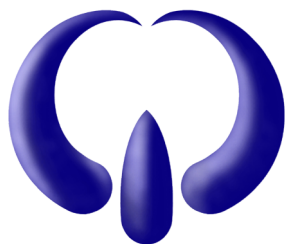




realtimepublishers.comtm

The Definitive Guidetm To

Windows Server 2003 Terminal Services



triCerat
Software®

Gresyon Mitchem

Chapter 3: Load Balancing and Session Directory	46
Terminal Server Hardware Configuration	46
Hard Disk Configuration.....	47
Memory	50
Processor	51
The Bottom Line	53
Fault Tolerance	53
Load Balancing	55
Microsoft Network Load Balancing	55
Configuring NLB	56
Third-Party Load Balancers	62
Session Directory	62
Configuring Session Directory	63
How Session Directory Works.....	65
Summary	67

Copyright Statement

© 2003 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

Realtimepublishers.com and the Realtimepublishers logo are registered in the US Patent & Trademark Office. All other product or service names are the property of their respective owners.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com.

Chapter 3: Load Balancing and Session Directory

If you have more users than a single terminal server can support or you need the ability to take a server offline for maintenance or application installations without impacting availability, load balancing will become an integral part of your Terminal Services architecture. In this chapter, I will introduce you to the native Microsoft Network Load Balancing (NLB) service. I will also cover a new feature of WS2K3—Session Directory—which enables Windows to track user sessions across multiple servers so that users can reconnect to the sessions when needed.

I'll begin by introducing you to the basics of terminal server hardware configuration, then we'll explore terminal server sizing. This foundation of information will enable you to determine how many servers you need to support your users' needs.

Terminal Server Hardware Configuration

When you get right down to it, terminal servers are more like workstations than servers. How many file and print servers have you encountered that have Microsoft Office installed on them? System integrators need to keep this idea in mind when configuring hardware for a terminal server.

Most file, print, and Web servers are configured to optimize disk access to improve performance when users read and write files on the server and to separate the data from the OS to enable easy backup and restore. Terminal servers, however, do not store data; they serve applications. So other than an image capture of the server for disaster recovery and System State backups between application installs, terminal servers do not require regular backups. If there is no data on the server, what are you going to back up?

That being said, terminal servers need to be optimized for the applications they serve. Disk reads need to be optimized, as the executables and DLLs of the applications are accessed by the user sessions. Memory and processors need to be sized to handle the demands of multiple instances of applications across user sessions. It is memory and processor power that are typically the limiting factors of terminal server capacity—not disk space.

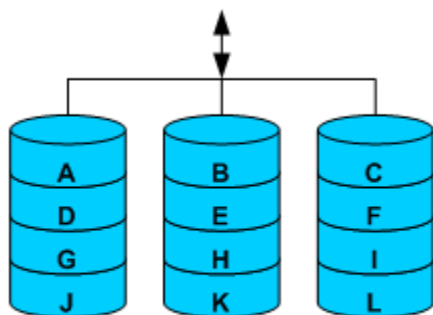
The application packages also come into play—most application installers are written for workstations, so they default to installing to the C drive under the Program Files directory. In fact, even today, there are still applications in popular use that will not even recognize a drive other than C as a valid installation destination. So if your normal server configuration assumes a D drive for data storage, that partition will end up being wasted space on a terminal server.

☞ If you want to separate your applications from your OS by moving the Program Files directory to the D drive, be sure to update the following registry value to reflect the new location—`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\ProgramFilesDir`. You may also need to create custom transforms or installation scripts if you are automating your application installs.

Hard Disk Configuration

As you might have determined by now, it is my recommendation that you configure your terminal servers with a single logical drive. Doing so simplifies application installation, more closely parallels a typical workstation configuration, and eliminates empty drive partitions. But even with a single logical drive configuration you still have options when it comes to the redundant array of independent disks (RAID) level. The RAID level you select is determined by the number of physical disks you have in the server, and the level of disk fault tolerance you require.

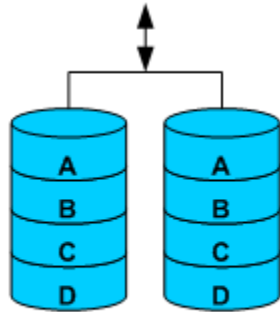
RAID level 0(zero), which Figure 3.1 illustrates, is defined as disk striping without parity. RAID 0 is not fault tolerant—losing a single disk results in all data being lost and the server will go down, so it should never be used in mission-critical environments. The main benefit of RAID 0 is very fast disk reads and writes. Performance can be further increased by placing each drive in the array on a separate controller, thus allowing blocks on each drive to be written simultaneously. Despite its increased performance, this configuration is not recommended for terminal servers as a result of its lack of fault tolerance.



RAID 0 = Fast Disk Read & Write. No fault tolerance.
Minimum 2 drives. Logical space = physical space

Figure 3.1: RAID 0.

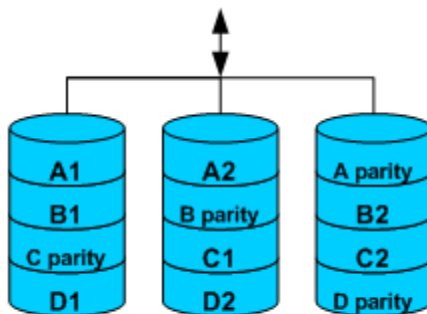
Figure 3.2 shows RAID 1—disk mirroring. RAID 1 is the most common form of RAID, as it is very simple to implement, only requires two disks, and is fault tolerant. RAID 1 doubles the disk-read speed of the server, but does not improve disk-write speed. In RAID 1, all data is written on both drives in the mirror. If one drive goes down, the server can continue to operate using the other drive. RAID 1 is a good option for terminal servers if you are limited on the number of drives available.



RAID 1 = Fast Disk Read. Single drive fault tolerance.
Minimum 2 drives. Logical space = $\frac{1}{2}$ physical space

Figure 3.2: RAID 1.

RAID 5, which Figure 3.3 shows, is also a very common array configuration. RAID 5 is defined as disk striping with parity. As in RAID 0, data is broken up across the disks in the array, but RAID 5 adds parity blocks distributed across the drives. The parity blocks can be used to calculate a missing block if one of the drives fails. Thus, RAID 5 is single-disk fault tolerant. Because data is distributed across multiple drives, read speed is increased dramatically; however write speed is decreased slightly as the controller must calculate the parity block for each unit of data (that is, two reads and two writes are required for each block written). RAID 5 requires a minimum of three disks and is a good option for terminal servers. RAID 5 is also the most cost-effective RAID option offering fault tolerance.



RAID 5 = Very Fast Disk Read. Single drive fault tolerance.
Minimum 3 drives. Logical space = physical space - 1 disk

Figure 3.3: RAID 5.



If you use RAID 5 and you lose a drive, the server will continue to operate for your users, but disk-read speed will be dramatically reduced as the controller must calculate the missing block for every chunk of data.

RAID 10 (sometimes called RAID 1+0) combines the speed of disk striping with the fault tolerance of disk mirroring. This RAID level, which Figure 3.4 shows, is a RAID 0 array, but each element of the array is actually two mirrored drives. RAID 10 provides the fastest disk access of the RAID levels while still maintaining fault tolerance. In fact, under RAID 10, you can lose multiple drives and still be operational (as long as you don't lose both the drives that make up a mirror). The disadvantage of RAID 10 is that you lose half of your disk space to fault tolerance; however, because a lot of drive space is not usually a requirement for a terminal server, so this potential drawback is not an issue (you are not storing data on the server). RAID 10 is an excellent choice for terminal servers that have at least four drives.

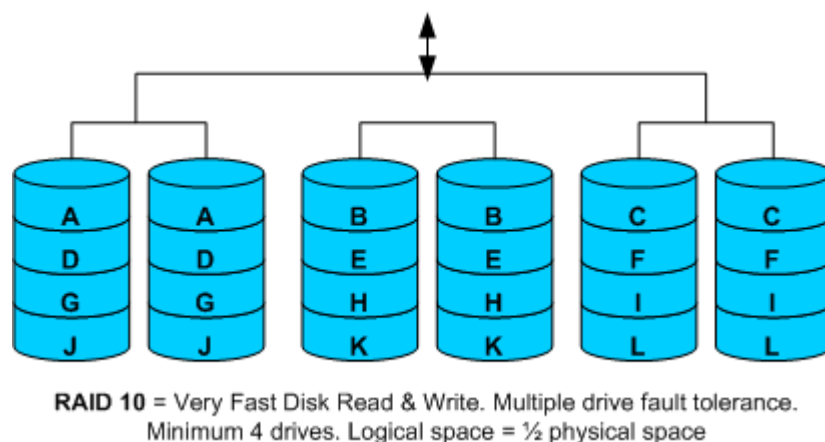


Figure 3.4: RAID 10.

At first glance, RAID 0+1 looks very similar to RAID 10 (see Figure 3.5). The difference is in how the disk controller treats the array. Instead of being a RAID 0 array with each element being mirrored, RAID 0+1 is treated as a RAID 0 array where the entire array is mirrored as a unit. Thus, unlike RAID 10, RAID 0+1 can only lose a single drive and maintain operation. The reason is that once a drive fails, the controller will take the entire RAID 0 array that contains that drive offline and continue to operate as a single RAID 0 array. RAID 0+1 is an excellent option for terminal servers, as it dramatically increases disk operations while maintaining fault tolerance. The determination is usually made by the disk controllers you use in your servers. Most controllers support either RAID 10 or RAID 0+1 but not both.

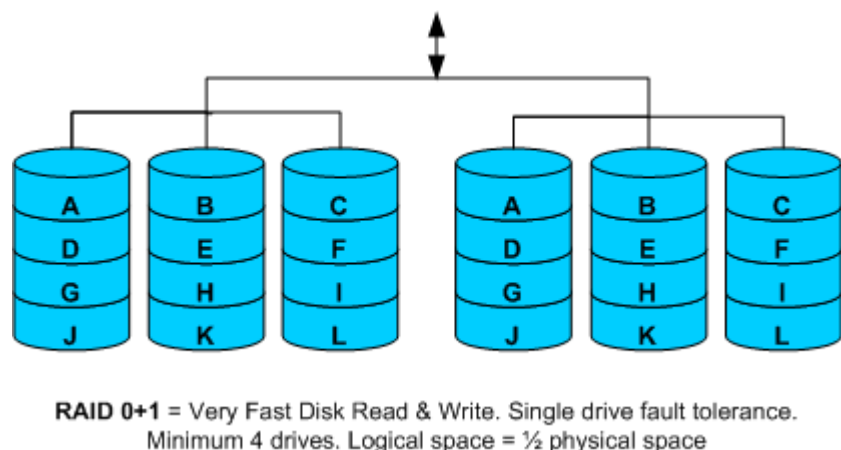


Figure 3.5: RAID 0+1.

Memory

When RAM was more expensive, memory often became the limiting factor in terminal server sizing. Today, it is fairly common to see terminal servers with 4GB of RAM—enough to handle hundreds of heavy user sessions. The amount of RAM in a server is more often limited by the OS and server hardware than the budget.



The first thing to consider when sizing memory for your server is the memory limitations of the motherboard and BIOS. You also want to pay attention to the number of RAM slots available, and purchase RAM in appropriate sizes. The edition of WS2K3 you choose also limits the amount of memory you can use. Table 3.1 shows the maximum RAM and processors for each edition.

Edition	Maximum RAM	Maximum Number of Processors
Standard Edition	4GB	4
Enterprise Edition (32-bit)	32GB	8
Datacenter Edition (32 bit)	64GB	32

Table 3.1: Maximum memory and processors for Server 2003 editions

It is one thing to just throw RAM into a server; it is yet another to accurately judge the amount of memory that your users need. Luckily, there are a number of resources available to assist in estimating the amount of RAM required to support users on a terminal server.

When Win2K was launched, Microsoft, NEC, and Groupe Bull teamed up and produced a white paper to help you determine your servers' memory needs. This document is still quite relevant and helpful for sizing WS2K3 terminal servers. In fact, you can see how the results of this study influenced Hewlett-Packard (HP) in its HP ProLiant Sizer for Citrix MetaFrame XP and Windows Server 2003 Terminal Services—an online tool for sizing HP servers for Terminal Services.

-  You can find the complete text of the white paper "Windows 2000 Terminal Services Capacity and Scaling" at <http://www.microsoft.com/windows2000/techinfo/administration/terminal/tscaling.asp>.
-  The HP ProLiant Sizer for Citrix MetaFrame XP and Windows Server 2003 Terminal Services tool is available <http://activeanswers.compaq.com/ActiveAnswers/Render/1,1027,4825-6-100-225-1,00.htm> (registration required).

Microsoft recommends starting your memory requirement calculations with 128MB of RAM for the OS. After that, you need to determine the type of work your users will perform on the terminal server. The Groupe Bull study divides workers into three categories:

- Structured task workers—Users who tend to use one application at a time in a sequential task-oriented process. These users are typically fast typists (about 60wpm) and are only partially dependant on computer availability. These users typically work in claims processing, account processing/accounts receivable, or customer service.

- Knowledge workers—Users who are dependent on their computers to get work done. These users tend to use multiple applications simultaneously, toggling between them. Their workflow is more ad hoc rather than sequential. These users are the executives, project managers, and analysts.
- Data entry workers—Users who input data into computer systems, such as transcription typists, order entry clerks, clerical workers, and manufacturing personnel.

HP's ProLiant Sizer tool uses the following user types:

- Light Users—Users of single-line-of-business applications. An example of a light user is a call center agent who uses a customer satisfaction database application.
- Medium Users—Users of basic office productivity applications such as word processors and spreadsheets; office administrators are typical medium users.
- Heavy Users—Users who have higher graphics requirements, do extensive Internet browsing, or send and receive a significant amount of email.

Once you determine the type and number of concurrent users your terminal servers must support, you can use Table 3.2 to calculate the amount of memory your terminal servers will need.

User Type	Memory Per User	System Memory	Total Memory
Structured task worker	9.3MB	128MB	System + (number of users × memory per user)
Knowledge worker	8.5MB	128MB	System + (number of users × memory per user)
Data entry worker	3.5MB	128MB	System + (number of users × memory per user)

Table 3.2: Terminal server memory requirements for user types.

The ProLiant Sizer tool goes a step beyond the Groupe Bull study and allows you to factor in additional memory for Advanced Heavy Users and 16-bit applications. 16-bit applications require additional RAM because in addition to the memory requirements of the application, you need to allocate RAM to the Windows 16-bit virtual machine subsystem—WOW.EXE. This addition can add as much as a 25 percent overhead in memory use compared with a 32-bit version of the same application.

Processor

Terminal servers support multiple concurrent users, and multiple concurrent users means multiple concurrent processes and threads. As a result, the demands placed on the processor in a terminal server can be quite high. So having the ability to process multiple threads simultaneously vastly improves the performance of a terminal server. Thus the reason why most terminal servers are multiprocessor servers, allowing the system to run a thread on each of the processors concurrently.

As you saw in Table 3.1, each edition of WS2K3 has a limit on the number of processors it can support. The processor limits of WS2K3 appear to be the same as those of Win2K Server, but WS2K3's native support for HyperThreading introduces a significant difference.

HyperThreading is a new processor technology introduced by Intel in 2002. This technology allows a single processor to handle two threads in parallel, virtually doubling the computing power of the processor.

The difference in the processor limits between the two OS versions comes from WS2K3's ability to distinguish between a physical processor and a virtual one created by HyperThreading. When Windows boots, the OS counts the number of processors in the system. If there are more processors than the OS is licensed to support, the OS will stop counting at the OS limit. It is important to note that the computer's BIOS enumerates physical processors before virtual ones. This behavior prevents Win2K from using virtual processors when there are still available physical ones. Figure 3.6 shows the order in which processors are counted by the OS when using HyperThreading.

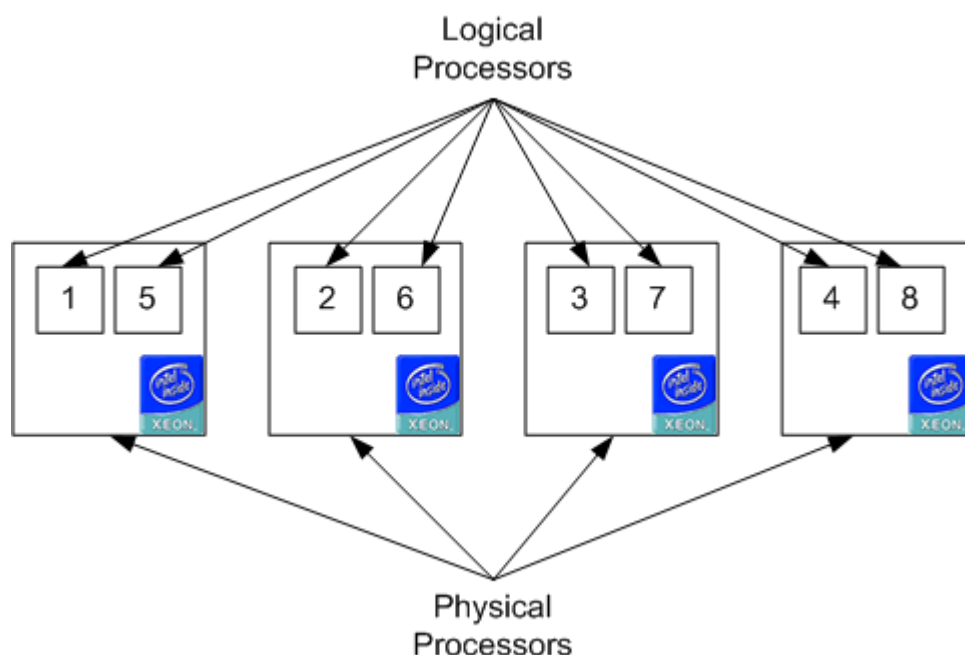


Figure 3.6: The order that processors are counted when using HyperThreading.

Let's compare the processor limits of the Standard Edition of each OS—each is limited to four processors. Because Win2K can not distinguish between physical and logical processors, the OS will stop at the fourth processor. In the example that Figure 3.6 shows, where there are four physical processors with HyperThreading, Win2K will stop at processor four and only place one thread on each processor. If there were only two physical processors with HyperThreading, Win2K would still stop at the fourth processor, but in this case, that would place two threads on each processor by taking advantage of the HyperThreading technology.

WS2K3, however, is capable of distinguishing between physical and logical processors and applies its processor limit only to physical processors. Thus, in the scenario that Figure 3.6 illustrates, WS2K3 would take advantage of all eight logical processors, thus doubling the computing power over Win2K on the same server.


What does this mean for terminal servers? In many cases, WS2K3's support for HyperThreading will enable you to see an increase in performance without having to add processors or upgrade to the Enterprise Edition of Windows. HyperThreading may also influence the class of server hardware you use. You may be able to purchase dual- or quad-processor hardware where you used to select 8-way systems.

The Bottom Line

When you put all this information together, the question that you will be attempting to answer is “How many users will a terminal server support?” Of course, there are a number of variables that go into the answer—the number of applications each user will employ on the server, the memory and processor requirements of each of the applications, and the frequency with which the users toggle between the applications. To clarify, let's explore an example based on current server-class hardware.

Let's start with a dual-processor server that has 2.4GHz Xeon processors with HyperThreading enabled. Next, we'll add 2GB of RAM and four SCSI hard drives in RAID 10 configuration. Such a server would probably cost around \$6000 retail. Based on the Groupe Bull study, this server would support as many as 200 Heavy/Structured task workers. If you are using the same server to publish a single application to your users instead of a full desktop environment, the same server may support as many as 400 concurrent users (depending on the requirements of the application).

If your terminal server desktop environment contains standard productivity applications—Microsoft Office, Outlook, Internet Explorer (IE), and so on—your limiting factor in this example is memory; the dual HyperThreading processors will not be running at capacity. So by simply doubling the RAM to 4GB, you could also double the number of concurrent users.

 There are also third-party utilities, such as Wyse Expedian (<http://www.wyse.com>), that optimize memory use on terminal servers, allowing you to increase the capacity of your servers even further.

Fault Tolerance

Fault tolerance—The ability for a system to continue to function after one of its components has failed.

I've already talked about ensuring hard disk fault tolerance by using RAID, and most server-class hardware provides options for redundant power supplies and network interface cards (NICs). In addition to ensuring fault tolerance in your hardware, you will want to consider the amount of fault tolerance you require at a server level. This decision will be based on how mission critical your terminal servers are:

- **Server fault tolerance**—This level of fault tolerance determines the ability to continue to support your users after losing a single server. To provide server fault tolerance, you calculate the number of servers required to support your users, then add one spare server. You still load balance across all servers, but maintain the capacity needed even if one server is offline.

- **Location fault tolerance**—This level of fault tolerance determines the ability to continue to support your users after losing all servers in a given location. This server loss can be the result of a failure in network connectivity, a building emergency, or a natural disaster. To provide location fault tolerance, you calculate the number of servers required to support your users, distribute them across your locations, then add spare servers equal to the number of servers in a single location. The spare servers are also distributed across locations.

In many cases, you may not need to provide enough fault tolerance to support *all* of your users during an outage—only critical users. To make the right decision about the amount of fault tolerance you need requires adequate data gathering; talking to management and your users to determine the number of mission-critical users and to assess the business impact of an outage.

Figure 3.7 illustrates an example comparison of server and location fault tolerance. In the example, there are 1200 mission-critical users. For server fault tolerance, this number of users would require four servers, each able to handle 400 concurrent users. During normal conditions, each server would only have 300 users, but in the event of a server outage, the remaining three servers could still handle the load. For location fault tolerance, with two buildings used to house servers, you would need 6×400 user servers. This way, if Building 1 were unreachable, the three servers in Building 2 could continue to support your users.

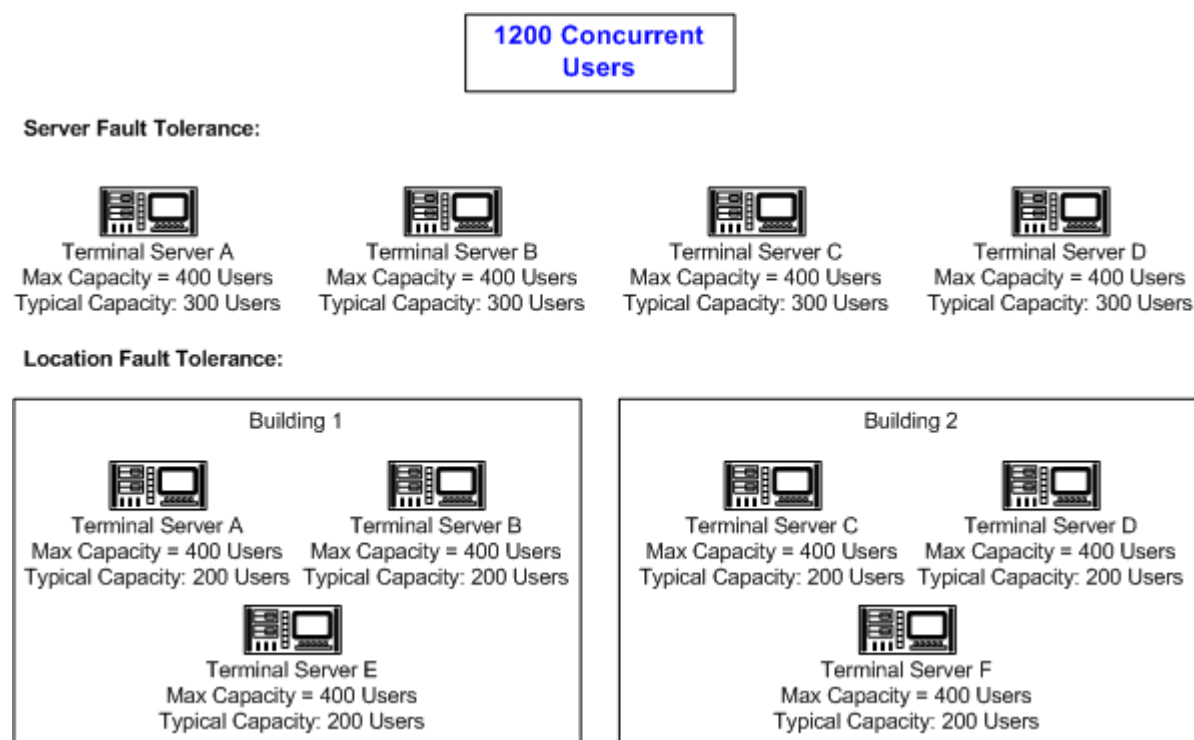


Figure 3.7: A comparison of server and location fault tolerance.

You should also consider usage schedules in your design. If your users work 24 hours a day 7 days a week, you will need to provide adequate capacity to allow you to take a server or servers offline for maintenance and new software installs. However, if your users work a 9-to-5 schedule, you will be able to perform maintenance in the evenings, thus saving you the cost of additional servers.

Once you determine the number of servers required to support your users and provide adequate fault tolerance, you then need a way to distribute user sessions across the servers. Enter load balancing.

Load Balancing

The most basic form of load balancing is done manually. You create separate connection files for each of your servers and distribute the files to your users, telling percentages of users to use a specific connection file as their primary. Going back to the example that Figure 3.7 shows, if you have 1200 users and four servers, you would create four connection files, give all four files to the users, but instruct 400 users to connect to server A, 400 to connect to server B, and so on. If one server goes down, users can then use another connection file as a backup. If you want your load balancing and failover to be automatic and not rely on users to remember a different procedure for failure scenarios, you will need to implement a load balancer of some kind.


Microsoft Network Load Balancing

NLB enables you to place a group of servers behind a virtual IP (VIP) address. NLB distributes connections made to the VIP among the servers in the cluster and maintains intra-server communication so that if a server becomes unavailable, NLB will stop directing clients to it.

In WS2K3, Microsoft has continued to evolve its native load balancer. Under Win2K, NLB was only available in Advanced and Datacenter editions. WS2K3 includes NLB in all four editions. In addition, Microsoft has included the following enhancements:

- **NLB Manager tool**—Win2K requires that you configure each server in the NLB cluster manually, which makes setting up a cluster very tedious and increases the opportunity for error. WS2K3 includes a new administrative tool that enables you to centrally create, configure, and control NLB clusters.
- **Virtual clusters**—When using NLB for clustering Web servers, you can create multiple clusters on the same set of servers by utilizing multiple IP addresses on each server. Each virtual cluster is represented by its own VIP.
- **Multi-NIC support**—WS2K3 allows you to bind NLB to every NIC in the system. Win2K only allowed NLB to be bound to one NIC on the server.
- **Multicast support**—WS2K3 NLB clusters can be configured to use multicast for intra-server communication.

Even with these enhancements, NLB still has limitations that you must consider in your design. First, all members in an NLB cluster must be in the same subnet. If you need to distribute your terminal servers across more than one subnet (for location fault tolerance perhaps), you will have to create a separate cluster for each subnet and design a way to load balance across the clusters. Second, NLB only considers the number of active connections to the server when determining which member server to direct a connection to; it does not have any way to measure the amount of memory or processor in use. Third, when used with a Session Directory, NLB requires that the IP addresses of the terminal servers must be visible to clients; it does not support routing tokens. Also, NLB is limited to 32 nodes in a cluster.

 Within the NLB Manager tool, Microsoft refers to a group of load-balanced servers as a cluster. Do not confuse this terminology with true server clustering, which allows servers to share processes and storage to effectively act as a single server. Most documentation on load balancing, including many of Microsoft's own white papers, refers to load-balanced groups of servers as *farms*. In this book, however, I will use the term *cluster* to be consistent with the NLB Manager tool.

Configuring NLB

Once your terminal servers are configured and your applications installed, you are ready to begin configuring them for NLB. Before you begin, you will want to gather the following information:

- The VIP address you have selected to represent the cluster
- The unique IP addresses of each server in the farm
- The Domain Name System (DNS) alias you want to use for the cluster
- The network protocol and port on which you want to load balance (for RDP, this port is TCP port 3389).
- A domain account with administrative rights to all servers in the cluster, or the local administrator account and password for each server

After gathering this information, you should launch the NLB Manager tool. You can do so from any WS2K3 system or from a Windows XP client with the WS2K3 Administrative Tools installed.

The NLB Manager tool, which Figure 3.8 shows, lets you create, configure, and manage NLB clusters on the network. In this window, you will want to create a new cluster by selecting New from the Cluster menu.

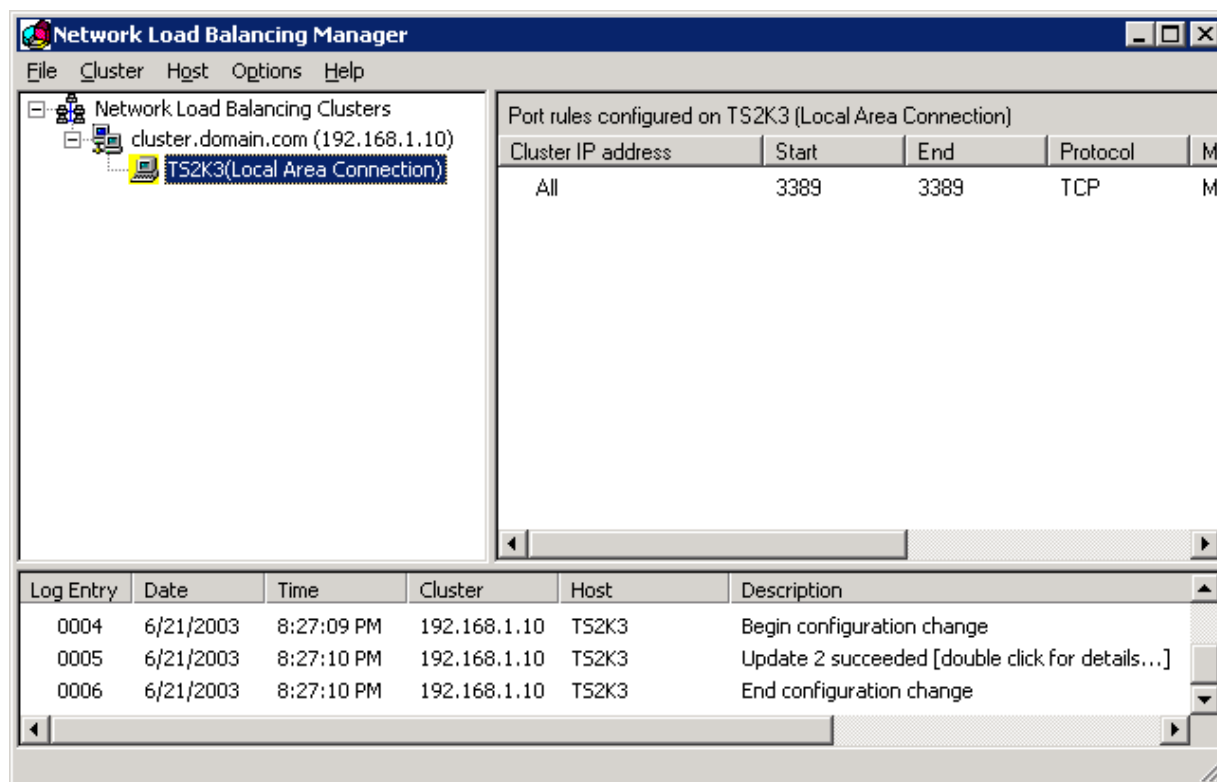
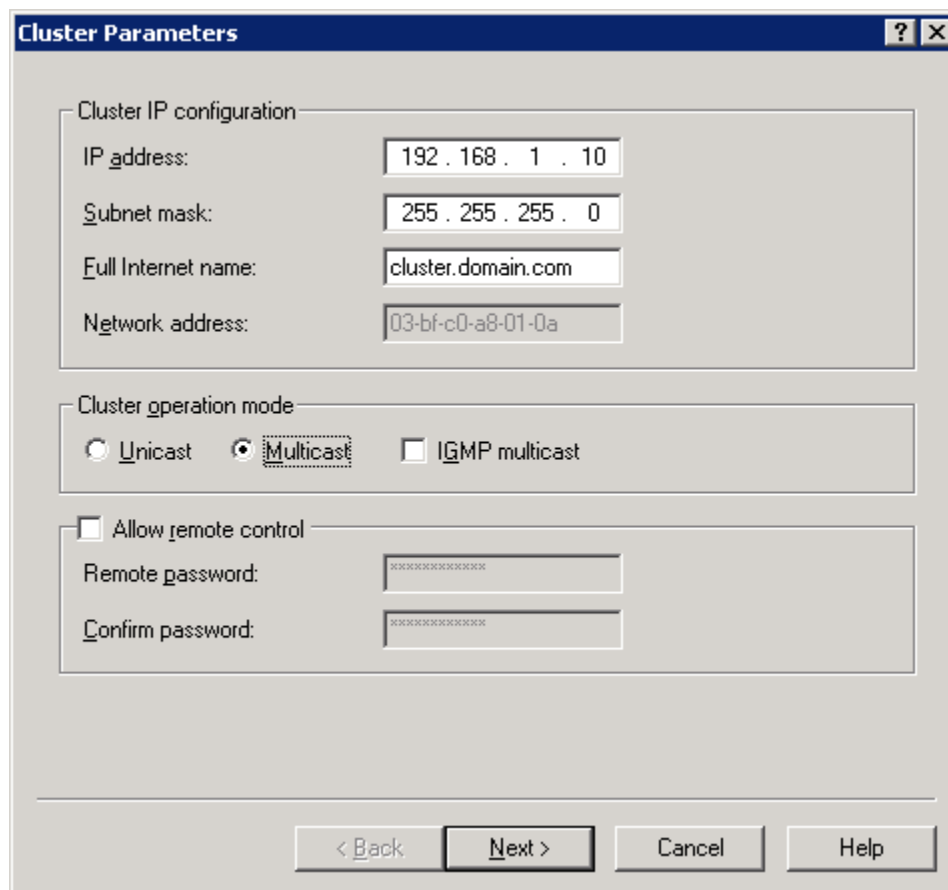


Figure 3.8: The NLB Manager tool.

You will then be presented with the Cluster Parameters dialog box, which Figure 3.9 shows. In this window, you will enter the VIP address for the new cluster, the subnet mask, and the DNS name of the cluster. If you plan to do all management of the cluster through the NLB Manager tool, you do not need to enable remote control or set a password. NLB Manager uses the Windows Management Interface—WMI—so it uses the user's credentials to make changes to the servers in the cluster.



The image shows a Windows-style dialog box titled "Cluster Parameters". It contains three main sections:

- Cluster IP configuration:** This section has four text input fields:
 - IP address: 192 . 168 . 1 . 10
 - Subnet mask: 255 . 255 . 255 . 0
 - Full Internet name: cluster.domain.com
 - Network address: 03-bf-c0-a8-01-0a
- Cluster operation mode:** This section contains three radio buttons:
 - Unicast (unselected)
 - Multicast (selected)
 - IGMP multicast (unselected)
- Allow remote control:** This section has a checkbox that is currently unchecked. Below it are two password input fields:
 - Remote password: (masked with asterisks)
 - Confirm password: (masked with asterisks)

At the bottom of the dialog box are four buttons: "< Back", "Next >", "Cancel", and "Help".

Figure 3.9: Configuring cluster parameters.

You can also select whether intra-server communication is done using unicast or multicast. If each of your terminal servers has only one NIC, and your network supports multicast within the subnet, you should enable multicast. Doing so allows you to manage the cluster through NLB Manager running on any of the servers in the cluster. After filling out your cluster's parameters, click Next.

 For additional information about enabling multicast support, see the NLB Clusters Help topic or the "Technical Overview of Windows Server 2003 Clustering Services" white paper found at <http://www.microsoft.com/windowsserver2003/techinfo/overview/clustering.mspx>.

The next window of the NLB Manager tool asks you for any additional IP addresses used in the cluster. This information is typically used for load balancing Web servers, where each server hosts multiple Web sites, each represented by a unique IP address. For terminal servers, you can leave this page blank, and click Next to bring up the Port Rules dialog box.

Port rules are used to determine how connections to the VIP are handled. The default rule states that all TCP and UDP traffic is distributed equally among the cluster nodes. For terminal servers, you only need to load balance RDP, so highlight the default rule, and click Remove. Then click Add to bring up the Add/Edit Port Rule window (see Figure 3.10).

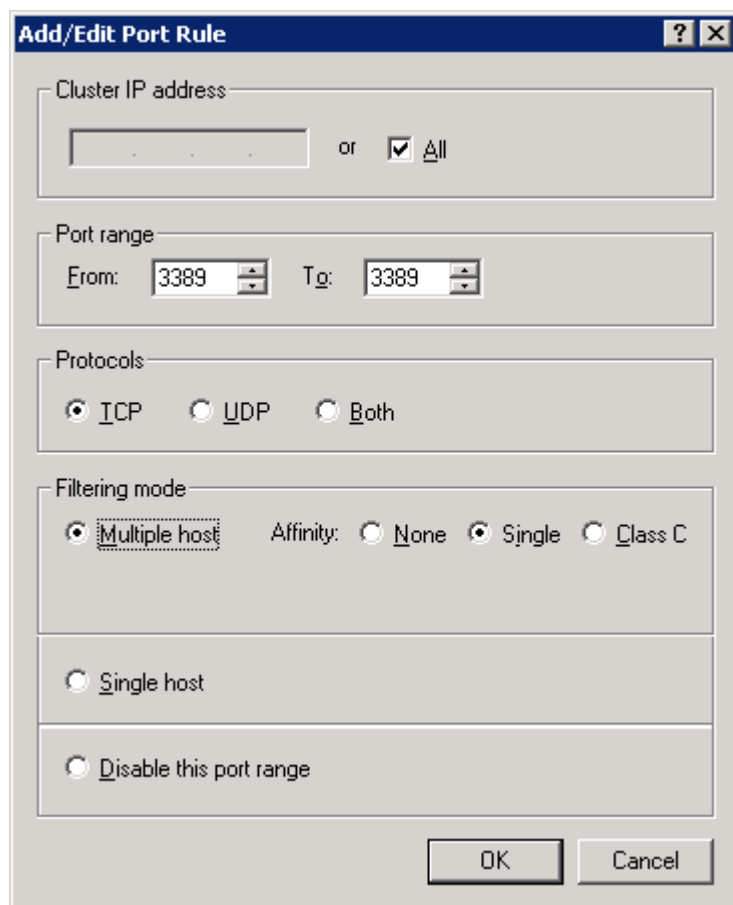


Figure 3.10: Editing a port rule in NLB Manager.

In this window, you will set the port range as From 3389 To 3389, and specify TCP as the protocol. This configuration will only load balance connections made by RDP. By limiting your port rule to RDP, you reduce overhead on your NLB service and on your node servers by shielding them from erroneous connections made to the VIP on other ports.

You should leave the filtering mode set to the default (multiple host, single affinity), but the following list provides information about the options available:

- **Multiple host**—With this setting selected, connections made to the specified port range will be load balanced across multiple nodes. When selecting multiple host filtering, you must also select an affinity mode. Affinity ensures that once a client is directed to a specific node, the client continues to be directed to that same node for all communication during the session. The affinity options are:
 - **None**—No affinity is used.
 - **Single**—Affinity is based on the IP address of the client; load balancing is performed on a per-client basis.
 - **Class C**—Affinity is based on the Class C subnet of the clients; once a single client establishes a connection to a specific node, all connections from that subnet will be directed to the same node.

- Single host—Connections made to the specified port range will be directed to a single, specific node, unless that node is unavailable, in which case connections are made to the next node calculated by Handling Priority number. The Handling Priority for the node number is set when you add a server to the cluster. This option is used to create server fault tolerance without load balancing.

You can also disable the rule in this window. Doing so allows you to temporarily disable load balancing of a specific port range without having to delete and re-create the rule.

After configuring your port rules, click Next to proceed to the Connect window (see Figure 3.11). In this dialog box, you enter the name of the first server to be added to the cluster. The NLB Manager tool will establish a connection to this server and display a list of the configured network adapters. You should select the adapter from which you want to accept incoming connections from the VIP, and click Next.

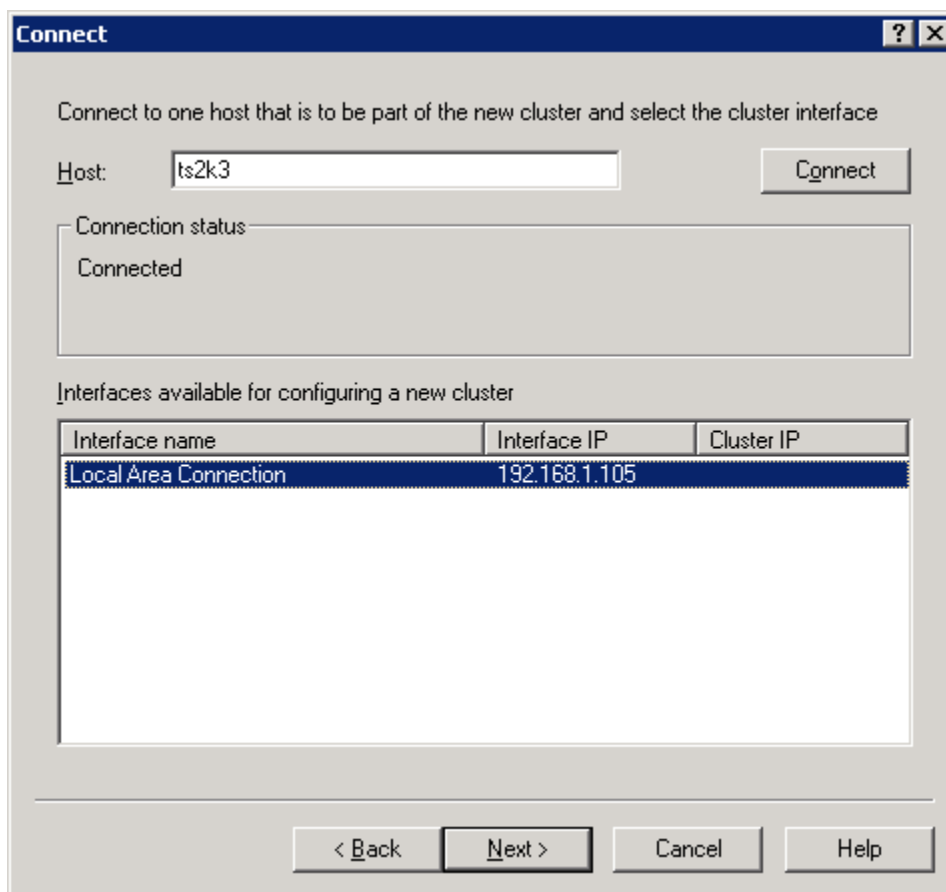


Figure 3.11: Adding the first server to a new NLB cluster.

In the resulting Host Parameters window (see Figure 3.12), you configure the specific server you have selected to join the cluster. In this window, you first set the priority number, which is used both as a unique identifier for the node and for failover order in a single host filtered cluster. You can change the physical IP address of the server and its subnet mask if needed, and set the default state for the load-balancing service when the server boots.

The dialog box titled "Host Parameters" contains the following fields and controls:


- Interface:** A text box containing "Local Area Connection".
- Priority (unique host identifier):** A dropdown menu with the value "1" selected.
- Dedicated IP configuration:** A section containing two text boxes:
 - IP address:** "192 . 168 . 1 . 105"
 - Subnet mask:** "255 . 255 . 255 . 0"
- Initial host state:** A section containing:
 - Default state:** A dropdown menu with "Started" selected.
 - Retain suspended state after computer restarts:** An unchecked checkbox.

At the bottom of the dialog are four buttons: "< Back", "Finish", "Cancel", and "Help".

Figure 3.12: Configuring host parameters for an NLB cluster node.

After clicking Finish, the NLB Manager tool will connect to the server using WMI and configure the NLB service. You will be returned to the main NLB Manager window where you will see your newly created cluster. In this window, you can right-click the cluster to add hosts. Each time you add a server, you will be required to select a network interface and set a priority number for the node.

When you are done adding the remaining terminal servers to the cluster, you can test NLB by using the Remote Desktop Connection client to connect to either the VIP address or the DNS name of the cluster. Your connection should be routed to a specific server in the cluster, and you will be prompted with the logon screen for that server.

 To connect to the DNS name of the cluster, you must either have Dynamic DNS (DDNS) running in your network, which allows the NLB service to register the cluster name, or manually add the cluster alias to your DNS database.

Third-Party Load Balancers

Although Microsoft NLB is robust, easy to configure and manage, and is free with all editions of WS2K3, there are many reasons to consider a third-party load balancer for your terminal server environment. Perhaps you need to distribute your servers across multiple subnets. Or you need more than 32 servers to support the number of users in your environment. Maybe you are using Session Directory and want to enable connection tokens so that you can hide the physical IP addresses of your terminal servers from your users. (We'll explore Session Directory in more detail in a moment.) Or your company may simply have a standard product and infrastructure for load balancing. Regardless of the reason, there are many excellent load balancers on the market.

One major factor that may place one load balancing product ahead of the rest is how it determines load. With many load balancers—Microsoft NLB included—load is simply the number of active connections to a node. In a terminal server environment, this method of measure is not always adequate. Perhaps you have a mixture of Heavy, Medium, and Light users in your environment. If load is calculated purely by the number of open connections, you have no way of preventing one server from handling more Heavy users than the rest—it is purely the luck of the draw.

Some load-balancing products are able to place metrics on the node servers and determine load based on available memory, processor utilization, or other performance factors. These types of products are a great advantage when load balancing terminal servers as they keep the actual performance of your servers at an equal level, even if that means that one server is handling more users than another.

 In addition to load balancers, there are also third-party products specifically designed to enhance Terminal Services. These products not only handle load balancing, but also allow you to publish applications, making it easier for your users to connect to a specific application rather than a terminal server desktop. The two leading products are Citrix MetaFrame (<http://www.citrix.com>) and New Moon Canaveral iQ (<http://www.newmoon.com>).

Session Directory

One of the challenges of creating a load-balanced cluster of native Windows terminal servers is how to deal with disconnected sessions. As we explored in Chapter 2, administrators have the ability to configure timeouts for idle and disconnected sessions on a terminal server. You can either set these timeouts very low, giving only enough time for a user to reconnect from the same client device and IP address in the event of a network failure, or you can set it fairly high, giving your users the ability to disconnect from a session, leave applications running, then reconnect at a later time to pick up where they left off.

These settings work perfectly well in a single server environment. When the user reconnects to the server, Session Manager reconnects them to his or her existing session. If, however, you have a load-balanced cluster of terminal servers, Session Manager is unaware of sessions on the other servers in the cluster. Microsoft addresses this challenge in WS2K3 with Session Directory.

The Session Directory server maintains a dynamic database that maps user names to open sessions on all terminal servers in the cluster. This feature enables users to be reconnected to existing sessions on any server in the cluster, regardless of which server the NLB service initially directs/connects them to.

Configuring Session Directory

To take advantage of the Session Directory feature, all terminal servers in the cluster must be running WS2K3 Enterprise or Datacenter edition. The Session Directory server can use any edition of WS2K3. You can even create the Session Directory on one of the terminal servers in the cluster, although it is not recommended because that would prevent you from taking that terminal server offline for software installations/upgrades without impacting the entire cluster.

To configure Session Directory, begin with the server you want to host the session database. On this server, open either the Computer Management or Services administrative tool to access the Terminal Services Session Directory (among the available services) Go into this service's properties, set the startup type to automatic, and start the service (see Figure 3.13).

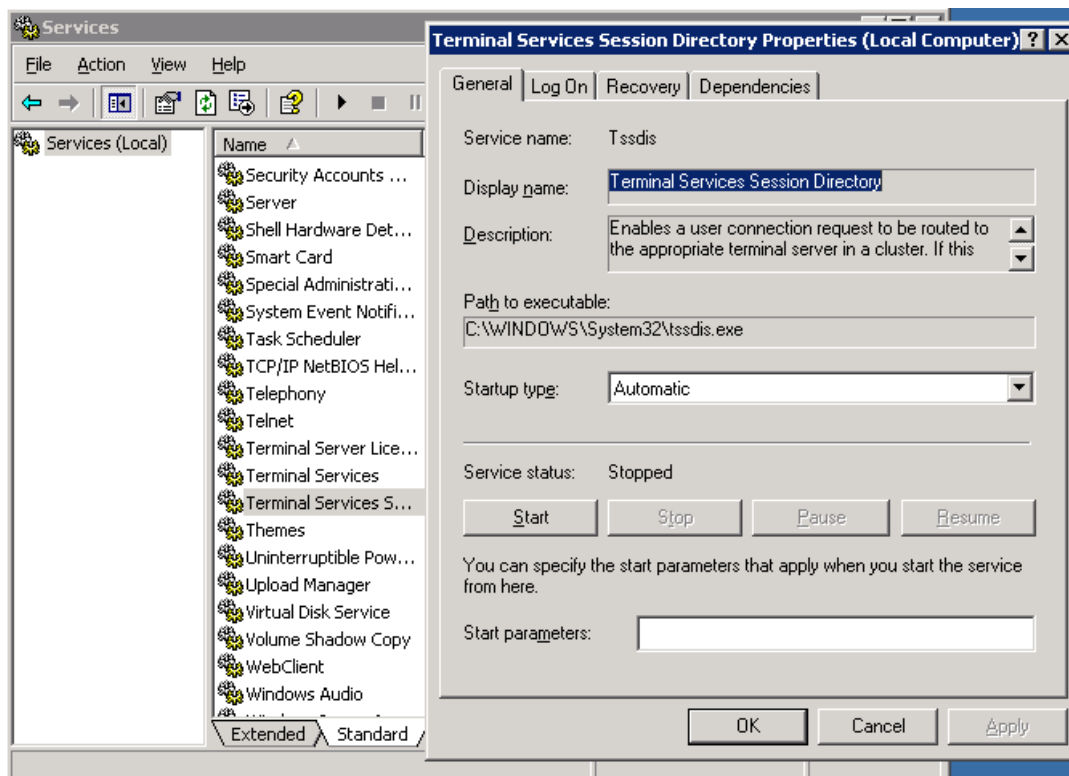
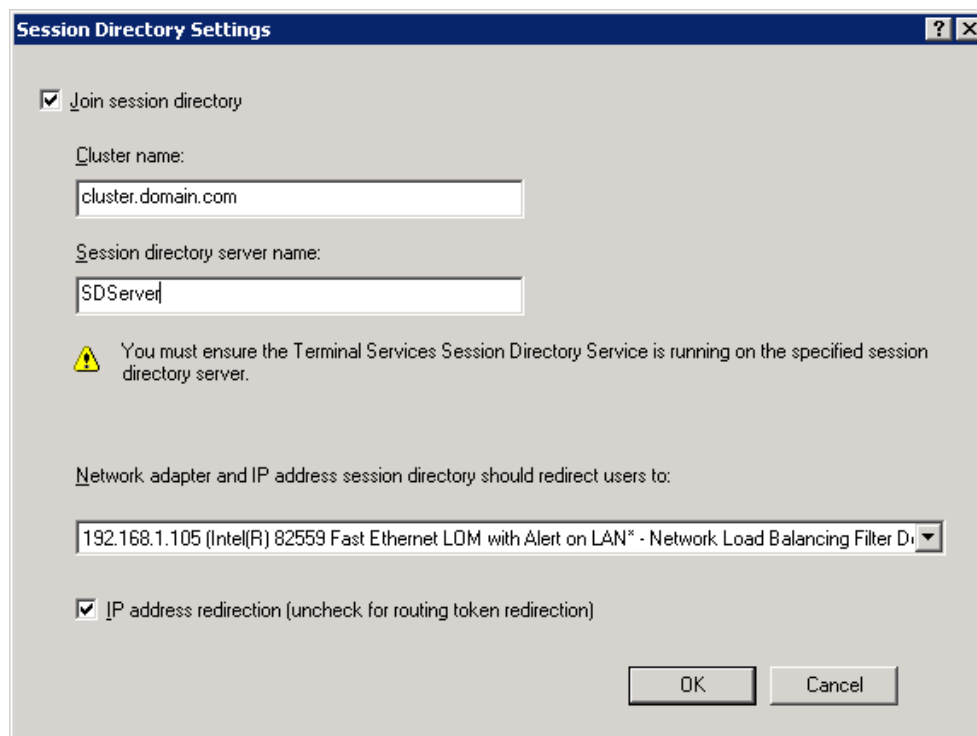


Figure 3.13: Enabling the Terminal Services Session Directory service.

The first time the Session Directory service is started, it will create a new local group on the server called Session Directory Computers. For a terminal server to inform the Session Directory server of the terminal server's sessions or query the Session Directory for sessions on other servers, the terminal server must be a member of this group. You can either add the individual computers in the cluster to the group or create a domain group containing the terminal servers and add that group to Session Directory Computers.

Once your Session Directory server is set up, you will then need to configure the terminal servers to take advantage of it. You can do so with either the Terminal Services Configuration tool or the Group Policy Object Editor. On WS2K3 Enterprise or Datacenter editions, the Server Settings node of the Terminal Services Configuration tool has an additional option—Session Directory. Figure 3.14 shows the properties page of this setting.




Session Directory Settings

☒ Join session directory

Cluster name:

Session directory server name:

 You must ensure the Terminal Services Session Directory Service is running on the specified session directory server.


Network adapter and IP address session directory should redirect users to:

☒ IP address redirection (uncheck for routing token redirection)

OK Cancel

Figure 3.14: Session Directory settings for a terminal server.

Here you enter the cluster name of the NLB cluster to which the terminal server belongs and the host name or IP address of the Session Directory server. If your terminal server has more than one network interface, you must select which one to redirect client connections to. You must also specify whether to instruct clients to reconnect to the unique IP address of this server by checking IP address redirection or to use routing token redirection if the server IP addresses are not visible to the clients.

 If you are using Microsoft NLB, you must select IP address redirection, as NLB does not support routing tokens.

If you choose to use the Terminal Services Configuration tool, you will have to configure the same settings on every terminal server in the cluster individually. If you are in an Active Directory (AD) environment, using Group Policy to configure your servers is a much simpler option as it reduces the chance for error and gives you a single location to make changes if needed.

To use Group Policy to configure your terminal servers for Session Directory, create or edit a GPO that applies to all terminal servers in the cluster. Then, in the Group Policy Object Editor, navigate to Computer Configuration, Administrative Templates, Windows Components, Terminal Services, Session Directory, as seen in Figure 3.15.

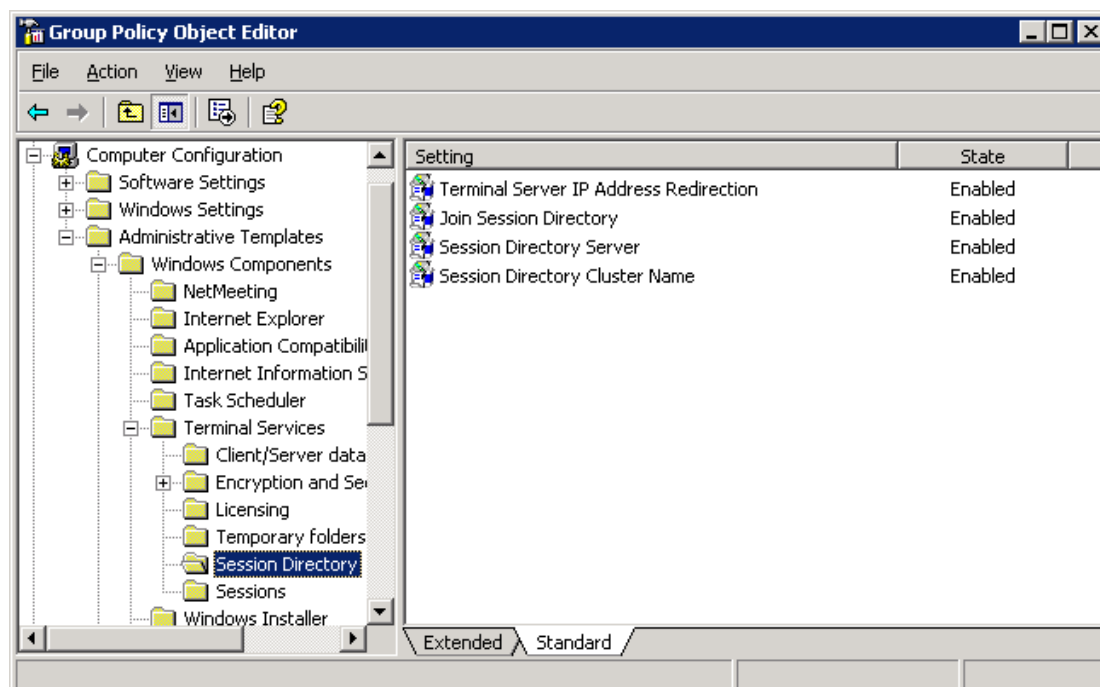


Figure 3.15: Session Directory Group Policy settings.

From this point, you can access settings that match those in the Terminal Services Configuration tool. Enable all four settings and enter the required information. At the next policy refresh interval, all servers in the cluster will be configured to take advantage of Session Directory. By using Group Policy to configure Session Directory, you make the task of adding servers to the cluster much easier because you will not need to configure each server individually; they will simply inherit the settings from the GPO.

☞ When using Session Directory, the server hosting the database becomes critical to the operation of your terminal servers. You might want to take advantage of Windows clustering to make your Session Directory fault tolerant. Microsoft includes instructions about how to do so in the white paper “Session Directory and Load Balancing Using Terminal Server” at <http://www.microsoft.com/windowsserver2003/techinfo/overview/sessiondirectory.mspx>.

How Session Directory Works

The following section provides an example scenario to give you an idea of how Session Directory works. In the morning, Joe User uses the Remote Desktop Connection client to open a connection to TSCluster.Domain.Com—the DNS alias of the NLB cluster that Figure 3.16 shows. The NLB service directs the connection to TServer02, the least loaded server at that time. Joe logs into TServer02, and because Joe does not have an existing session on TServer02, the server queries the Session Directory service for any existing sessions belonging to Joe User on any other servers in the cluster. Joe does not have any other sessions, so TServer02 accepts the connection, and Joe gets to work.

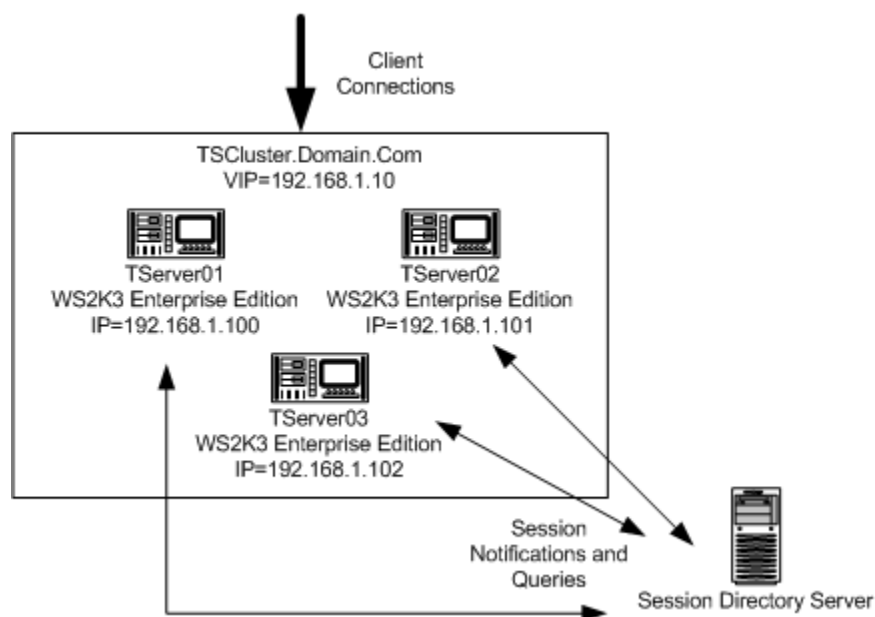


Figure 3.16: An example Session Directory architecture.

Around 3:00, Joe decides that he is going to head home and continue working from there. He is in the middle of a document, and has three Web browsers open for reference material, so he disconnects from the session rather than logging off. This way, he can pick up right where he left off when he gets home. When Joe disconnects from his session, TServer02 informs the Session Directory server of the disconnected session. The Session Directory server creates an entry in the database that lists Joe's username and domain, the server hosting the session, the time the session was created and disconnected, and the resolution and color depth of the session.

At home, Joe establishes a virtual private network (VPN) connection to the corporate network. He launches the Remote Desktop Connection client on his home computer, and enters TSCluster.Domain.Com. The NLB service directs the connection to the unique IP address of the server node with the least number of active connections—this time it is server TServer01. Joe is then presented with the logon screen for TServer01 and enters his credentials.

TServer01 first checks its own list of disconnected sessions to see whether Joe has a session to reconnect with. When it finds none, TServer01 then queries the Session Directory database. The database finds a session listed for Joe on server TServer02, so it sends Joe's Remote Desktop Connection client the IP address of TServer02 and Joe's encrypted credentials.

The Remote Desktop Connection client then automatically connects to TServer02, and sends the credentials. TServer02 finds that Joe has an existing session and reconnects Joe, who happily continues to work on his document.

Summary

When designing a terminal server infrastructure for a large number of users, there are several factors to be considered. In this chapter, I covered terminal server sizing, which will enable you to determine how many concurrent users a given server will handle. I then talked about the different levels of fault tolerance, which may impact the number of servers you use.

Once you determine how many servers are required, you then need a way to distribute client connections across the servers. This chapter introduced you to the Microsoft NLB service—the native option for distributing connections across a cluster of servers.

Finally, I covered WS2K3's new Session Directory service, which enables you to ensure that users get reconnected to disconnected sessions regardless of which server in the cluster they connected to initially. By using these techniques and technologies together, you can create a large, fault-tolerant set of terminal servers ready to accept connections from thousands of clients.

In Chapter 4, I will cover the techniques and tools used to administer and manage your terminal servers and the user sessions on them. I will cover common administrative tasks, from configuring user profiles to assisting users via remote control. I will also cover some advanced techniques for managing servers within a load-balanced cluster, and provide some helpful scripts for automating repetitive tasks.