be able to claim the same name; this would cause endless confusion for any machine that wanted to communicate with either machine. There are two different approaches to ensuring that this doesn't happen:

- Use a *NetBIOS Name Server* (NBNS) to keep track of which hosts have registered a NetBIOS name.
- Allow each machine on the network to defend its name in the event that another machine attempts to use it.

Figure 1-8 illustrates a (failed) name registration, with and without a NetBIOS Name Server.

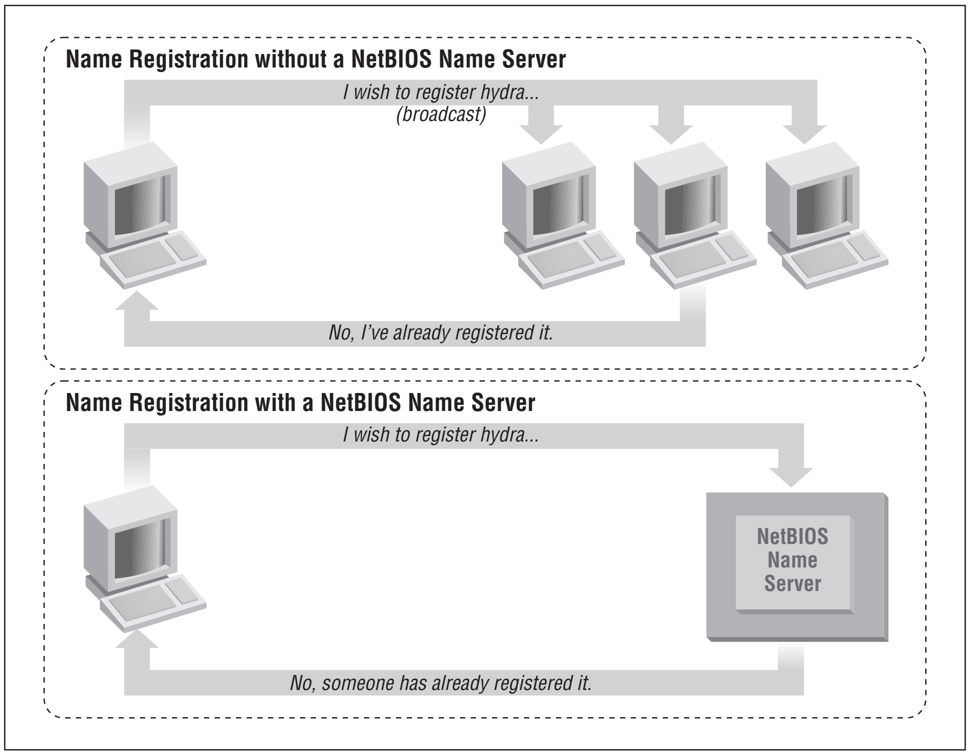

Name Registration without a NetBIOS Name Server
I wish to register hydra... (broadcast)
No, I've already registered it.
Name Registration with a NetBIOS Name Server
I wish to register hydra...
NetBIOS Name Server
No, someone has already registered it.

Figure 1-8. NBNS versus non-NBNS name registration

In addition, there must be a way to resolve a NetBIOS name to a specific IP address as mentioned earlier; this is known as *name resolution*. There are two different approaches with NBT here as well:

- Have each machine report back its IP address when it "hears" a broadcast request for its NetBIOS name.
- Use the NBNS to help resolve NetBIOS names to IP addresses.

Figure 1-9 illustrates the two types of name resolution.

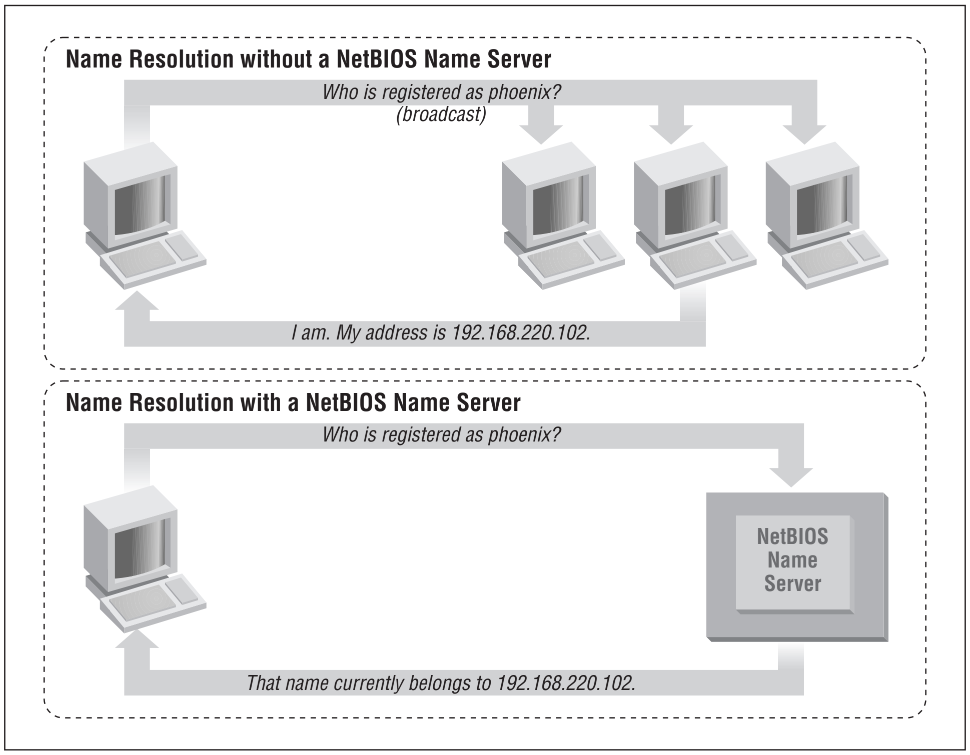

Name Resolution without a NetBIOS Name Server
Who is registered as phoenix? (broadcast)
I am. My address is 192.168.220.102.
Name Resolution with a NetBIOS Name Server
Who is registered as phoenix?
NetBIOS Name Server
That name currently belongs to 192.168.220.102.

Figure 1-9. NBNS versus non-NBNS name resolution

As you might expect, having an NBNS on your network can help out tremendously. To see exactly why, let's look at the non-NBNS method.

Here, when a client machine boots, it will broadcast a message declaring that it wishes to register a specified NetBIOS name as its own. If nobody objects to the use of the name after multiple registration attempts, it keeps the name. On the other hand, if another machine on the local subnet is currently using the requested name, it will send a message back to the requesting client that the name is already taken. This is known as *defending* the hostname. This type of system comes in handy when one client has unexpectedly dropped off the network—another can take its name unchallenged—but it does incur an inordinate amount of traffic on the network for something as simple as name registration.

With an NBNS, the same thing occurs, except that the communication is confined to the requesting machine and the NBNS server. No broadcasting occurs when the machine wishes to register the name; the registration message is simply sent directly from the client to NBNS server and the NBNS server replies whether or not the name is already taken. This is known as *point-to-point communication*, and is often beneficial on networks with more than one subnet. This is because routers are often preconfigured to block incoming packets that are broadcast to all machines in the subnet.

The same principles apply to name resolution. Without an NBNS, NetBIOS name resolution would also be done with a broadcast mechanism. All request packets would be sent to each computer in the network, with the hope that one machine that might be affected will respond directly back to the machine that asked. At this point, it's clear that using an NBNS server and point-to-point communication for this purpose is far less taxing on the network than flooding the network with broadcasts for every name resolution request.

Node Types

How can you tell what strategy each client on your network will use when performing name registration and resolution? Each machine on an NBT network earns one of the following designations, depending on how it handles name registration and resolution: b-node, p-node, m-node, and h-node. The behaviors of each type of node are summarized in Table 1-1.

Table 1-1. NetBIOS Node Types

Role	Value
b-node	Uses broadcast registration and resolution only.
p-node	Uses point-to-point registration and resolution only.
m-node	Uses broadcast for registration. If successful, it notifies the NBNS server of the result. Uses broadcast for resolution; uses NBNS server if broadcast is unsuccessful.
h-node (hybrid)	Uses NBNS server for registration and resolution; uses broadcast if the NBNS server is unresponsive or inoperative.

In the case of Windows clients, you will usually find them listed as *h-nodes* or *hybrid nodes*. Incidentally, h-nodes were invented later by Microsoft, as a more fault-tolerant route, and do not appear in RFC 1001/1002.

You can find out the node type of any Windows machine by typing the command `ipconfig /all` and searching for the line that says `Node Type`.

```
C:\>ipconfig /all
Windows 98 IP Configuration
...
  Node Type . . . . . . . . . . . : Hybrid
...
```

What's in a Name?

The names NetBIOS uses are quite different from the DNS hostnames you might be familiar with. First, NetBIOS names exist in a flat namespace. In other words, there are no qualifiers such as *ora.com* or *samba.org* to section off hostnames; there is only a single unique name to represent each computer. Second, NetBIOS names are allowed to be only 15 characters, may not begin with an asterisk (*), and can consist only of standard alphanumeric characters (a-z, A-Z, 0-9) and the following:

```
! @ # $ % ^ & ( ) - ' { } . ~
```

Although you are allowed to use a period (.) in a NetBIOS name, we recommend against it because those names are not guaranteed to work in future versions of NetBIOS over TCP/IP.

It's not a coincidence that all valid DNS names are also valid NetBIOS names. In fact, the DNS name for a Samba server is often reused as its NetBIOS name. For example, if you had a machine `phoenix.ora.com`, its NetBIOS name would likely be PHOENIX (followed by 8 blanks).

Resource names and types

With NetBIOS, a machine not only advertises its presence, but also tells others what types of services it offers. For example, `phoenix` can indicate that it's not just a workstation, but is also a file server and can receive WinPopup messages. This is done by adding a 16th byte to the end of the machine (resource) name, called the *resource type*, and registering the name more than once. See Figure 1-10.

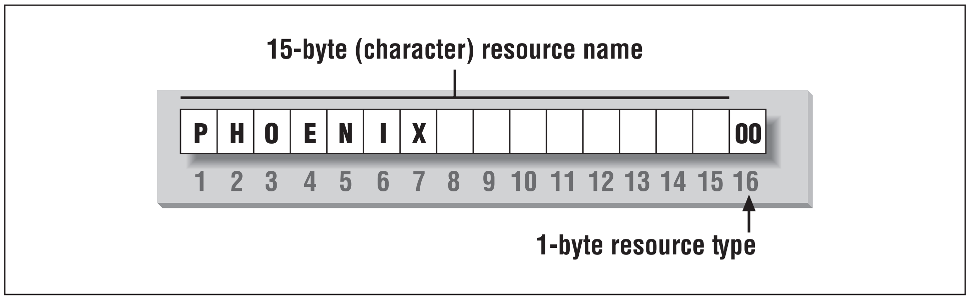


Figure 1-10. The structure of NetBIOS names

The one-byte resource type indicates a unique service the named machine provides. In this book, you will often see the resource type shown in angled brackets (<>) after the NetBIOS name, such as:

```
PHOENIX<00>
```

You can see which names are registered for a particular NBT machine using the Windows command-line NBTSTAT utility. Because these services are unique (i.e., there cannot be more than one registered), you will see them listed as type UNIQUE in the output. For example, the following partial output describes the `hydra` server:

```
D:\>NBTSTAT -a hydra

       NetBIOS Remote Machine Name Table
   Name               Type         Status
---------------------------------------------
HYDRA          <00>  UNIQUE      Registered
HYDRA          <03>  UNIQUE      Registered
HYDRA          <20>  UNIQUE      Registered
...
```

This says the server has registered the NetBIOS name `hydra` as a machine (workstation) name, a recipient of WinPopup messages, and a file server. Some possible attributes a name can have are listed in Table 1-2.

Table 1-2. NetBIOS Unique Resource Types

Named Resource	Hexidecimal Byte Value
Standard Workstation Service	00
Messenger Service (WinPopup)	03
RAS Server Service	06
Domain Master Browser Service (associated with primary domain controller)	1B
Master Browser name	1D
NetDDE Service	1F
Fileserver (including printer server)	20
RAS Client Service	21
Network Monitor Agent	BE
Network Monitor Utility	BF

Note that because DNS names don't have resource types, the designers intentionally made hexidecimal value 20 (an ASCII space) default to the type for a file server.

Group names and types

SMB also uses the concept of groups, with which machines can register themselves. Earlier, we mentioned that the machines in our example belonged to a *workgroup*, which is a partition of machines on the same network. For example, a business might very easily have an ACCOUNTING and a SALES workgroup, each with different servers and printers. In the Windows world, a workgroup and an SMB group are the same thing.

Continuing our NBTSTAT example, the `hydra` Samba server is also a member of the SIMPLE workgroup (the GROUP attribute hex 00), and will stand for election as a browse master (GROUP attribute 1E). Here is the remainder of the NBTSTAT utility output:

```
           NetBIOS Remote Machine Name Table, continued
       Name              Type         Status
    ---------------------------------------------
    SIMPLE          <00>  GROUP       Registered
    SIMPLE          <1E>  GROUP       Registered
    .._ _MSBROWSE_ _.<01>  GROUP       Registered
```

The possible group attributes a machine can have are illustrated in Table 1-3. More information is available in *Windows NT in a Nutshell* by Eric Pearce, also published by O'Reilly.

Table 1-3. NetBIOS Group Resource Types

Named Resource	Hexidecimal Byte Value
Standard Workstation group	00
Logon Server	1C
Master Browser name	1D
Normal Group name (used in browser elections)	1E
Internet Group name (administrative)	20
<01><02>_ _MSBROWSE_ _<02>	01

The final entry, `_ _MSBROWSE_ _`, is used to announce a group to other master browsers. The nonprinting characters in the name show up as dots in a NBTSTAT printout. Don't worry if you don't understand all of the resource or group types. Some of them you will not need with Samba, and others you will pick up as you move through the rest of the chapter. The important thing to remember here is the logistics of the naming mechanism.

Datagrams and Sessions

At this point, let's digress to introduce another responsibility of NBT: to provide connection services between two NetBIOS machines. There are actually two services offered by NetBIOS over TCP/IP: the *session service* and the *datagram service*. Understanding how these two services work is not essential to using Samba, but it does give you an idea of how NBT works and how to troubleshoot Samba when it doesn't work.

The datagram service has no stable connection between one machine and another. Packets of data are simply sent or broadcast from one machine to another, without regard for the order that they arrive at the destination, or even if they arrive at all. The use of datagrams is not as network intensive as sessions, although they can bog down a network if used unwisely (remember broadcast name resolution earlier?) Datagrams, therefore, are used for quickly sending simple blocks of data to one or more machines. The datagram service communicates using the simple primitives shown in Table 1-4.

Table 1-4. Datagram Primitives

Primitive	Description
Send Datagram	Send datagram packet to machine or groups of machines.
Send Broadcast Datagram	Broadcast datagram to any machine waiting with a Receive Broadcast Datagram.
Receive Datagram	Receive a datagram from a machine.
Receive Broadcast Datagram	Wait for a broadcast datagram.

The session service is more complex. Sessions are a communication method that, in theory, offers the ability to detect problematic or inoperable connections between two NetBIOS applications. It helps to think of an NBT session in terms of a telephone call.* A full-duplex connection is opened between a caller machine and a called machine, and it must remain open throughout the duration of their conversation. Each side knows who the caller and the called machine is, and can communicate with the simple primitives shown in Table 1-5.

Table 1-5. Session Primitives

Primitive	Description
Call	Initiate a session with a machine listening under a specified name.
Listen	Wait for a call from a known caller or any caller.
Hang-up	Exit a call.
Send	Send data to the other machine.
Receive	Receive data from the other machine.
Session Status	Get information on requested sessions.

Sessions are the backbone of resource sharing on an NBT network. They are typically used for establishing stable connections from client machines to disk or printer shares on a server. The client "calls" the server and starts trading information such as which files it wishes to open, which data it wishes to exchange, etc. These calls can last a long time—hours, even days—and all of this occurs within the context of a single connection. If there is an error, the session software (TCP) will retransmit until the data is received properly, unlike the "punt-and-pray" approach of the datagram service (UDP).

* As you can see in RFC 1001, the telephone analogy was strongly evident in the creation of the NBT service.

In truth, while sessions are supposed to be able to handle problematic communications, they often don't. As you've probably already discovered when using Windows networks, this is a serious detriment to using NBT sessions. If the connection is interrupted for some reason, session information that is open between the two computers can easily become invalidated. If that happens, the only way to regain the session information is for the same two computers to call each other again and start over.

If you want more information on each of these services, we recommend you look at RFC 1001. However, there are two important things to remember here:

- Sessions always occur between *two* NetBIOS machines—no more and no less. If a session service is interrupted, the client is supposed to store sufficient state information for it to re-establish the connection. However, in practice, this is rarely the case.
- Datagrams can be broadcast to multiple machines, but they are unreliable. In other words, there is no way for the source to know that the datagrams it sent have indeed arrived at their destinations.

Microsoft Implementations

With that amount of background, we can now talk about some of Microsoft's implementations of the preceding concepts in the CIFS/SMB networking world. And, as you might expect, there are some complex extensions to introduce as well.

Windows Domains

Recall that a workgroup is a collection of SMB computers that all reside on a subnet and subscribe to the same SMB group. A *Windows domain* goes a step further. It is a workgroup of SMB machines that has one addition: a server acting as a *domain controller*. You must have a domain controller in order to have a Windows domain.* Otherwise, it is only a workgroup. See Figure 1-11.

There are currently two separate protocols used by a domain controller (logon server): one for communicating with Windows 95/98 machines and one for communicating with Windows NT machines. While Samba currently implements the domain controller protocol for Windows 95/98 (which allows it to act as a domain controller for Windows 9*x* machines), it still does not fully support the protocol for Windows NT computers. However, the Samba team promises that support for the Windows NT domain controller protocol is forthcoming in Samba 2.1.

* Windows domains are called "Windows NT domains" by Microsoft because they assume that Windows NT machines will take the role of the domain controller. However, because Samba can perform this function as well, we'll simply call them "Windows domains" to avoid confusion.

,ch01.26764 Page 19 Friday, November 19, 1999 3:27 PM

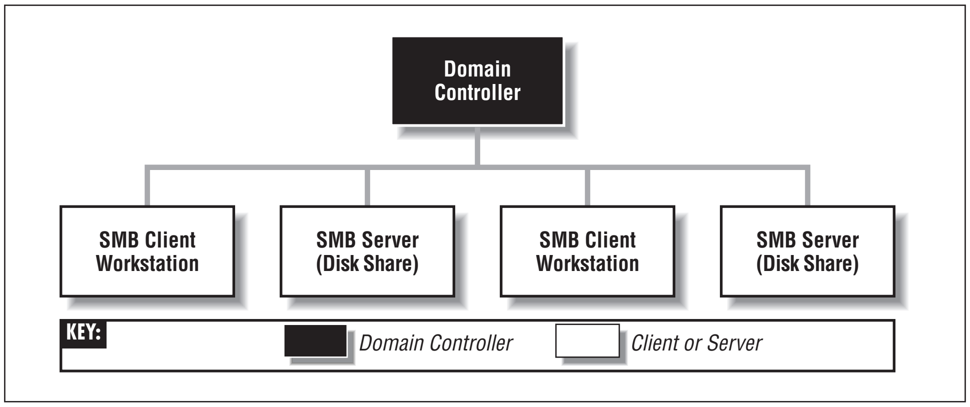


Figure 1-11. A simple Windows domain

Why all the difficulty? The protocol that Windows domain controllers use to communicate with their clients and other domain controllers is proprietary and has not been released by Microsoft. This has forced the Samba development team to reverse-engineer the domain controller protocol to see which codes perform specific tasks.

Domain controllers

The domain controller is the nerve center of a Windows domain, much like an NIS server is the nerve center of the Unix network information service. Domain controllers have a variety of responsibilities. One responsibility that you need to be concerned with is *authentication*. Authentication is the process of granting or denying a user access to a shared resource on another network machine, typically through the use of a password.

Each domain controller uses a *security account manager* (SAM) to maintain a list of username-password combinations. The domain controller then forms a central repository of passwords that are tied to usernames (one password per user), which is more efficient than each client machine maintaining hundreds of passwords for every network resource available.

On a Windows domain, when a non-authenticated client requests access to a server's shares, the server will turn around and ask the domain controller whether that user is authenticated. If it is, the server will establish a session connection with the access rights it has for that service and user. If not, the connection is denied. Once a user is authenticated by the domain controller, a special authenticated token will be returned to the client so that the user will not need to relogin to other resources on that domain. At this point, the user is considered "logged in" to the domain itself. See Figure 1-12.

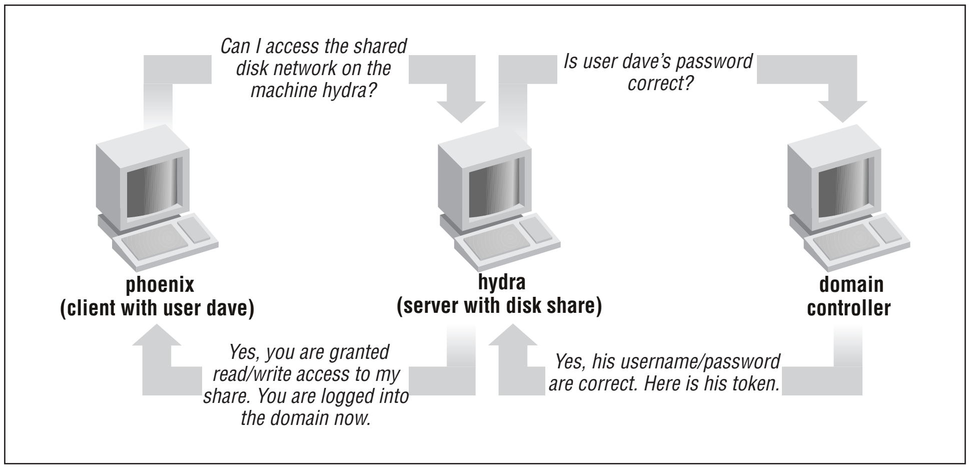

Can I access the shared disk network on the machine hydra?
Is user dave's password correct?
phoenix (client with user dave)
hydra (server with disk share)
domain controller
Yes, you are granted read/write access to my share. You are logged into the domain now.
Yes, his username/password are correct. Here is his token.

Figure 1-12. Using a domain controller for authentication

Primary and backup domain controllers

Redundancy is a key idea behind a Windows domain. The domain controller that is currently active on a domain is called the *primary domain controller* (PDC). There can be one or more *backup domain controllers* (BDCs) in the domain as well, which will take over in the event that the primary domain controller fails or becomes inaccessible. BDCs frequently synchronize their SAM data with the primary domain controller so that, if the need arises, any one of them can perform DC services transparently without impacting its clients. Note that BDCs, however, have only read-only copies of the SAM; they can update their data only by synchronizing with a PDC. A server in a Windows domain can use the SAM of any primary or backup domain controller to authenticate a user who attempts to access its resources and logon to the domain.

Note that in many aspects, the behaviors of a Windows workgroup and a Windows domain overlap. This is not accidental since the concept of Windows domains did not evolve until Windows NT 3.5 was introduced, and Windows domains were forced to remain backwards compatible with the workgroups present in Windows for Workgroups 3.1. The key thing to remember here is that a Windows domain is simply a Windows workgroup with one or more domain controllers added.

Samba can function as a primary domain controller for Windows 95/98 machines without any problems. However, Samba 2.0 can act as a primary domain controller only for authentication purposes; it currently cannot assume any other PDC responsibilities. (By the time you read this, Samba 2.1 may be available so you can use Samba as a PDC for NT clients.) Also, because of the closed protocol used by Microsoft to synchronize SAM data, Samba currently cannot serve as a backup domain controller.

Browsing

Browsing is a high-level answer to the user question: "What machines are out there on the Windows network?" Note that there is no connection with a World Wide Web browser, apart from the general idea of "discovering what's there." And, like the Web, what's out there can change without warning.

Before browsing, users had to know the name of the specific computer they wanted to connect to on the network, and then manually enter a UNC such as the following into an application or file manager to access resources:

```
\\HYDRA\network\
```

With browsing, however, you can examine the contents of a machine using a standard point-and-click GUI—in this case, the Network Neighborhood window in a Windows client.

Levels of browsing

As we hinted at the beginning of the chapter, there are actually two types of browsing that you will encounter in an SMB/CIFS network:

- Browsing a list of machines (with shared resources)
- Browsing the shared resources of a specific machine

Let's look at the first one. On each Windows workgroup (or domain) subnet, one computer has the responsibility of maintaining a list of the machines that are currently accessible through the network. This computer is called the *local master browser*, and the list that it maintains is called the *browse list*. Machines on a subnet use the browse list in order to cut down on the amount of network traffic generated while browsing. Instead of each computer dynamically polling to determine a list of the currently available machines, the computer can simply query the local master browser to obtain a complete, up-to-date list.

To browse the actual resources on a machine, a user must connect to the specific machine; this information cannot be obtained from the browse list. Browsing the list of resources on a machine can be done by clicking on the machine's icon when it is presented in the Network Neighborhood in Windows 95/98 or NT. As you saw at the opening of the chapter, the machine will respond with a list of shared resources that can be accessed if that user is successfully authenticated.

Each of the servers on a Windows workgroup is required to announce its presence to the local master browser after it has registered a NetBIOS name, and (theoretically) announce that it is leaving the workgroup when it is shut down. It is the local master browser's responsibility to record what the servers have announced. Note that the local master browser is not necessarily the same machine as a NetBIOS name server (NBNS), which we discussed earlier.

The Windows Network Neighborhood can behave oddly: until you select a particular machine to browse, the Network Neighborhood window may contain data that is not up-to-date. That means that the Network Neighborhood window can be showing machines that have crashed, or can be missing machines that haven't been noticed yet. Put succinctly, once you've selected a server and connected to it, you can be a lot more confident that the shares and printers really exist on the network.

Unlike the roles you've seen earlier, almost any Windows machine (NT Server, NT Workstation, 98, 95, or Windows 3.1 for Workgroups) can act as a local master browser. As with the domain controller, the local master browser can have one or more *backup browsers* on the local subnet that will take over in the event that the local master browser fails or becomes inaccessible. To ensure fluid operation, the local backup browsers will frequently synchronize their browse list with the local master browser. Let's update our Windows domain diagram to include both a local master and local backup browser. The result is shown in Figure 1-13.

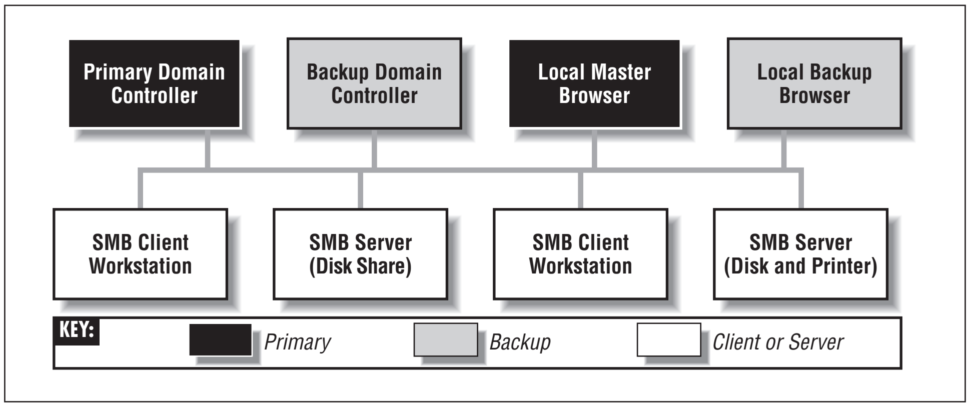


Figure 1-13. A Windows domain with a local master and local backup browser

Here is how to calculate the minimum number of backup browsers that will be allocated on a workgroup:

- If there are between 1 and 32 Windows NT workstations on the network, or between 1 and 16 Windows 95/98 machines on the network, the local master browser allocates one backup browser in addition to the local master browser.
- If the number of Windows NT workstations falls between 33 and 64, or the number of Windows 95/98 workstations falls between 17 and 32, the local master browser allocates two backup browsers.

- For each group of 32 NT workstations or 16 Windows 95/98 machines beyond this, the local master browser allocates another backup browser.

There is currently no upper limit on the number of backup browsers that can be allocated by the local master browser.

Browsing elections

Browsing is a critical aspect of any Windows workgroup. However, not everything runs perfectly on any network. For example, let's say that the Windows NT Server on the desk of a small company's CEO is the local master browser—that is, until he switches it off while plugging in his massage chair. At this point the Windows NT Workstation in the spare parts department might agree to take over the job. However, that computer is currently running a large, poorly written program that has brought its processor to its knees. The moral: browsing has to be very tolerant of servers coming and going. Because nearly every Windows machine can serve as a browser, there has to be a way of deciding at any time who will take on the job. This decision-making process is called an *election*.

An election algorithm is built into nearly all Windows operating systems such that they can each agree who is going to be a local master browser and who will be local backup browsers. An election can be forced at any time. For example, let's assume that the CEO has finished his massage and reboots his server. As the server comes online, it will announce its presence and an election will take place to see if the PC in the spare parts department should still be the master browser.

When an election is performed, each machine broadcasts via datagrams information about itself. This information includes the following:

- The version of the election protocol used
- The operating system on the machine
- The amount of time the client has been on the network
- The hostname of the client

These values determine which operating system has seniority and will fulfill the role of the local master browser. (Chapter 6, *Users, Security, and Domains*, describes the election process in more detail.) The architecture developed to achieve this is not elegant and has built-in security problems. While a browsing domain can be integrated with domain security, the election algorithm does not take into consideration which computers become browsers. Thus it is possible for any machine running a browser service to register itself as participating in the browsing election, and (after winning) being able to change the browse list. Nevertheless, browsing is a key feature of Windows networking and backwards compatibility requirements will ensure that it is in use for years to come.

Can a Windows Workgroup Span Multiple Subnets?

Yes, but most people who have done it have had their share of headaches. Spanning multiple subnets was not part of the initial design of Windows NT 3.5 or Windows for Workgroups. As a result, a Windows domain that spans two or more subnets is, in reality, the "gluing" together of two or more workgroups that share an identical name. The good news is that you can still use a primary domain controller to control authentication across each of the subnets. The bad news is that things are not as simple with browsing.

As mentioned previously, each subnet must have its own local master browser. When a Windows domain spans multiple subnets, a system administrator will have to assign one of the machines as the *domain master browser*. The domain master browser will keep a browse list for the entire Windows domain. This browse list is created by periodically synchronizing the browse lists of each of the local master browsers with the browse list of the domain master browser. After the synchronization, the local master browser and the domain master browser should contain identical entries. See Figure 1-14 for an illustration.

Sound good? Well, it's not quite nirvana for the following reasons:

- If it exists, a primary domain controller always plays the role of the domain master browser. By Microsoft design, the two always share the NetBIOS resource type <1B>, and (unfortunately) cannot be separated.
- Windows 95/98 machines cannot become *or even contact* a domain master browser. The Samba group feels that this is a marketing decision from Microsoft that forces customers to have at least one Windows NT workstation (or Samba server) on each subnet of a multi-subnet workgroup.

Each subnet's local master browser continues to maintain the browse list for its subnet, for which it becomes authoritative. So if a computer wants to see a list of servers within its own subnet, the local master browser of that subnet will be queried. If a computer wants to see a list of servers outside the subnet, it can still go only as far as the local master browser. This works because, at appointed intervals, the authoritative browse list of a subnet's local master browser is synchronized with the domain master browser, which is synchronized with the local master browser of the other subnets in the domain. This is called *browse list propagation.*

Samba can act as a domain master browser on a Windows domain if required. In addition, it can also act as a local master browser for a Windows subnet, synchronizing its browse list with the domain master browser.

,ch01.26764 Page 25 Friday, November 19, 1999 3:27 PM

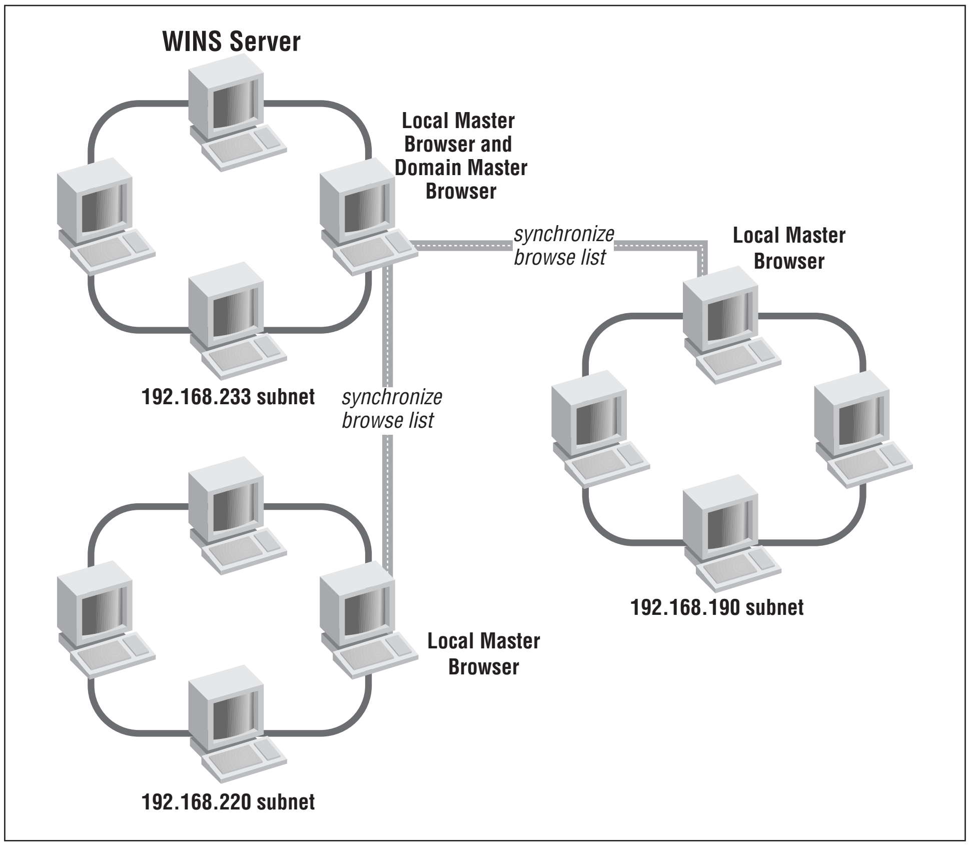


Figure 1-14. A workgroup that spans more than one subnet

The Windows Internet Name Service (WINS)

The Windows Internet Name Service (WINS) is Microsoft's implementation of a NetBIOS name server (NBNS). As such, WINS inherits much of NetBIOS's characteristics. First, WINS is flat; you can only have machines named `fred` or workgroups like CANADA or USA. In addition, WINS is dynamic: when a client first comes online, it is required to report its hostname, its address, and its workgroup to the local WINS server. This WINS server will retain the information so long as the client periodically refreshes its WINS registration, which indicates that it's still connected to the network. Note that WINS servers are not domain or workgroup specific; they can appear anywhere and serve anyone.

Multiple WINS servers can be set to synchronize with each other after a specified amount of time. This allows entries for machines that come online and offline on the network to propagate from one WINS server to another. While in theory this seems efficient, it can quickly become cumbersome if there are several WINS servers covering a network. Because WINS services can cross multiple subnets (you'll either hardcode the address of a WINS server in each of your clients or obtain it via DHCP), it is often more efficient to have each Windows client, no matter how many Windows domains there are, point themselves to the same WINS server. That way, there will only be one authoritative WINS server with the correct information, instead of several WINS servers continually struggling to synchronize themselves with the most recent changes.

The currently active WINS server is known as the *primary WINS server*. You can also install a secondary WINS server, which will take over in the event that the primary WINS server fails or becomes inaccessible. Note that there is no election to determine which machine becomes a primary or backup WINS server—the choice of WINS servers is static and must be predetermined by the system administrator. Both the primary and any backup WINS servers will synchronize their address databases on a periodic basis.

In the Windows family of operating systems, only an NT Workstation or an NT server can serve as a WINS server. Samba can also function as a primary WINS server, but not a secondary WINS server.

What Can Samba Do?

Whew! Bet you never thought Microsoft networks would be that complex, did you? Now, let's wrap up by showing where Samba can help out. Table 1-6 summarizes which roles Samba can and cannot play in a Windows NT Domain or Windows workgroup. As you can see, because many of the NT domain protocols are proprietary and have not been documented by Microsoft, Samba cannot properly synchronize its data with a Microsoft server and cannot act as a backup in most roles. However, with version 2.0.*x*, Samba does have limited support for the primary domain controller's authentication protocols and is gaining more functionality every day.

Table 1-6. Samba Roles (as of 2.0.4b)

Role	Can Perform?
File Server	Yes
Printer Server	Yes
Primary Domain Controller	Yes (Samba 2.1 or higher recommended)
Backup Domain Controller	No
Windows 95/98 Authentication	Yes
Local Master Browser	Yes
Local Backup Browser	No

Table 1-6. Samba Roles (as of 2.0.4b) (continued)

Role	Can Perform?
Domain Master Browser	Yes
Primary WINS Server	Yes
Secondary WINS Server	No

An Overview of the Samba Distribution

As mentioned earlier, Samba actually contains several programs that serve different but related purposes. Let's introduce each of them briefly, and show how they work together. The majority of the programs that come with the Samba distribution center on its two daemons. Let's take a refined look at the responsibilities of each daemon:

smbd
: The *smbd* daemon is responsible for managing the shared resources between the Samba server machine and its clients. It provides file, print, and browser services to SMB clients across one or more networks. *smdb* handles all notifications between the Samba server and the network clients. In addition, it is responsible for user authentication, resource locking, and data sharing through the SMB protocol.

nmbd
: The *nmbd* daemon is a simple nameserver that mimics the WINS and NetBIOS name server functionality, as you might expect to encounter with the LAN Manager package. This daemon listens for nameserver requests and provides the appropriate information when called upon. It also provides browse lists for the Network Neighborhood and participates in browsing elections.

The Samba distribution also comes with a small set of Unix command-line tools:

smbclient
: An FTP-like Unix client that can be used to connect to Samba shares

smbtar
: A program for backing up data in shares, similar to the Unix *tar* command

nmblookup
: A program that provides NetBIOS over TCP/IP name lookups

smbpasswd
: A program that allows an administrator to change the encrypted passwords used by Samba

smbstatus
: A program for reporting the current network connections to the shares on a Samba server

testparm
A simple program to validate the Samba configuration file

testprns
A program that tests whether various printers are recognized by the *smbd* daemon

Each significant release of Samba goes through a significant exposure test before it's announced. In addition, it is quickly updated afterward if problems or unwanted side-effects are found. The latest stable distribution as of this writing is Samba 2.0.5, the long-awaited production version of Samba 2.0. This book focuses on the functionality supported in Samba 2.0, as opposed to the older 1.9.*x* versions of Samba, which are now obsolete.

How Can I Get Samba?

Samba is available in both binary and source format from a set of mirror sites across the Internet. The primary home site for Samba is located at *http://www.samba.org/*.

However, if you don't want to wait for packets to arrive all the way from Australia, mirror sites for Samba can be found at any of several locations on the Internet. A list of mirrors is given at the primary Samba home page.

In addition, a CD-ROM distribution is available in the back of this book. We strongly encourage you to start with the CD-ROM if this is your first time using Samba. We've included source and binaries up to Samba 2.0.5 with this book. In addition, several of the testing tools that we refer to through the book are conveniently packaged on the CD-ROM.

What's New in Samba 2.0?

Samba 2.0 was an eagerly-awaited package. The big additions to Samba 2.0 are more concrete support for NT Domains and the new Samba Web Administration Tool (SWAT), a browser-based utility for configuring Samba. However, there are dozens of other improvements that were introduced in the summer and fall of 1998.

NT Domains

Samba's support for NT Domains (starting with version 2.0.*x*) produced a big improvement: it allows SMB servers to use its authentication mechanisms, which is essential for future NT compatibility, and to support *NT domain logons*. Domain logons allow a user to log in to a Windows NT domain and use all the computers in the domain without logging into them individually. Previous to version 2.0.0, Samba supported Windows 95/98 logon services, but not NT domain logons. Although domain logons support is not complete is Samba 2.0, it is partially implemented.

Ease of Administration

SWAT, the Samba Web Administration Tool, makes it easy to set up a server and change its configuration, without giving up the simple text-based configuration file. SWAT provides a graphical interface to the resources that Samba shares with its clients. In addition, SWAT saves considerable experimentation and memory work in setting up or changing configurations across the network. You can even create an initial setup with SWAT and then modify the file later by hand, or vice versa. Samba will not complain.

On the compilation side, GNU *autoconf* is now used to make the task of initial compilation and setup easier so you can get to SWAT quicker.

Performance

There are major performance and scalability increases in Samba: the code has been reorganized and *nmbd* (the Samba name service daemon) heavily rewritten:

- Name/browsing service now supports approximately 35,000 simultaneous clients.
- File and print services support 500 concurrent users from a single medium-sized server without noticeable performance degradation.
- Linux/Samba on identical hardware now consistently performs better than NT Server. And best of all, Samba is improving.
- Improved "opportunistic" locking allows client machines to cache entire files locally, greatly improving speed without running the risk of accidentally overwriting the cached files.

More Features

There are several additional features in Samba 2.0. You can now have multiple Samba aliases on the same machine, each pretending to be a different server, a feature similar to virtual hosts in modern web servers. This allows a host to serve multiple departments and groups, or provide disk shares with normal username/password security while also providing printers to everyone without any security. Printing has been changed to make it easier for Unix System V owners: Samba can now find the available printers automatically, just as it does with Berkeley-style printing. In addition, Samba now has the capability to use multiple code pages, so it can be used with non-European languages, and to use the Secure Sockets Layer protocol (SSL) to encrypt all the data it sends across the Internet, instead of just passwords.*

Compatibility Improvements

At the same time as it's becoming more capable, Samba is also becoming more compatible with Windows NT. Samba has always supported Microsoft-style password encryption. It now provides tools and options for changing over to Microsoft encryption, and for keeping the Unix and Microsoft password files synchronized while doing so. Finally, a Samba master browser can be instructed to hunt down and synchronize itself with other SMB servers on different LANs, allowing SMB to work seamlessly across multiple networks. Samba uses a different method of accomplishing this from the Microsoft method, which is undocumented.

Smbwrapper

Finally, there is an entirely new version of the Unix client called *smbwrapper*. Instead of a kernel module that allows Linux to act as a Samba client, there is now a command-line entry to load the library that provides a complete SMB filesystem on some brands of Unix. Once loaded, the command `ls /smb` will list all the machines in your workgroup, and `cd /smb/`*`server_name`*`/`*`share_name`* will take you to a particular share (shared directory), similar to the Network File System (NFS). As of this writing, *smbwrapper* currently runs on Linux, Solaris, SunOS 4, IRIX, and OSF/1, and is expected to run on several more operating systems in the near future.

And That's Not All...

Samba is a wonderful tool with potential for even the smallest SMB/CIFS network. This chapter presented you with a thorough introduction to what Samba is, and more importantly, how it fits into a Windows network. The next series of chapters will help you set up Samba on both the Unix server side, where its two daemons reside, as well as configure the Windows 95, 98, and NT clients to work with Samba. Before long, the aches and pains of your heterogeneous network may seem like a thing of the past. Welcome to the wonderful world of Samba!

* If you reside in the United States, there are some federal rules and regulations dealing with strong cryptography. We'll talk about his later when we set up Samba and SSL in Appendix A, *Configuring Samba with SSL*.