Cross-platform file sharing between Unix and Windows

Paul Shields

Abstract

The need to integrate Unix and Windows servers is becoming more imperative as companies deploy Windows NT workstations and servers into Unix-based server environments. Many companies want to combine the best of both worlds, having the application availability of NT and the scalability and security of Unix. There are a number of ways to integrate the systems, each with their own caveats.

1.0 Introduction

The integration of Unix and Windows is more important everyday as Windows 95/NT becomes the pervasive desktop solution and Unix retains its stronghold in the high-end server market. IT organizations must find ways to integrate the platforms allowing seamless access to all data from every platform.

Integration is not always as easy as it sounds and a number of solutions are available. When picking a solution, system administrators must make a conscientious effort to identify their business needs and select a system that meets those needs. Selecting the wrong system will lead down a path of frustration because of the complexity of implementation or the lack of critical features.

2.0 Client versus Server solutions

There are several ways to integrate UNIX and Windows filesystems.

- PC NFS clients
- Windows NT gateway servers
- Unix-based SMB servers

2.1 PC NFS clients

Recently, Microsoft released Services For Unix (SFU), a package of tools that enhance Windows ability to coexist with Unix. Most of the tools are client solutions that provide typical Unix tools like Telnet, NFS, VI, etc. SFU does not provide a way to share NFS mounts to other Windows users. Like all PC NFS clients, SFU is a solution for a single user and administrators must install and configure the software on every PC.

2.2 Windows NT NFS gateways

Another Windows-based solution is a NFS gateway product. These products offer an NFS client that runs on an NT server and mounts NFS share points. The server then shares each mount out to the Windows world. These packages also provide services and tools for meshing NIS and Windows domains. While functional and robust, most of these solutions lack scalability. They either have a limit to the number of NFS mounts possible or internal performance bottlenecks that limit the number of simultaneous users.

2.3 Unix server solutions

The reason for the Unix server focus is that in environments with an established Unix server base, such a solution is more flexible and easier to administer. The exception might be environments with only a few (5 - 10) users who need access to Unix filesystems. In such an environment, the administrative load of managing local PC clients is low.

The Unix SMB market is developing at a rapid rate. Over the last 12-18 months, there have been significant upgrades to TotalNet Advanced Server (TAS) by Syntax, updates to Samba (an open-source solution), new products from a number of Unix vendors, and a growing number of appliance servers (NetApp).

In some cases, the maturity of these products is making them a viable alternative to native Windows NT servers. It is possible with today's solutions to build an "NT" network without a single Windows NT server.

There are three classes of UNIX SMB servers, AT&T Services for UNIX (ASU), Samba/TAS, and Network Attached Storage (embedded processor hybrid systems).

Several years ago, AT&T negotiated a license for the Windows networking code for porting to UNIX. They have since sub-licensed this code to several vendors (DEC, Sun, and Auspex). These systems offer the advantage of being a direct port of the native Windows NT networking code and thus work with all the remote administration tools provided with Windows NT. Administrators can manage these systems like an NT server.

Samba and TAS are UNIX applications and daemons built using publicly available information on how SMB works, combined with packet analysis of the network traffic Windows machines generate when sharing files. This "reverse-engineering" effort means that these packages may have slight differences in functionality and behavior than a native NT server. It also means that standard NT administrative tools will not work. Both applications provide an extensive Web interface for administration.

Network Attached Storage (NAS) is the final alternative. The functionality of these systems varies widely. Devices like the NetApp use a port of native NT networking code and offer features similar to the ASU systems. Others use implementation more like TAS or Samba and rely on Web browser for administration. When selecting a NAS system, test them thoroughly to ensure they meet all the specification and requirements of your environment.

3.0 Why select a Unix server for Windows file services?

The most common reasons to select a UNIX server for SMB file services are:

- Stability
- Scalability
- Performance

3.1 Stability

Unix has a history of development that extends back almost 25 years. During that time, programmers have spent a significant amount of time refining and optimizing the core Unix OS. Thousands of programmers have contributed to the Unix source code base and the result is a stable product that implements many of the features that other commercial OS still lack.

Windows is a relatively immature product compared to Unix; Windows NT 4.0 is only four years old. Over the last few years, NT has come a long way with the release of multiple service packs, numerous hot-fixes, and an impressive array of third-party applications. Each of these has made NT a more viable solution in the server realm, but stability of the core OS still limit its usefulness in many mission-critical environments.

The various Unix (Sun, IBM, HP, etc.) vendors offer solutions that deliver a high-level of reliability and stability. Administrators can further improve stability with high availability options like clustering, fault-tolerance, and redundant hardware. Each of these options has more refinement and is more feature-rich and stable than their NT equivalents.

Another reason for the stability of Unix is the tight integration between hardware and software. Since most Unix vendors sell an OS that runs only on their hardware, they have the ability to tune the interaction between hardware and software.

The tight integration between hardware and software also allows Unix platforms to support advanced hardware features such as internal NIC fail-over, hot-swappable CPUs, hot-swappable expansion cards, and network trunking.

As Windows matures it may catch and surpass the stability of Unix, but the early betas of NT 5/Windows 2000 still lack the focus on stability. At some point Microsoft will realize the importance of stability and make a concerted effort to improve Windows.

3.2 Scalability

Unix also has an advantage when it comes to system scalability. As mainframe-based organizations migrated to Unix, they demanded the same levels of scalability that were previously available on their older systems. We are starting to see the same demands placed on Windows NT servers, but the maturity of the OS gives Unix the lead in overall scalability.

Unix systems scale to support multiple CPUs (up to 64), network interface cards (dozens of cards per server), huge disk arrays, and large memory addresses (up to 64 bits) without trouble. Every major

Unix vendor ships a 64-bit version of their OS that runs in tandem with their 64-bit processors. Windows 2000 promises 64-bit addressing with support for the Merced chip, but this is months if not years away.

Microsoft is still struggling to add advanced SMP capabilities to Windows NT. While 64 CPU systems are common in the UNIX world, it is rare to find a Windows machine with more than four. Windows 2000 Data Center will support up to 16 CPUs, but its release is almost a year away.

Network throughput is a critical measure of a file server. Unix servers can support multiple Fast Ethernet, GigaBit Ethernet, ATM, and FDDI connection. With adequate CPU power, the server can fill these network pipes with massive amounts of data.

Limitations in the current Microsoft TCP/IP stack limit the ability of the OS to scale to a level of supporting multiple network interfaces. Driver support is another issue. Every major Unix vendor ships Quad Fast Ethernet cards for their systems. Such cards are rare in the Windows realm.

Finally, high-end storage solutions are another area of strength for Unix. The drivers and OS are more capable of handling the high data rates of LVD SCSI and Fibre-Channel then their Windows counterparts. Many of the Unix-based drive arrays offer features not common in the NT world such as dual-path architectures and better support for dynamic reconfiguration.

3.3 Performance

In the end, it comes down to system performance. Performance is more than just raw benchmarks but is the overall impact the server has on user productivity.

Configuring a Unix server to out-perform an NT server is a trivial task in most situations for a number of reasons. The large memory address space and efficiency of the OS allow Unix systems to cache a significant amount of data.

No matter how fast your disk subsystem, accessing data from RAM is always faster. In our environment, we expect a typical Unix fileserver to show memory utilization above 95% due to the caching. Why is this not a performance problem as it might be on an NT server? The Unix OS is more efficient at memory allocation and page swapping and can manage large amounts of data cache without affecting performance.

In a small environment, less than 15 - 20 users, it is difficult to show a significant difference between a native NT server and a Unix hybrid. With hundreds of users, the differences are more obvious. Unix threads and scheduling algorithm are more robust and allow a greater number of concurrent processes without the OS becoming a bottleneck.

Even with these advantages, performance is heavily dependent on the configuration of the server, network, and clients. Figure 1 shows data from recent PC Week lab tests demonstrating the high performance capabilities of an SGI running Samba, and a Network Appliance F760.

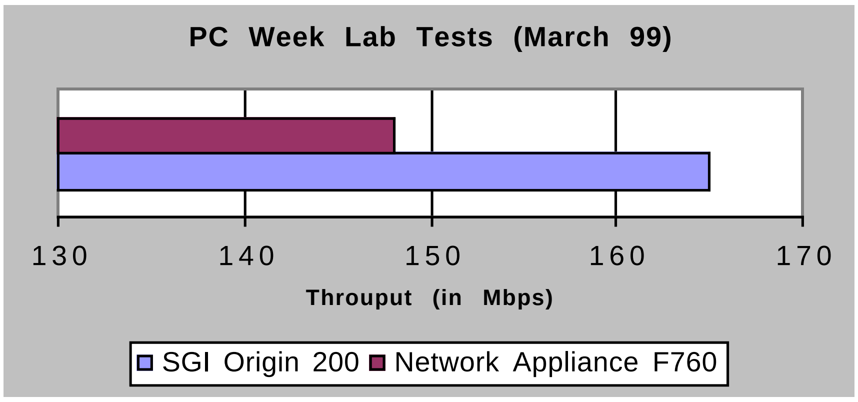

Figure 1 PC Week tests of SGI Origin 200 running Samba 2.0.3. Network Appliance performance listed for comparison purposes. Results demonstrate Samba's ability to scale across multiple processors on high-end Unix machines.

Figure 2 shows the performance comparison of Linux running Samba versus NT on identical hardware. These tests show two things. First that the SMP enhancements in the Linux 2.2 kernel provide significant performance gains. They also show that the Linux/Samba combination offers scalability to match NT on SMP hardware.

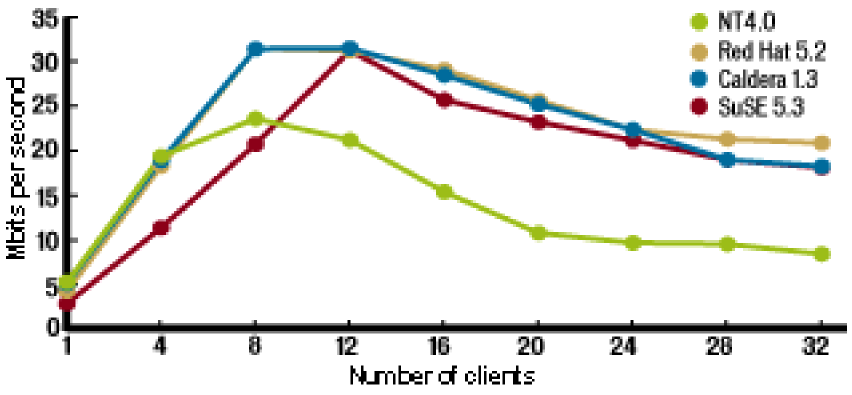

Figure 2 PC Week labs comparison of Windows NT and Linux on similar hardware.

4.0 Drawbacks of a Unix solution

As all system administrators know, no single solution is perfect in every situation. The primary drawbacks of Unix are:

- Costs
- Hardware and software availability

4.1 Costs

Windows NT has the advantage of being an OS built on commodity hardware. Dozens of vendors offer a variety of NT server solutions. These realities make NT cheaper for smaller workgroup class servers. On high-end systems, some of this price advantage disappears, but there are still discrepancies.

Every implementation must balance the constraints of costs, performance, and features. A Unix based solution for a small workgroup that requires hardware investment may not make sense. An NT server may provide the functionality at a lower cost. In environments where stability, scalability, and performance are tantamount to success, then a Unix solution may be the only viable solution.

4.2 Hardware and software availability

The general problem is not availability of core components like CPU, RAM, and disk subsystems but the more esoteric products like X.25, serial ports, and other emerging technologies. The huge marketshare difference between Windows NT and Unix draws developers who perceive a smaller risk in going with a high-volume platform.

Most of these components are not critical for general file servers, but may be an issue if plans include using the server as application server.

As with hardware, software availability for Unix lags behind the Windows platform. Again, the core functionality is there, but smaller niche products may be either non-existent or difficult to find.

With the relatively recent emergence of Unix-based Windows file servers, there is also a lack of monitoring and administrative tools. Administrators with a background in Unix support will not view this as a hindrance because they have an arsenal of tools they already use. For those new to the Unix, this is a little overwhelming. This can be a difficult transition for administrators with an NT background accustomed to the variety of disk, domain, performance, and system monitoring tools with GUI interfaces.

5.0 The technologies at work

5.1 SMB

System Message Block (SMB) is the standard communication method for transferring data between Windows systems.

In the early 80's, Microsoft, IBM, and 3Com worked to define a series of protocols for sharing files and printers over a local area network. The result of these efforts was version 1.0 of LanManager, a file and print server for DOS. LanManager evolved over the years to the point where it became the method for sharing files over a number of protocols. The focus of this paper is SMB over TCP/IP since that is the common protocol in most LAN/WAN environments.

5.1.1 SMB basics

SMB is a connection-oriented protocol that operates at the application layer of the OSI model as shown in Figure 3. The NETBIOS Session layer protocol handles binding of network names to TCP/IP addresses, as specified in the RFC 1001 and RFC 1002. NETBIOS names are up to 15 characters long. When a client wants to open a connection to a server, it first does a lookup on the NETBIOS name. NETBIOS uses WINS, LMHOSTS files, or a network broadcast to resolve names to IP addresses.

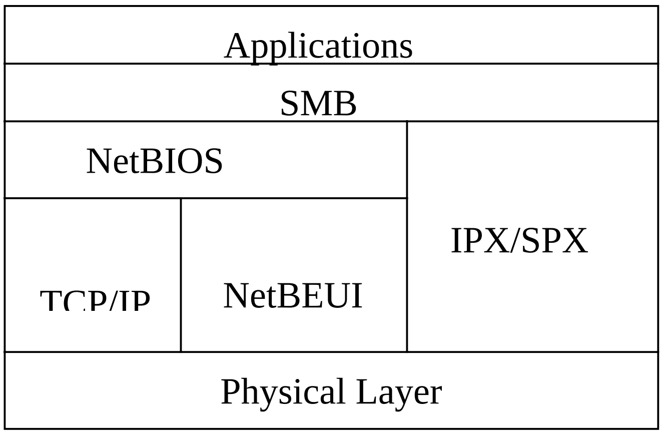

Figure 3 OSI model representation of SMB. This model shows that SMB sits just below the application layer with other protocols handling Name resolution and connection.

In contrast, NFS is a stateless connection. This allows NFS to recover when a server fails. NFS will retransmit the packet until it receives an acknowledgement from the server. The NFS protocols resolve machine names by either using Domain Name Services (DNS) or a local hosts file.

5.1.2 Resolving NETBIOS names

Windows Internet Name Service is a function similar to DNS in the Unix world although, by nature WINS address tables are dynamic. Recent modifications to the DNS specification (RFC 2136) enhance DNS by allowing for dynamic DNS.

Microsoft labels any client configured with WINS server information as WINS-enabled. Most of the current Unix SMB servers support WINS registration.

When a WINS-enabled client boots, it contacts the WINS server and registers its current computer name and IP address. The WINS database provides support for registering multi-home hosts and special entries for domains and domain controllers.

WINS-enabled clients use a name resolution method called h-node broadcast. This Microsoft defined term is a hybrid (server query and broadcast) method for name lookups. When a client wants to resolve NETBIOS names, it first queries the WINS server. If the WINS server does not have a record for the NETBIOS name, the client will broadcast on the local subnet. If the machine is on the same subnet, it will respond with its IP address.

Unlike DNS, the NETBIOS naming structure is flat. There is only one machine with a given NETBIOS name registered in the WINS server. This means there is no organizational hierarchy built into WINS making it difficult to organize and manage computer names in large organizations. Windows domains are simply a logical grouping and not an integral part of the machine address as in a DNS domain.

Windows NT can optionally use a DNS server as a lookup service. You lose the dynamic nature of WINS, however, and you should continue to follow the 15 character naming structure to maintain compatibility with WINS dependent clients. Most administrators recommend that you keep the computer name (NetBIOS/WINS) the same as the host name (DNS).

Another method for name lookups is a local LMHOSTS file. Similar to the hosts file in Unix, the LMHOSTS file allows an administrator to define a list of servers in a local file. This reduces the reliance on network lookups. There is a limit to the number of entries an LMHOSTS file can have and there are the usual problems of keeping a local file updated.

5.1.3 Cruising the neighborhood

As part of the changes made to the SMB specification in Windows 95, Microsoft added support for network browsing. Network browsing services attempt to provide an easy to use graphical interface for finding network services. Browsing is complicated and fragile by nature. Using network browsing in a large enterprise can be frustrating for users due to the delays associated updating the browse lists.

Browsing is somewhat hierarchical and based on a broadcast registration process. In a domain environment, the primary domain controller becomes the Domain Master Browser.

For each subnet, there is a Master Browser. The Master Browser for a subnet is chosen via an election process. When a Windows machine boots, it broadcast on the subnet looking for a Master Browser with which to register its name. IP address, and workgroup/domain. If it fails to locate a Master Browser, the machine sends out a packet that starts an election process.

There is a preferred order to deciding which machine wins the election. The selection preferences in descending order are Windows NT Server, Windows NT workstation, Windows 98, Windows 95, and WFW 3.11. Each machine looks at the packet and decides if it has precedence over the previous machine. When the packet gets back to the originator, there will be a winner. In the Windows NT world, registry changes allow you to control whether the machine will become a Master Browser. Unix implementations that support browsing also offer ways to configure the server's response to election packets.

Once elected, the Master Browser maintains a list of all machines on the subnet. The Master Browser passes this list to the Domain Master Browser of the machine's Domain. The Domain Master Browser builds the entire list and sends it back out to the Browse Master on each subnet.

When a machine is properly shutdown, it sends a packet to the Master Browser informing the server to update its list. When a machine crashes or is not shutdown properly, the record will remain until it times out (approximately 10 minutes). This can cause confusion since a machine will remain in the browse list until it times out. When the user double-clicks on the computer, they get a generic non-descriptive error message.

There is some redundancy in the browsing mechanism with an election for a backup Browse Master on each subnet. Worse case scenarios prevent a machine from appearing in the browse list for up to 57 minutes. Whether you have the Unix SMB server become a Browse Master will depend on the stability of your network clients. If there are NT servers or workstations on each subnet that are stable than you will be fine. Otherwise, the choice is to either enable the Browse Master functions of the server or live without the browsing capability.

5.1.4 Establishing the connection

Once the name is resolved, the client sends a connection request to the server. Depending on the access controls placed on the resource, the server may send back a request for authentication. When a Windows client logs onto the network at startup, the system caches the user ID and password and provides them for logon validation when requested.

Before the release of Service Pack 3 for Windows NT, authentication used a clear-text format to transmit the password. Windows NT 4 SP3 and Windows 98 now use encrypted passwords. While it is possible to change these machines back to clear text passwords, this is not a recommended configuration. All the Unix SMB servers support encrypted password authentication.

If the password is not cached or is incorrect, the user will be prompted for a username and password (Under Windows 95, the user will only be prompted for a password because the system always uses the cached username). Once authenticated, the server opens a connection to the client. The client then makes requests like open file, lock file, close file, change attributes, etc.

Windows uses domai8n to centralize account administration and authentication. The method for assigning access privileges recommended by Microsoft is to create Domain accounts and place them in Domain groups. On each member server you then create local groups and place the domain groups inside the local groups. You then use the local groups to assign access to resources on the server. When a client accesses the server, it sends its domain account and password, which the member server passes to a Domain Controller for authenticates.

The process for handling this in Unix depends on the server installation and network configuration.

Unix servers offer more flexibility on authenticating the user that may prove advantageous in some environments. The default method in most systems is to authenticate the user to the local password file (or NIS).

Another option is what most vendors call proxy authentication. This means that the server passes authentication to another server. When the Windows client connects, the server sends the ID and password to the proxy server for authentication. The proxy server responds with an acknowledgement of whether or not the user ID and password matched. Most packages can authenticate to either another Unix machine or a Windows NT server (PDC, BDC, or member server).

For non-AT&T Services for Unix systems (e.g. TAS or Samba), this method has more complications. An ASU-based system understands Windows Domain and intrinsically knows how to communicate with the Domain Controller. The advantage of this method is better support for redundancy of Domain Controllers. When a machine wants to authenticate, it queries the WINS server for a list of Domain Controllers. It will then work through this list starting with the closest one, to find a server that responds. If a particular Domain Controller is down, the client goes to the next server in the list.

For systems that are not Domain-aware, the administrator must explicitly define the IP addresses or nodenames of the Domain Controllers. Most implementations do not provide for more than one or two entries. If the communications link between the servers breaks the Unix SMB server will deny any new connections because it cannot authenticate the users.

In most environments, two Domain Controllers provide the stability and reliability needed, but in large organization with highly distributed Domain environments, this can be an issue.

Authentication is only the first step to gaining access. Once connected, users must understand and deal with the issues relating to mapping of permissions between the Unix and Windows world.

6.0 Cross-platform issues

6.1 File locking

To successfully implement and manage a hybrid server, administrators must understand how the server handles file locking across environments. Unix file locking mechanisms are very different from Windows and this is a common cause of frustration.

Some applications that run seamlessly on multiple platforms like Adobe FrameMaker support application driven locking and this is the best solution. In the case of Frame, the application creates a lock file in the same directory as the opened file. If another instance of FrameMaker attempts to open the same file, it will detect the lock file and let the user know the file is already open.

Many other applications support internal file locking but most are not cross-platform. There is no application driven method to keep a Unix user from using VI to open a file that a Windows user has open in Notepad.

6.1.1 A lack of communication

In most Unix SMB implementations, there is no method for the Unix-based lock daemon to communicate with the Windows locking mechanism.

The only systems that promise a combined lock manager are servers commonly referred to as Network Attached Storage (NAS) such as the Network Appliance Filer. Even these systems have weaknesses and do not provide a high level of integration. Without this layer of communication, there is no authoritative lock database.

6.1.2 Two different languages

The different locking features of SMB and NFS are the root of the problem. Administrators make assumptions that are not valid across platforms.

In both SMB and NFS, there are two basic functional locks, open modes (file lock or flock under NFS) and record locking. Open modes control concurrent access to entire files, such as in the following example:

> Program A, when opening a file, specifies an open mode, which tells the server what kinds of access by other clients and programs that A will allow. This can be as extreme as not allowing any other access (exclusive use by A), allowing any access of any type, or somewhere in between, e.g., allow other opens for read-access only.

Record locks (called POSIX locks in Unix) control concurrent access to sections of open files. For example:

> Program A opens a file, specifying the system to allow other "opens" of any access. At some point, A wants to ensure that a "record" in the file belongs exclusively to A, so it requests a lock on the range of bytes in the file that represent that record. While A holds the lock, no other programs or clients that open the file can read or write to the locked range.

Programs such as word processors usually use an open mode to control access and ensure that no other application modifies the file. Programs that maintain databases must allow multiple users to update a file and use record locks to ensure that only one user at a time updates a given record.

Unix deviates from Windows at this point. In an NFS environment, locks are advisory. That means they are a convention and applications do not have to check for or create a lock when opening a file. While locked, other processes that do not check for existing locks can still read or write to the file.

In SMB, locks are essentially mandatory. SMB is a connection-oriented protocol and part of that protocol relies on using locks to maintain server connections.

There is the concept of mandatory locks in NFS, but the implementation of mandatory locks is not universal. In addition, few applications take advantage of mandatory locking when it is available.

SMB has another feature called opportunistic locking that widens the gap between the environments even further.

6.1.3 Opportunistic locking

Opportunistic locking (Oplock) is a very aggressive form of local file caching. There is no equivalent of opportunistic locking in NFS (depending on your perspective of NFS V3 and AFS this may not be completely true).

If an application opens a file in a non-exclusive (deny none) mode, the redirector requests an opportunistic lock of the entire file. As long as no other process has the file open, the server will grant this oplock, giving the redirector exclusive access to the specified file. This will allow the redirector to perform read-ahead, write-behind, and lock caching, as long as no other process tries to open the file.

When a second process attempts to open the file, the server asks the first application to Break Oplock or Break to Level II Oplock. At that point, the redirector must invalidate cached data, flush writes and locks, and release the oplock, or close the file.

Opportunistic Locking level II, provides a method for granting read access to a file by more than one workstation, and these workstations can cache read data locally (read-ahead). As long as no station writes to the file, multiple stations can have the file open with level II oplock.

Opportunistic locking has risks. If the implementation of opportunistic locking is not rock-solid, and some administrators question the stability of current Windows implementations, one client can update a file without the server properly notifying other clients to flush their cache. This creates a situation where two applications successfully write to a file, resulting in unexpected data loss..

Unix and Windows clients are working in separate worlds when it come to file locking and it is easy to create a situation where users on both platforms are modifying the same file without protecting who has write access.

6.2 File naming conventions

If Unix, Windows NT, and Windows 95 are the only platforms in use, then filename mapping is not a major problem. Both systems support similar features and filename restrictions.

The only difference is that Unix is case sensitive and Windows NT and 95 are not. Windows NT will preserve case when copying or moving files though. Table 1 shows an example of how TotalNet Advanced Server from Syntax handles file name collisions. As can be seen from this example, the primary problem is with older DOS and Windows 3.11 clients and not with the combination of NT, 95, and UNIX.

Client	Filename
Windows 95, and NT	ReallyLongFile.name ReallyLongFile2.name
Dos, Windows for Workgroups, NetWare	REALL~18.NAM REALL~4C.NAM
UNIX ls	ReallyLongFile.name ReallyLongFile2.name

Table 1 an example of how TotalNet Advanced Server handles mapping filenames across platforms. Windows 95, NT and UNIX have similar restrictions on filenames, although NT is not case sensitive.

There are two minor usability issues to keep in mind. First, Windows users have a tendency to use spaces as part of a filename. This makes it difficult to work with the file at the command line in Unix. While not a burden, it is a nuisance. The second thing to remember is that Windows maps files to applications based on the three-letter extension (.txt, xls, .doc, etc.). Unix users should take care to preserve these extensions and add them where appropriate when creating new files.

With DOS, Windows 3.1, Macintosh, and NetWare clients filename mapping is significantly more complicated. Few applications support all platforms having simultaneous access and the method of handling filename mapping varies. When building a multi-platform server, administrators should evaluate how the server software will map filenames and ensure that the appropriate documentation is available to the user.

6.3 Permissions

When configuring a fileserver, proper security is a major concern. Security is more than just keeping out the latest Internet attackers, but also providing a mechanism to control file and folder access. Both Windows and Unix support advanced capabilities like Access Control Lists, but their implementation of permission, inheritance, and the fundamental logic of access control differ. Similar to file locking, few systems offer integrated permissions capability.

6.3.1 Two methods of control

When selecting a Unix SMB server, administrators have two methods of access control from which to choose. Servers like Samba and TAS, use the Unix file permissions and access control lists. Neither package maintains NT style Access Control Lists. In order to determine if a user should have access to a particular file or folder these systems map Windows accounts to a UNIX ID. The default mapping is by name, but administrators can define username maps for those whose Windows and UNIX IDs do not match.

TAS and Samba lack any knowledge of Windows domain groups. This means that administrators manage groups in the Unix realm. The problem in a mixed platform environment is that administrators and users must maintain two group databases, Unix and Windows.

The alternative is a system based on the AT&T Services for Unix (ASU) code, which understands Windows permissions and Access Control Lists. This lets administrators set permissions based on Windows domain accounts and groups. There is no mapping between worlds with each side maintaining unique sets of permissions. Again, this makes cross-platform file sharing difficult. When Windows users set file permissions, there may be no adjustment to the Unix permissions and the user may not get the desired result.

Some solutions attempt to integrate the two permission systems but most are simplistic in nature. Typical behavior is to map to the lowest common denominator. When a user updates the permissions of a file in the Windows world, the software will attempt to update the owner and group permissions on the UNIX side. Most systems do not map the entire Access Control List between environments.

The advantage of the Samba and TAS method is that there is one set of permissions. The problem is that these systems provide no graphical Windows tools to administer permissions. While not a loss to Windows 95 users who lack access to permissions even on a native NT server, NT users will be disappointed when they go to the Security tab, click Permissions, and get an error message.

Users must telnet to the server and use traditional Unix utilities to set file permissions. There is an effort underway to enhance Samba to support setting the ACL from a Windows machine. Once developed, these systems will offer the advantage of a single permission set and access to them from both worlds.

6.3.2 A different way of thinking

When setting file and folder access, Unix users tend to think at the file level, leaving top-level directories more open. However, in the Windows world the mindset is to secure the folder and not worry about individual file permissions. You will see this in the basic structure of the Windows NTFS file system. By default, the "Everyone" user has full control on the entire file system. To secure a directory, the most common solution is to take the "Everyone" user out of the permissions for the folder. This keeps the user from accessing the directory and makes individual file permissions less significant.

Each method has advantages; the primary one for the Windows model is that it limits the number of files/folders that need explicit permissions. Neither method is inherently better but the transition will require some user training.

6.3.3 Who gets what by default?

This is the most significant permission issue, administrators will encounter when setting up a non-Windows SMB server. In Unix, the method for assigning default permissions is the umask. The umask defines default permissions for new files and folders. The owner and group are set based on the UID and GID of the process creating the file or folder. The problem with the umask method is that it has no concept of Access Control Lists. Setting owner and group permissions by default may not be enough.

The second problem is that none of the current Unix implementations offers a personal umask. There is a single umask for the entire server or share point. The confusion this creates is that users want individual control over the default permissions of files/folders.

In the Windows realm, users expect files and folder to inherit permissions from the enclosing folder. They set the permissions on the folder and when they or anyone else places a file or folder in the directory, it inherits those permissions.

There are a few options on some Unix platforms to simulate this behavior such as setuid and setgid, along with default ACL permissions, but they require a significant amount of Unix administrative knowledge and still lack one crucial feature. If Mary creates a file in John's directory, the owner of the file will default to Mary. If the system umask is restrictive and the setgid bit is not on, John may wind up with a file, which he cannot access or delete.

ASU-based products resolve this problem for Windows permissions because they behave like a native Windows NT server. There is still an issue with default Unix permissions.

6.4 File attributes

NFS and SMB have a slightly different set of file attributes they store. While this does not create any inherit compatibility issues, it does affect how administrators must handle backups and restores.

Since most systems are hybrid in nature and the underlying filesystems (UFS) do not support SMB attributes, they must store these attributes in "shadow" files. These are usually hidden "dot" files in the same directory as the file with the SMB attributes.

This means that when an administrator does a restore, they may need to restore the shadow file to retrieve all the attributes. Whether this is critical depends on the value of these attributes. The SMB attributes stored include creation date and property bits like locked and archived.

For appliance type server like the Network Appliance Filer, the problem is slightly different. The NetApp supports SMB attributes as part of its native filesystem. The problem is that because of their good NFS performance and inability to support a local backup client, many sites backup the servers over NFS. Such a backup configuration fails to capture the SMB attributes. The alternative is to backup over CIFS, to backup to a local tape drive, or to use NBMP.

7.0 Tuning performance

SMB does not offer administrators the same flexibility and control over performance as NFS does. NFS allows administrators to control read/write sizes, timeouts, retries, and a number of other parameters. None of these is accessible in an SMB environment.

There are two basic ways to tune performance.

- UNIX server tuning
- Multi-homing with WINS

7.1 UNIX server tuning

Any change that enhances the underlying filesystem, network, or processor performance will enhance SMB server performance. A few basic configuration parameters will significantly enhance overall system performance.

RAM

Increasing the amount of RAM will have an immediate effect by allowing the server to cache more data, improving read performance. On systems with large numbers of users, the general recommendation would be no less than 1 GB of RAM.

SMP

Adding multiple CPUs will work as long as the server software is threaded and scalable. Before adding CPUs, ensure that the SMB server application creates an individual thread for each connected user, and does not have any ties back to the parent thread that might cause sever bottlenecks under load.

Filesystems and disk subsystems

Filesystem performance tuning is another area where administrators can enhance overall server performance. Which parameters to change depend on the exact filesystem. Refer to the system documentation for information on tuning a particular filesystem. One thing to watch for is journaled filesystems. Such filesystems have advantages but also have a significant amount of overhead. Most offer parameters to control the level of jounaling and administrators should explore thee options.

The final piece to look at is the hardware controller on the disk subsystems. Systems that lack NVRAM for caching suffer in performance comparisons. NVRAM is basically a write-ahead cache that store disk writes and allows system calls to return immediately instead of waiting for the write to disk to complete. This allows writes to occur asynchronously and improves overall performance. NVRAM has a significant impact on write performance and almost no impact on read performance. Administrators should avoid systems without NVRAM unless the vendor clearly demonstrates that there is no performance disadvantage.

7.2 Multi-homing with WINS

Windows Internet Naming Service (WINS) supports registering multiple IP addresses under a single computer name. When a client sends a name lookup request, the WINS server returns the entire list of IP addresses. The client starts with the "closest" network interface when establishing a connection.

With minimal effort and expense, administrators can add enough network interfaces to a server to allow for connections on every subnet. Clients automatically route to the interface on their subnet. Doing this distributes traffic across multiple cards, reduces the load on network routers, and improves reliability by removing one of the failure points.

8.0 Summary

Any one of the commercial products or the freeware Samba fulfills the role as an alternative to a native Windows-based fileserver. ASU-based systems offer the advantage of integrating tightly into existing Windows NT environments and a growing number of companies are licensing and supporting this code.

Solutions like TAS and Samba offer the advantage of independence from the ASU software license. The ASU code license only covers Windows NT 4 networking code. Companies using ASU code lack the experience to re-engineer the Windows protocols and may be at a loss when it comes time to upgrade and support Windows 2000.

The non-ASU solutions still struggle to catch up to NT in some ways but generally offer better performance and scalability.

If you have an existing Windows infrastructure and want to integrate existing Unix servers, TAS and Samba are the fastest and easiest alternative. If you need tight integration between Unix and NT, you will probably be more interested in an ASU solution. The problem is that only a few Unix vendors offer an ASU solution for their OS, and there is no generic provider of such a solution.