HP CIFS Server and Terminal Server

Version 1.06 October, 2007

SNSL Advanced Technology Center

E0300

Printed in: U.S.A.

Legal Notices

The information in this document is subject to change without notice.
Hewlett-Packard makes no warranty of any kind with regard to this manual, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material.

Warranty. A copy of the specific warranty terms applicable to your Hewlett-Packard product and replacement parts can be obtained from your local Sales and Service Office.

Restricted Rights Legend. Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Hewlett-Packard Company
19420 Homestead Road
Cupertino, California 95014 U.S.A.

Use of this manual and flexible disk(s) or tape cartridge(s) supplied for this pack is restricted to this product only. Additional copies of the programs may be made for security and back-up purposes only. Resale of the programs in their present form or with alterations, is expressly prohibited.

Contents

Chapter 1 Introduction

Many organizations host file server and print server services on HP CIFS Server and Samba open source servers, usually running on UNIX or Linux operating systems. Client access to these services is typically achieved by direct network connectivity from the client to the server. However, client access can also be hosted and consolidated on a Windows Terminal Server.

A Terminal Server can be thought of as a client application and connectivity hub. A Terminal Server can host applications, and also connect to system resources on other servers (and operating system platforms), and export these services to remote clients. A client can then connect to the Terminal Server, run applications, mount remote shares, and utilize the Terminal Server processing power with little resource requirements on the remote client itself.

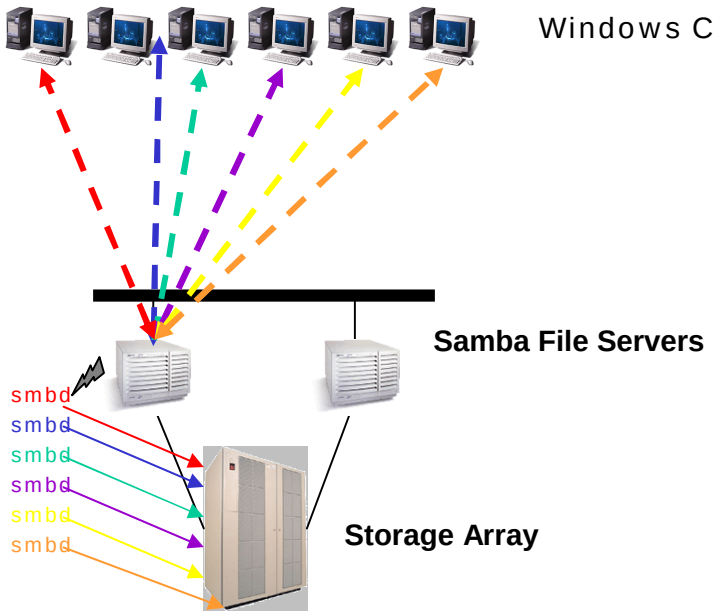
Like any other hub, the Terminal Server must have enough processing power and network throughput to accept the level of client requests, process them, and/or distribute the requests to the destination networked resources. Clearly, an effective Terminal Server usage design cannot exist on a constrained processing platform or network. For example, if a user base of 20 clients in a gigabit Ethernet LAN segment access a Terminal Server and then connect shares to a file serving platform on an overloaded 10 base-T network, the network bottleneck from the 10 base-T LAN segment will constrain client connectivity to the back-end file server and could result in visible performance degradation when accessing resources from that machine.

Windows NT4 Terminal Server integrates effectively with HP CIFS Server and Samba. However, operating system changes in Windows 2000 and Windows 2003 have caused a bottleneck scenario that can constrain HP CIFS Server and Samba connectivity integration with Terminal Server on these operating system platforms. A Microsoft hotfix addressed the problem for Windows 2000. Recently, the Windows 2000 fix has been incorporated into Windows 2003. It is important to understand the source of the Terminal Server Windows operating system bottleneck, and then consider ways to workaround it on HP CIFS Server, Samba, and Windows for those cases where the hotfix cannot be installed.

HP CIFS Server is based upon Samba open source, and the two product names are used interchangeably in this document.

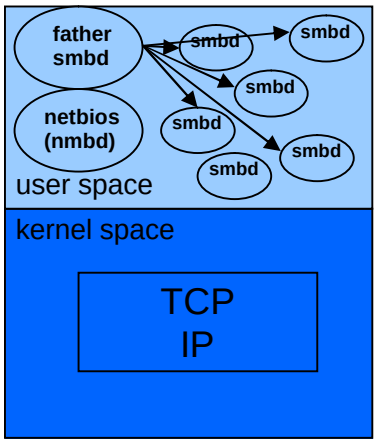
Chapter 2 Samba and Terminal Server Integration

The fundamental Samba design is to manage each client connection to the server with a discrete user process called a smbd daemon. During the client session setup to the Samba server the father Samba process starts the smbd from an incoming client TCP/IP session connection. Thus, for every client that has mounted one or more shares, there exists a smbd process.

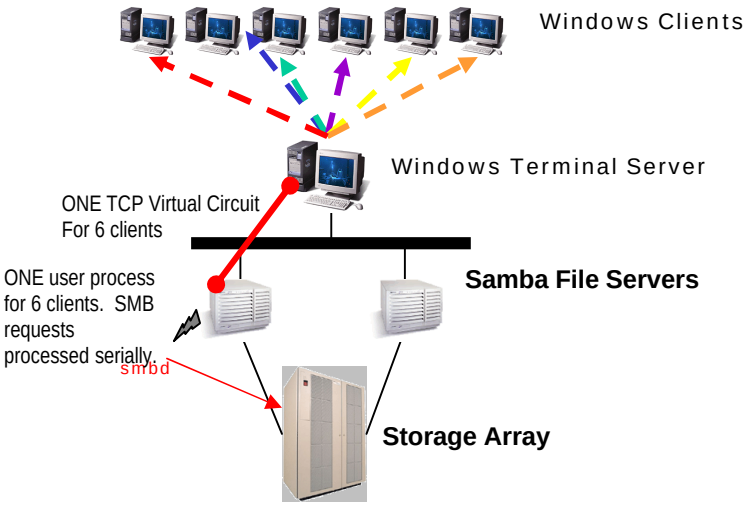


Each Windows client connection to the Samba Server starts as a TCP/IP session (after whatever name resolution is required) on the Samba server, and the TCP session is serviced by a new smbd for the incoming client request. The end effect is that a unique smbd daemon exists for each client connected to the server. Thus, if 1000 clients are connected, 1001 smbd daemons will be present on the system (one process is the smbd father process).

Samba Server



Naturally, the expectation of Terminal Server is that the six remote client connections and subsequent share mounts to the Samba server will result in 6 separate TCP/IP connections, resulting in the expected 6 smbd process to service each virtual client. However, Terminal Server does not operate in the expected manner. Instead, Terminal Server relies on the underlying Windows operating system to establish the transport for the client pool, and Windows will only issue one TCP/IP connection to the remote server – in this case Samba. This results in all six virtual client sessions and share mounts being multiplexed over a single TCP/IP transport pipe to the Samba server. More importantly, only one smbd user process is started on the Samba server, and all 6 client sessions are multiplexed on the single smbd.

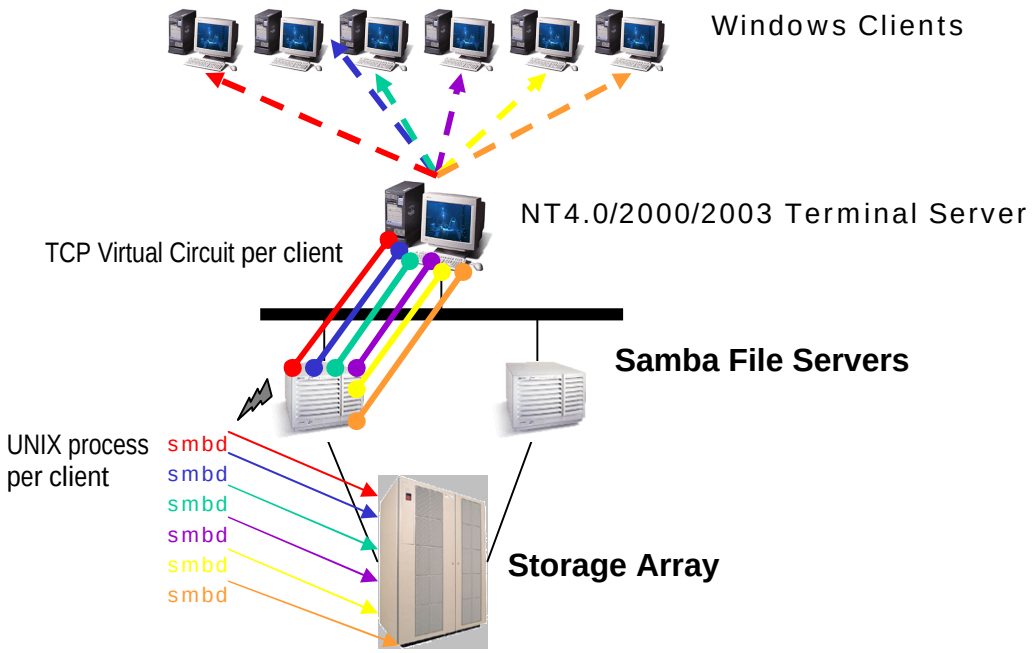


The smbd user process will serially service multiple incoming client requests from multiple clients – in this example six clients. The smbd process is single-threaded, and therefore will block processing on 5 clients while one client is serviced, then service the next client in round-robin priority (sequentially). In most cases this scenario will produce some level of performance degradation, depending upon the request load generated from the client connection pool.

Chapter 3 Samba with TS on Windows NT4/2000/2003

Terminal Server on Windows NT4, 2000, and 2003 is configurable to allow the underlying Windows operating system to appropriately handle multiple incoming client connections for Samba (or other) servers. It is configurable via the *MultipleUsersOnConnection* registry parameter on the Terminal Server NT4 OS platform, the *EnableMultipleUsers* parameter on the Terminal Server Windows 2000 platform, and the *MultiUserEnabled* parameter on the Terminal Server Windows 2003 OS platform. For Windows 2000, as of August 2004, the *EnableMultipleUsers* parameter is only accessible after installing the Microsoft Hotfix from Q818528. For Windows 2003, as of February 2006, the *MultiUserEnabled* parameter is only accessible after installing the Microsoft Hotfix from Q913835 (included in ServicePack1).

MultipleUsersOnConnection is described in the Microsoft Q article Q190162, "Terminal Server and the 2048 Open File Limitation." As implied in the title, the registry parameter was actually created to address a limitation on the number of file handles that a Terminal Server session could utilize, but the end result was the establishment of unique virtual circuits (TCP/IP connections) for individual client connections. This behavior provided exactly the functionality that Terminal Server clients required to efficiently mount Samba file server services, and resulted in widespread usage in the Terminal Server user community for this specific purpose.



With the NT4 registry parameter *MultipleUsersOnConnection* , or the Windows 2000 *EnableMultipleUsers* set to 1 (enabled), or the Windows 2003 *MultiUserEnabled* set to 1 (enabled), the Samba server acknowledges a discrete TCP/IP connection request for each unique Terminal Server client, and therefore starts a new smbd user process to service each client. This

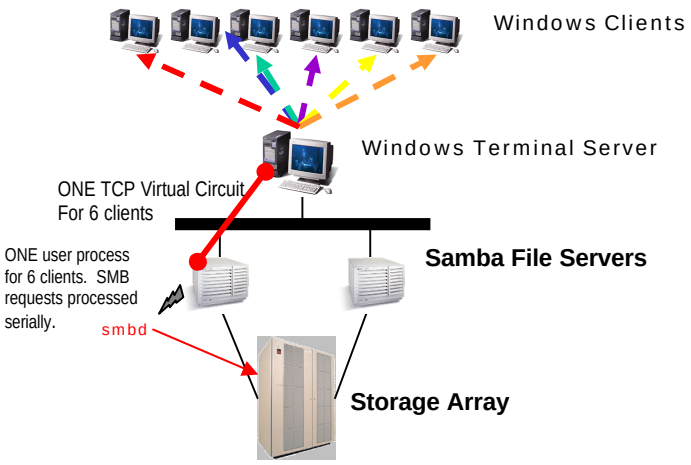
behavior provides the system resources per client connection that Samba was designed for, and thus Samba performance for Terminal Server connections is consistent with standard client sessions (note that Samba performance does not account for the actual Terminal Server system resources, which may be constrained due to the nature multitasking numerous client connections on one host).

Listed below are the Windows server versions, the parameter names, and their associated Q articles that include instructions for implementing the feature on Windows Server OS Platforms.

Windows NT4	*MultipleUsersOnConnection*	Q190162
Windows 2000	*EnableMultipleUser*	Q818528
Windows 2003	*MultiUserEnabled*	Q913835

Chapter 4 Without the Hotfix

If the Windows Terminal Server is not configurable with the hotfixes listed in Chapter 3, the resulting Terminal Server functionality of no configurable option for multiple TCP transport sessions renders the Samba server default configuration behavior incapable of starting more than one smbd user process. Thus, the single smbd must service all incoming client connections from a particular Terminal Server, resulting in potential performance degradation.



Levels of performance degradation (if any) are entirely dependent upon user load and environment variables. Heavy client processing can exhibit noticeable degradation with only 2 Terminal Server users. However, user feedback has proven that some installations have run as many as 100 Terminal Server clients on a single smbd, with adequate performance (this level of usage is not recommended). This particular usage scenario experienced a non-performance resource limitation, which is addressed below.

The single smbd process servicing multiple users and associated IDs will correctly handle switching permissions and access rights for shared resources on the Samba server.

Potential workarounds to Terminal Server integration issues are identified in the next chapter.

Chapter 5 Terminal Server Workarounds

There is no easy way to generate a new TCP/IP connection for every Terminal Server client that connects to a back-end file server. Interestingly, multiplexing numerous discrete connections over a single TCP/IP pipe (the default Windows behavior) has potential reliability issues by itself. Potential workarounds for Samba and Terminal Server integration exist primarily on the Samba platform and name resolution mechanisms.

5.1 NetBIOS Aliases

Terminal Server users identify Samba servers by their NetBIOS names. The underlying Windows operating system uses the Samba server NetBIOS name to uniquely identify the server and find it. Then it negotiates the connection protocol, sets up the session, and connects to the requested service. During the session setup the TCP/IP session from the Terminal Server to the Samba server is established (or multiplexed for existing connections on the TCP session). There are several methods of defining NetBIOS aliases.

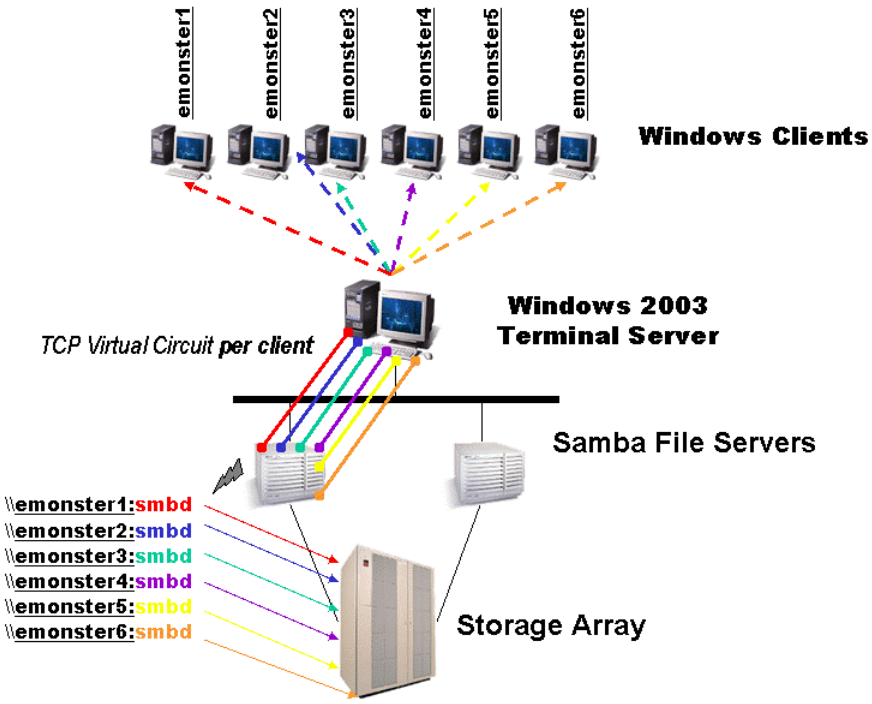
5.1.1 Samba NetBIOS Aliases

The Samba smb.conf parameter “netbios aliases =” allows the creation of multiple NetBIOS pseudonyms for the Samba server. Each NetBIOS pseudonym is a NetBIOS legal name for the Samba resources that are shared by the server. Therefore, when multiple Terminal Server users mount a Samba service, referring to multiple unique NetBIOS names for the server will effectively allow for the generation of a separate TCP/IP connection for the Terminal Server operating system (Windows 2003 included) for each unique NetBIOS name. This provides for the distribution of the client session load over multiple TCP sessions and associated Samba smbd processes.

An smb.conf example looks like:

```
[global]
    workgroup = SNSLATC
    netbios name = EMONSTER
    netbios aliases = emonster1 emonster2 emonster3 emonster4 emonster5 emonster6
    server string = Samba Server
    security = DOMAIN
    encrypt passwords = Yes
    password server = *
    username map = /etc/opt/samba/usermap.txt
    log level = 3
    syslog = 0
    log file = /var/opt/samba/log.%m
    name resolve order = bcast host wins lmhosts
    getwd cache = No
    add user script = /etc/opt/samba/smb_add_user %u
    local master = No
```

Using the familiar diagram, the Terminal Server connectivity looks like:



Prior to Samba version 3.0.2, the Samba code data structure for “netbios aliases =” was 1024 bytes long. Therefore, the total number of aliases that could be defined was limited by the total length of all defined alias names:

(Alias1+Alias2+.....AliasN) <= 1024 (Total Aliases)

Terminal Server itself defaults to “unlimited connections”, or a maximum number of connections may be specified.

Other SMB (non-Samba) applications have been designed for multiple connection servicing on a single process, so the function is neither unprecedented or a showstopper. Because each user environment is so unique, actual production load testing should be done to accurately estimate the number of Terminal Server client connections that can be serviced per smbd. Each Samba environment should also be tested for the functional behavior limit for the “netbios aliases =” parameter.

Note: Samba 3.0.2 and later was enhanced to eliminate the 1024 length issue. HP CIFS Server A.01.10 was based upon Samba 2.2.8a, and is now obsolete.

5.1.2 Terminal Server Hosts File Aliases

The Windows Terminal Server can be configured with a hosts file that is similar in function to the UNIX/Linux /etc/hosts file. The Terminal Server hosts file can be configured to supply Terminal Server aliases for a back-end Samba file/print server. The resulting behavior is the initiation of a discrete TCP/IP connection for each configured alias, which then starts a separate smbd process on the Samba server associated with the transport connect. The default hosts file location is:

C:\WINNT\SYSTEM32\DRIVERS\ETC\HOSTS

The format of hosts file configuration entries is similar to /etc/hosts: an IP address followed by a name. Multiple alias naming strategies are possible. Using the same naming strategy as the Samba "netbios alisases =" from the example above, a sample hosts file would look like:

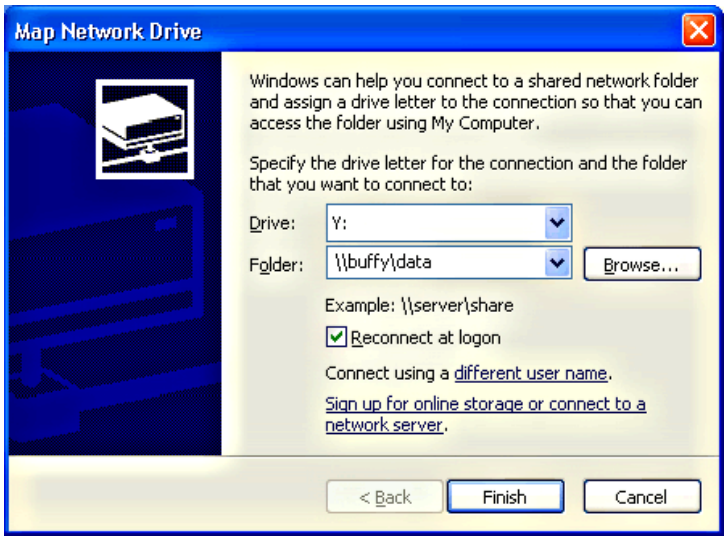
```
127.0.0.1       localhost
192.168.0.1     emonster1
192.168.0.1     emonster2
192.168.0.1     emonster3
192.168.0.1     emonster4
192.168.0.1     emonster5
192.168.0.1     emonster6
```

This strategy would result in the same access behavior as the Samba netbios aliases method: the alias must be manually configured, and the user must know the share name (\\emonster3\share) to connect to.

Another naming strategy is to create an alias with the same name as the Terminal Server user name:

```
127.0.0.1       localhost
192.168.0.1     buffy
192.168.0.1     spike
192.168.0.1     willow
192.168.0.1     oz
192.168.0.1     giles
192.168.0.1     cordelia
```

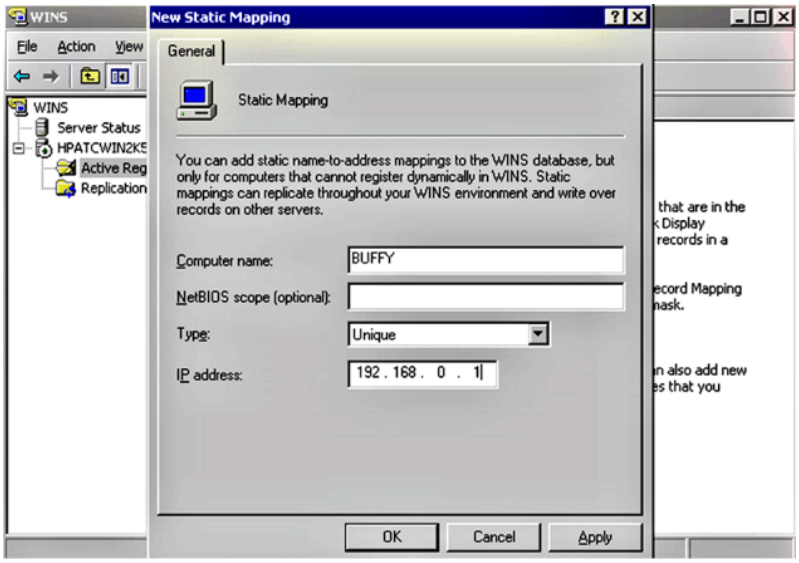
This strategy would result in the Terminal Server user mapping the drive using their own user name (\\buffy\share) instead of the Samba server NetBIOS name (\\emonster\share):



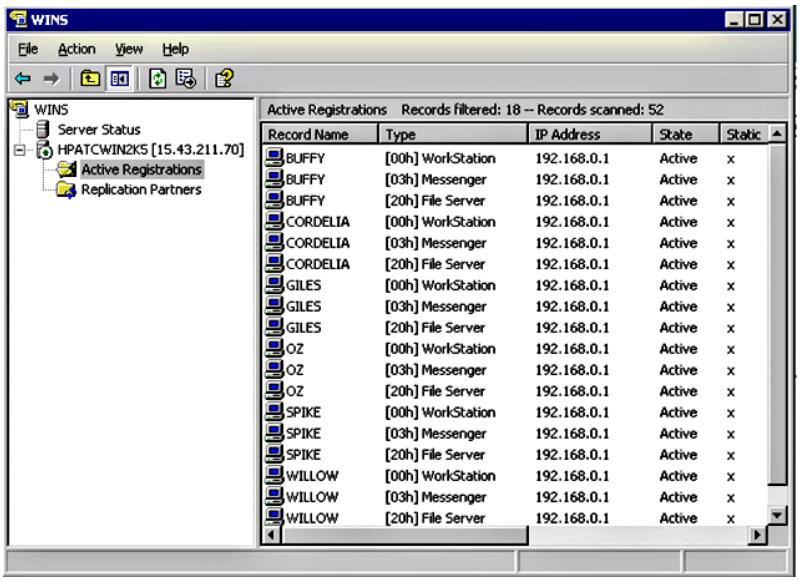
Managing synchronization between the user logon and the Samba share alias could occur via numerous methods in a consolidated fashion on the Terminal Server.

5.1.3 WINS Server NetBIOS Aliases

NetBIOS aliases can also be defined on the WINS server, and they operate similarly to the names defined above in the hosts file. The following graphic shows the static mapping option of the WINS management console from a Windows 2003 Enterprise WINS server. Static mapping allows the administrator to map an IP address to an arbitrary NetBIOS name. Like the hosts file, this allows multiple NetBIOS alias names to be mapped to the Samba server IP address.



Filtering the WINS display for the Samba server emonster IP address shows the static mapping table for the users that looks similar to the hosts file we created above (except with multiple NetBIOS name suffixes per user).



The user can now map their share using the familiar syntax: \\buffy\share. This results in a separate TCP/IP connection per client and a separate smbd process. The same process can be used to create multiple server names per IP address, like emonster1, emonster2, etcetera, as in the hosts file example above.

5.2 Maximum Usernames (security = share)

Prior to Samba version 3.0.2, the Samba code data structure for the username (session_users) was 1024 bytes long. Therefore, the total number of usernames that an individual smbd process could service was limited by the session_users data structure length, but only if "security = share":

(Username1+Username2+.....UsernameN) <= 1024 (session_users)

If the system username policy dictates a name length of 8 characters, then the single smbd can service a Terminal Server session user base of:

Example: 1024(session_users) / 8(username) = 128 sessions

The 129th Terminal-Server-user connection request to the Samba server will be denied by the smbd because it will not be able to allocate space for the additional username. Note that configuring Samba or Terminal Server for aliasing the Samba server will likely workaround this limitation (when each alias has a discrete smbd that is handling either a single username or a subset of the Terminal Server username pool).

Note: Due to the requirement of "security = share" for Samba 2.2 to encounter this issue, the probability of its appearance is slight. Samba 3.0.2 was enhanced to eliminate the 1024 length issue. HP CIFS Server A.01.10 was based upon Samba 2.2.8a, and is now obsolete.

5.3 Home Share Configuration

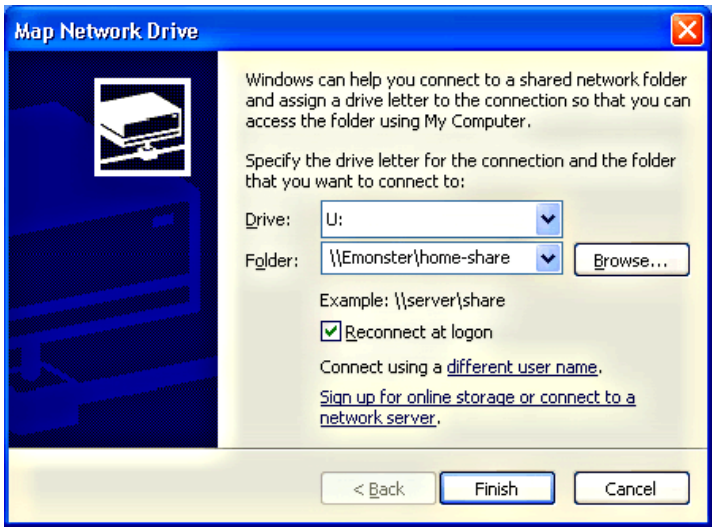
Samba allows for considerable customization of user home share definitions. At least one method of home share configuration is not advisable when servicing multiple Terminal Server users per smbd process.

The most common home share definition is the Samba [homes] share. Using the [homes] share with or without Terminal Server results in accurate and effective handling of user home shares on the Samba server.

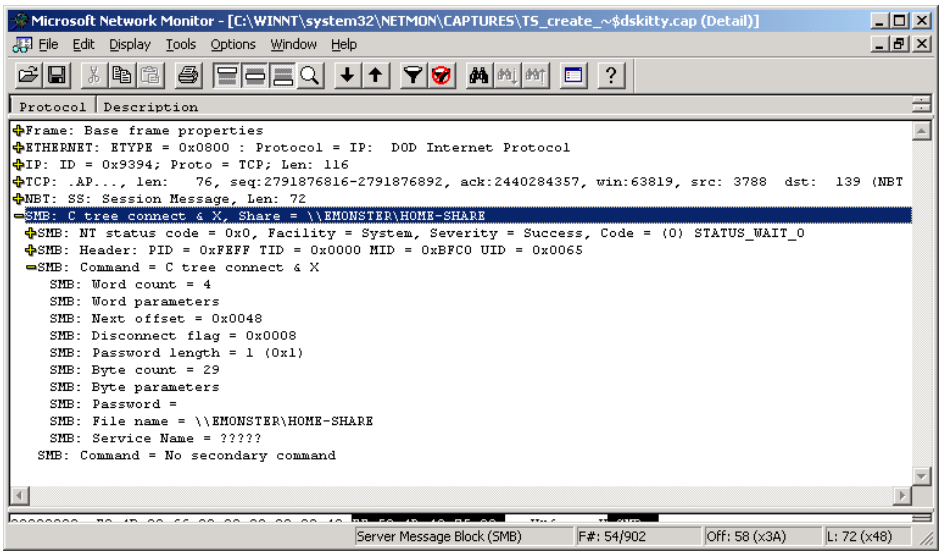
However, using the %U substitution variable for a home share definition could result in unexpected behavior when serving multiple users per smbd through Terminal Server. Observe the following smb.conf share definition:

```
[home-share]
    path = /home/%U
```

The %U substitution variable inserts the "session user name" into the /home/ location. If user buffy mounts home-share, then samba tree connect (tconx) mounts the /home/buffy directory as the service (share). An example from Terminal Server user buffy:



The end result is that the client connects to the /home/buffy Samba service, but that is not how the Terminal Server interprets the service name. By examining the tconx SMB, the Terminal Server service name is observed:

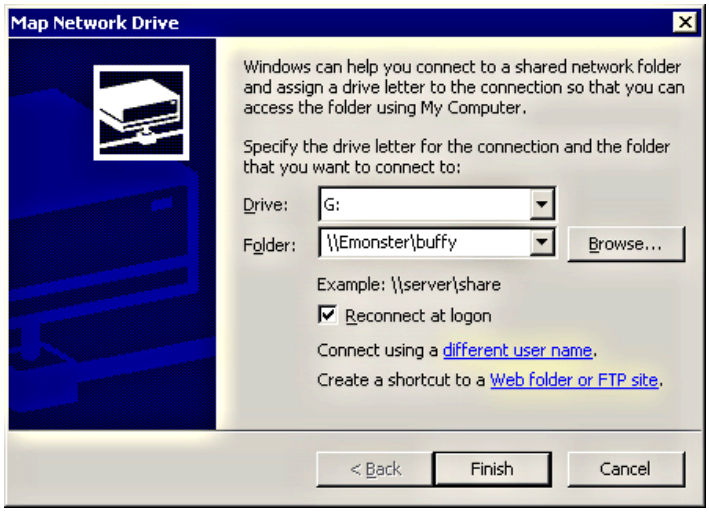


The Terminal Server sees the service name as \\EMONSTER\HOME-SHARE, and not \\emonster\buffy. If the user spike opens a session on the same Terminal Server and mounts the home-share using the same procedure as buffy, Terminal Server will use the same service name as buffy. If both users now access an identical filename on their respective shares using an application that locks the files (like Word or PowerPoint) then the applications via Terminal Server will try to lock the same file, and access will be denied to the second user – even though they are opening different files that exist on their separate home directories. This behavior appears to be application and locking dependant. Note that non-locking applications (like Notepad) do not exhibit this behavior, and correctly read and write to the unique files.

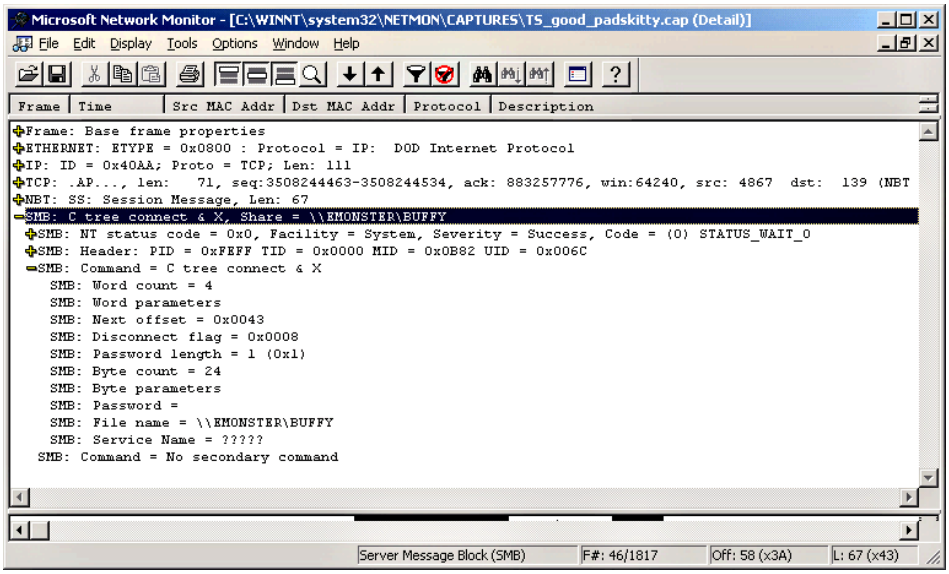
The Samba [homes] feature allows the automatic mapping of users to home shares without the unpredictable behavior described above:

```
[homes]
    comment = user home share
```

The user connects to the home share using the user name, and Samba tconx makes an explicit map to the share:



Using the [homes] share definition, Terminal Server sees the service name as \\EMONSTER\BUFFY. File access and file locking tasks perform correctly.



When configuring Samba for home shares with Terminal Server usage, it is best to avoid defining a share mnemonic with a substitution variable in the path (previous example). The standard Samba [homes] feature is a more reliable option when used with Terminal Server.

5.4 MAX_CONNECTIONS

MAX_CONNECTIONS is a Samba static data structure that defines the number of services that any one smbd process can have open simultaneously. The default value is 128, which means that no client can have more than 128 shares open. In most cases this is a reasonable limit, but for Terminal Server connections it can be exhausted. An actual case example of 32 clients on a single Terminal Server was discovered when each client opened 4 shares at startup (the Terminal Server client session startup). The 33rd user could not connect because MAX_CONNECTIONS had been exhausted for that smbd process:

```
32(clients) * 4(shares) = 128(MAX_CONNECTIONS)
```

Users of open source Samba will have to recompile with MAX_CONNECTIONS (in conn.c) set to a larger number. Hewlett-Packard has contributed code for Samba 2.2.8a on HP-UX that provides a smb.conf configuration variable called “max connections per client =.” This allows the user to specify a customized connection limit without recompiling. The default value for “max connections per client =” is 128. This smb.conf parameter should not be confused with “max connections =”, which limits the number of smbd processes concurrently connected to any one service (share) on the server. Samba 3.0.2 has been enhanced to automatically increase MAX_CONNECTIONS when the default limit is exhausted.

When using the Samba “netbios aliases =” workaround or the Terminal Services hosts file for Samba aliases, the MAX_CONNECTIONS issue does not occur (when each Terminal Server user is allocated a separate smbd process).

Note: Samba 3.0.2 is enhanced to eliminate the maximum (128) issue. HP CIFS Server A.01.10 was based upon Samba 2.2.8a, and is now obsolete.

5.5 Logging

Samba has many smb.conf logfile naming options for the logging feature. A common log file configuration is:

```
log file = /usr/local/samba/log.%m
```

The %m substitution variable supplies the NetBIOS name of the client machine, and the resulting logfile is named “log.machinename”. When a single smbd process is serving multiple Terminal Server users, this configuration will result in all of the Terminal Server user sessions log events being written to a single logfile with the NetBIOS name of the Terminal Server itself substituting for machinename.

A more usable log file naming convention for Terminal Server usage is to use the %U substitution variable:

```
log file = /usr/local/samba/log.%U
```

This will result in individual logfiles for every unique Terminal Server user name.

Chapter 6 Summary

The default behavior of Terminal Server on Windows is to multiplex all user connections to individual machines (Samba file and print servers) over a single TCP/IP connection, which potentially results in multiple Terminal Server user sessions being serviced by one Samba smbd process. The function of the TCP connection establishment behavior is an operating system limitation, and not due to Terminal Server itself. Therefore, other applications (like Citrix Metaframe) will encounter the same behavior when connecting to a Samba server.

At this time, Windows Server NT4, 2000, and 2003 all provide Terminal Server hotfixes that allow Samba client connections to initiate separate TCP connections, which allow Samba to start a discrete smbd process for every Terminal Server client connection.

In cases where the hotfixes may not be deployable, the configuration flexibility and versatility of Samba can compensate for the Terminal Server on Windows default behavior, and thus provide fast and reliable file and print services. Also, many environments run Terminal Server and Samba with no modifications at all, with perfectly acceptable performance. A representative test environment and test suite is recommended for vetting new Terminal Server and Samba configurations using the workarounds, or the default behavior.