realtimepublishers.com™

The Definitive Guide™ To

Windows 2000 Terminal Services

NEW MOON SYSTEMS™

Greyson Mitchem

Chapter 5: Performance, Capacity Planning, and Availability 97
Capacity Planning 97
The Testing Environment 97
The Tools 98
RoboClient (ROBOCLI.EXE) 98
RoboServer (ROBOSRV.EXE) 99
QueryIdle (QIDLE.EXE) 100
SimulatedClient (SMCLIENT.EXE) 100
Test Scripts 101
The Test Domain 101
Performance Monitoring and Troubleshooting 101
Hardware Monitors 101
Network Traffic 102
Scheduled Reboots 102
Hardware Fault Tolerance 102
System Monitors 102
Performance Monitor (PerfMon) 102
Logging and Alerts 105
Task Manager 106
Load Balancing 107
Installing Load Balancing 108
Affinity Mode 112
Controlling Load Balancing 112
Application-Level Load Balancing 113
Terminal Server Maintenance 113
Summary 115

Chapter 5: Performance, Capacity Planning, and Availability

With all the advantages that the server-based computing model provides, there are also a number of challenges. By centralizing your users' computing environment, you create a much greater need for fault tolerance, performance, and availability. In the traditional computing model, if a workstation goes down, it typically will affect only a single user, but in a terminal services environment, if a system becomes unavailable, it will impact a large number of clients.

As you saw in Chapter 4, careful use of permissions, user rights, and group policies will reduce the possibility that a user will be able to negatively impact the server or other users. But in addition to protecting the system, you need to implement a strategy to address capacity needs, fault tolerance, and system maintenance.

In this chapter, I will go into depth about the capacity planning tools that I introduced you to in Chapter 2. I will also introduce you to the Microsoft load-balancing service, which you can use to implement fault tolerance at a per-server level. Once you have the tools needed to design a fault-tolerant architecture that will handle your users' needs, we will dive into techniques for monitoring system health and maintaining top performance through preventative maintenance.

Capacity Planning

As server and memory prices continue to drop, capacity planning is not the daunting task that it once was. In Chapter 2, we went through the NEC and Groupe Bull study and established some baseline memory requirements for your terminal servers. In most cases, generous application of these guidelines will prove sufficient, and when you implement load balancing as part of your infrastructure, you can easily add a server to the cluster if your server hardware reaches maximum capacity.

I will introduce you to the tools that Microsoft, NEC, and Groupe Bull used in their study, but I should strongly caution you that these tools were developed exclusively for the study and were not originally intended for public release. Therefore, they are poorly documented and difficult to use. In most cases, I would recommend implementing a test server and using live QA testers to establish the load of an unusual application.

> To obtain the capacity-planning tools, download them from ftp://ftp.microsoft.com/reskit/win2000/roboclient.zip. The tools are also included in the Win2K resource kit, but the versions in the kit simply do not work. If you have the resource kit and want to patch the capacity-planning tools, you can download a hotfix at http://download.microsoft.com/download/win2000platform/Tscpt/1.0/NT5/EN-US/tscpt_hotfix.exe.

The Testing Environment

To use the capacity-planning tools, you will need to set up a testing environment similar to the one that Figure 5.1 illustrates. The lab environment will need to include the following computers:

- A terminal server with applications you want to test as well as the QueryIdle (QUIDLE.EXE) program.
- A test manager system to run RoboServer.
- A domain controller to authenticate your test's virtual user accounts.

- A number of test clients. These clients must be workstations and not thin client devices. Depending on the workstation hardware, you can run a number of test connections from each machine. Each client machine will need SimulatedClient and RoboClient installed in addition to the Terminal Services client program.

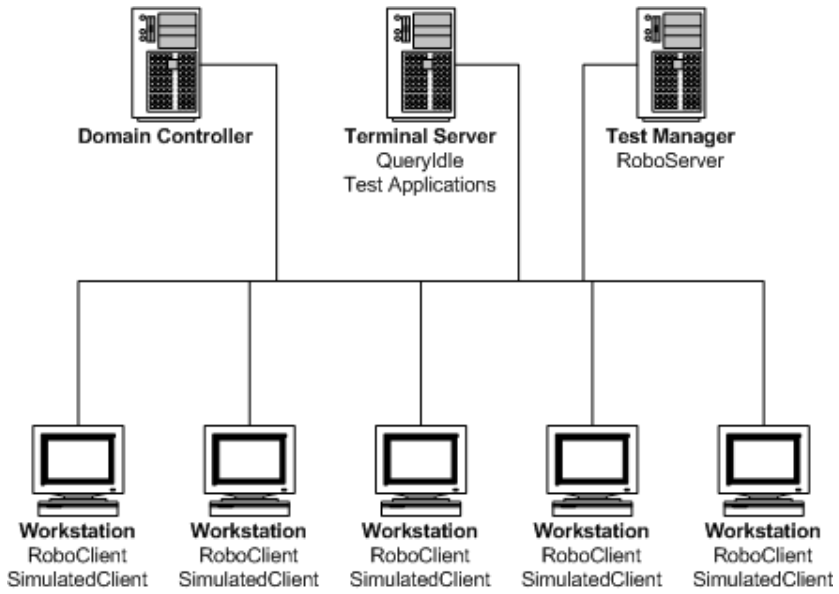

Figure 5.1: Capacity planning lab computers.

The Tools

As Figure 5.1 shows, there are four components to the capacity-planning toolkit. These components include RoboClient, RoboServer, QueryIdle, and SimulatedClient.

RoboClient (ROBOCLI.EXE)

The RoboClient program is installed on each test workstation. It communicates with the test manager system and controls execution of your test scripts. By default, when you launch ROBOCLI.EXE, it looks for a test manager system named ts-dev. If your test manager system has a different name, you can use an –s switch to specify another server name:

```
robocli.exe [-s servername]
```

(where *servername* is the name of the test manager system), or enter the name in the GUI, as Figure 5.2 shows.

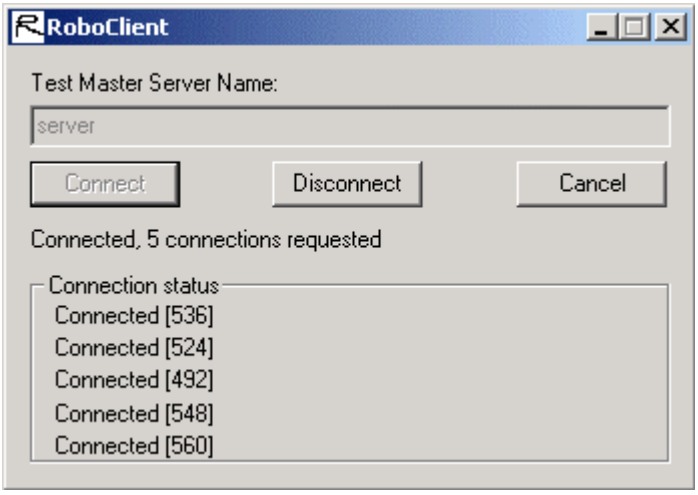

Figure 5.2: The RoboClient GUI.

RoboServer (ROBOSRV.EXE)

RoboServer is installed on the test manager system. This tool is the conductor of all activity during the test. RoboServer instructs each instance of RoboClient when to open a new connection to the terminal server and which test script to run in the session. From the GUI, you can specify the following parameters:

- The name of the terminal server to use
- The number of sessions each client computer should establish
- How many sessions make up a test set
- The delay time between test sets
- The delay between multiselect commands (if you instruct a number of sessions to begin a test script, RoboServer will wait this number of seconds between execution)

You can also specify the terminal server name and the number of sessions per client from the command line using an –s and –n switch, respectively:

```
robosrv.exe [-s TS_Name] [-n sessions_per_client]
```

Figure 5.3 shows the RoboServer GUI.

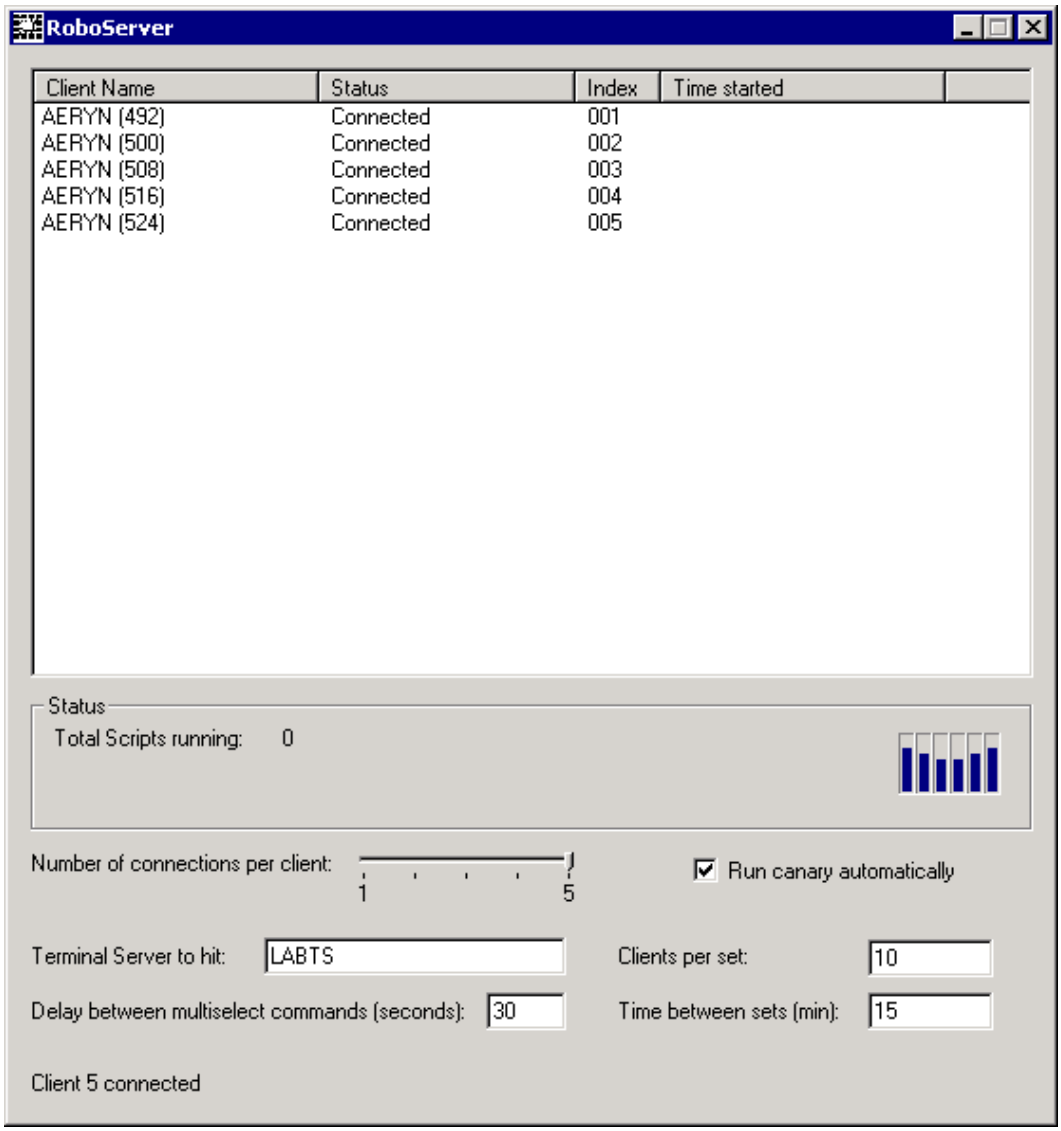

Figure 5.3: The RoboServer GUI.

QueryIdle (QIDLE.EXE)

The QueryIdle program is typically run from the console of the terminal server. This program monitors sessions and updates its display every 30 seconds. If a session becomes idle for 2 minutes, QueryIdle beeps and displays the username and computer name of the idle session so that you can reset the session or troubleshoot your test script. You launch QueryIdle from the command line; include a server name as an argument if you want to monitor a server other than the one on which you launch QueryIdle, as the following example shows:

```
QIDLE.EXE TS_Name
```

SimulatedClient (SMCLIENT.EXE)

The SimulatedClient program runs on each client system in the test. This program waits for RoboClient to kick it off, then opens a session on the terminal server and begins running a

specified script. During the test, SimulatedClient is completely controlled by RoboClient and RoboServer. However, if you are writing your own test scripts, you can use the following command to manually launch SimulatedClient to test your script before beginning an automated test:

```
smclient.exe –f file_name [-s TS_Name]
```

Test Scripts

Once each instance of RoboClient connects to RoboServer, you can right-click the connection and instruct that instance to execute a test script. Doing so passes the correct command-line parameters to SMCLIENT on that system and begins the test.

> You can download template testing scripts from http://www.microsoft.com/windows2000/zipdocs/TSScale.exe.

To write your own test scripts, you will need to learn the language that SMCLIENT uses—a fairly simple language that requires every action to be specified, right down to key ups and key downs. I will go through the command syntax of the scripting language in the appendix.

> The script names are hard-coded into RoboServer and match the template scripts that you downloaded. If you are writing your own scripts, you will need to use the same filenames that Microsoft used.

The Test Domain

You will need to set up user accounts on the test domain controller for your sessions to use. Once again, the tools expect a default set of usernames, so you should build accounts named smc001, smc002, smc003, and so on. For simplicity, you might want to use blank passwords for all accounts.

As you can see, the capacity-planning tools are fairly high-maintenance to use. Most terminal server administrators will find that setting up this type of test is far more work that it is worth. Instead they use the published figures as a guideline and carefully monitor the performance of the systems in the real world making capacity projections from there and implementing additional systems when necessary.

Performance Monitoring and Troubleshooting

As with any server implementation, monitoring the health and performance of your systems is critical. In most cases, your existing monitoring systems should be able to address the needs of you terminal server hardware, but there are some specific metrics that you might want to watch within the systems, especially if you are evaluating the capacity of the systems to determine when to implement additional processors, RAM, or parallel servers.

Hardware Monitors

There are many excellent server-monitoring systems. Depending on your server hardware and existing infrastructure, you should be able to incorporate your terminal servers into the existing systems.

> Compaq Insight Manager, Hewlett-Packard OpenView, and NetIQ AppManager are all common system-monitoring systems and can all handle the unique requirements of terminal servers.

Rather than go into the specifics of hardware monitoring, as they are no different than with any high-availability Wintel servers, I will point out how terminal server monitoring is unique.

Network Traffic

If you have a large number of clients connecting to a terminal server, you may see elevated network traffic. You might want to implement either a teamed-NIC solution or a high-speed LAN connection (100Mb or gigabit Ethernet) to improve network performance.

Scheduled Reboots

As I will explain later, many terminal server implementations use regularly scheduled reboots to help maintain system performance. Be sure that your system monitor takes this behavior into account by either having "ignore periods" or adequate delays before issuing an alert.

Hardware Fault Tolerance

Most servers today have fault-tolerant components—multiple power supplies, dual NICs, and RAID disks. If any of these components fail during normal operation, the backup systems will take over and allow the server to continue to run. Typically failure will trigger an alert with the monitoring system. In a terminal server environment, backup systems are critical but pose an unusual challenge—many systems will stop the boot sequence if one of the redundant systems is down and wait for an administrator to OK the boot on backup systems. This behavior can be a problem with scheduled reboots. If you can, change the default setting in the server's BIOS to continue the boot process without intervention.

System Monitors

In Win2K, Microsoft provides a number of utilities for monitoring and troubleshooting system performance. You are probably already familiar with most of these tools, but when you enable Terminal Services, a number of new options become available. You should always run performance monitors during your testing and staging phases to establish baselines for comparison once you migrate your Terminal Services-based applications to production systems.

> In addition to the native Win2K tools, a number of third-party systems are available to help monitor the health of your systems and alert you in the event of a problem. NetIQ and Quest Systems both make excellent products.

Performance Monitor (PerfMon)

PerfMon is the tried and true native monitoring tool for Windows systems. As Figure 5.4 shows, you can use PerfMon to view, log, and create alerts based on performance counters. In addition to the counters that you may be familiar with—Free Memory, % Processor Time, and so on—Terminal Services also has a number of counters specific to it. I will define some of the key

counters to look at, both standard and Terminal Services-specific, then go over how to set up logging and alerts.

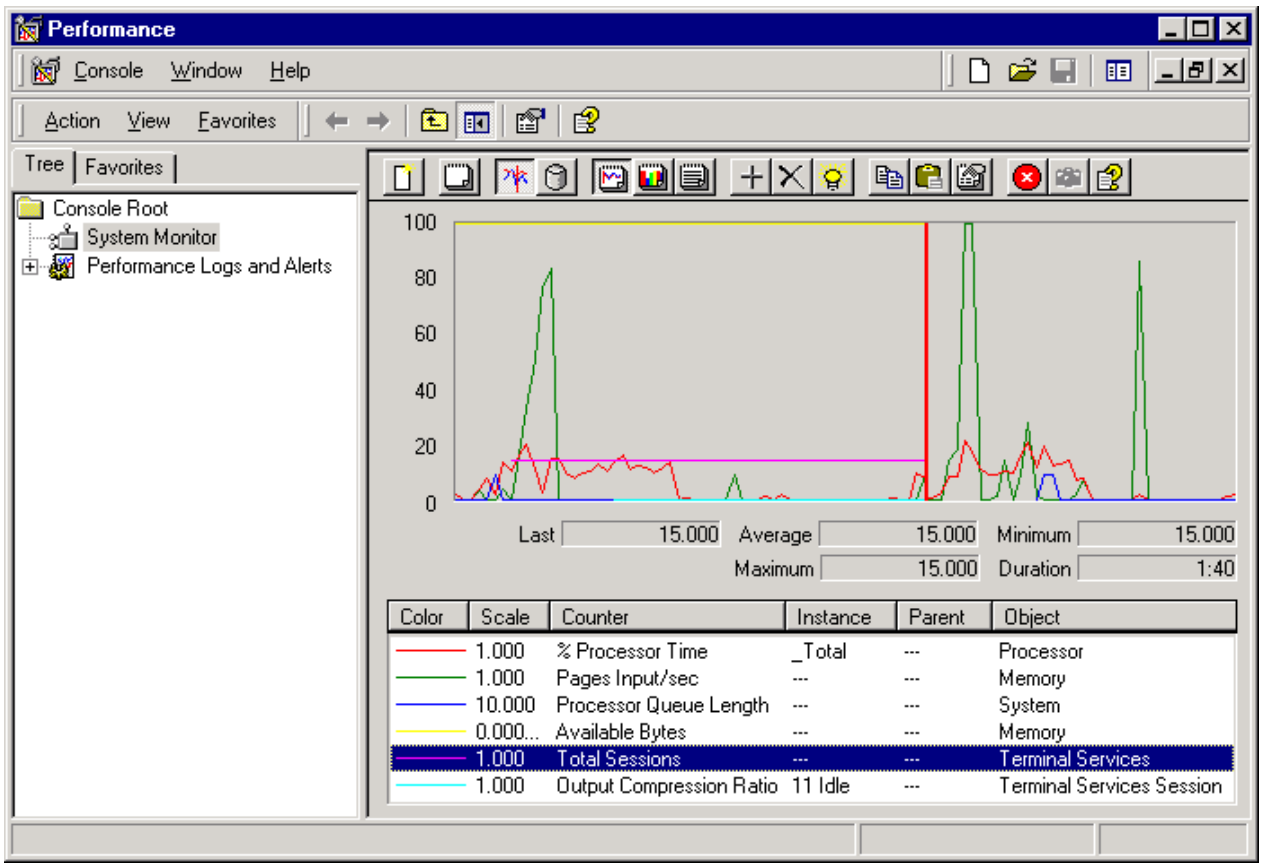

Figure 5.4: Example Performance Monitor counters.

To add counters to the PerfMon window, right-click the graph or click the plus (+) icon on the toolbar. This will bring up the Add Counters dialog box that Figure 5.5 shows. Here you specify the counter you want to add to the graph and the server you want to monitor. Clicking Explain is very helpful, as it adds a lower pane to the dialog box that gives a definition of the counter that you have highlighted.

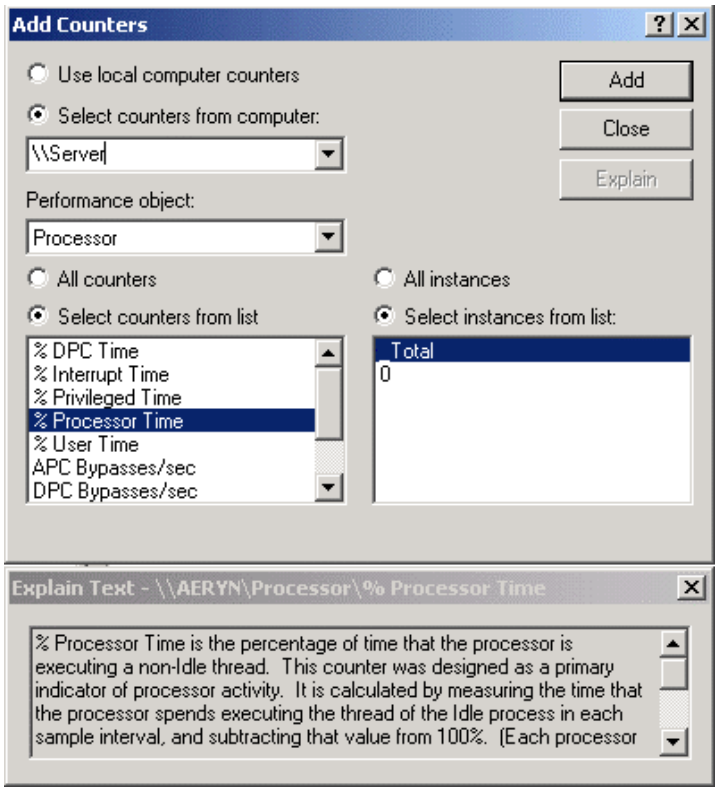

Figure 5.5: The Add Counters dialog box.

Here are some of the key counters to look at:

- % Processor Time—Displays the percentage of time that a processor is busy with a non-idle thread. In a multi-processor system, you can monitor an individual processor or the total. This counter is found in the Processor Object.
- Processor Cue Length—This counter is found in the System Performance Object and displays the number of threads waiting for an available processor. If you see a cue of greater than 2 over an extended period of time, you may have a congested system. In this case, you need to add processors or parallel servers to lighten the load.
- Available Memory (in bytes, kilobytes, or megabytes)—These three counters under the Memory Object display available physical memory on the system. Monitoring this level will help you determine whether additional RAM is needed.
- Page Inputs/Sec—This counter is key to measuring system performance. It measures the number of times per second that the system must go to the page file instead of physical memory. Also found under the Memory Object, this counter should always be looked at in conjunction with the available memory on the system.
- Active Sessions—When monitoring a terminal server, two new objects are available. Under the Terminal Server Object, you can monitor Active, Idle, and Total sessions. By using these counters as a baseline when looking at the other performance counters, you can establish a healthy total number of users to allow on each server.

- Terminal Services Session Object—This object isn't an individual counter but an object with a number of counters that can help you monitor performance of the session manager and RDP. This object includes counters for compression ratio, RDP input and output, and the efficiency of the client-side bitmap cache.

Logging and Alerts

Through PerfMon, you can set up log files to track system performance over time and alerts to inform you if the system becomes over taxed. To create a log file, in the left pane of PerfMon, expand the Performance Logs and Alerts node, then right-click Counter Logs. In the resulting dialog box, you create a log in the same way that you added counters to the live graph. Figure 5.6 shows the logging interface.

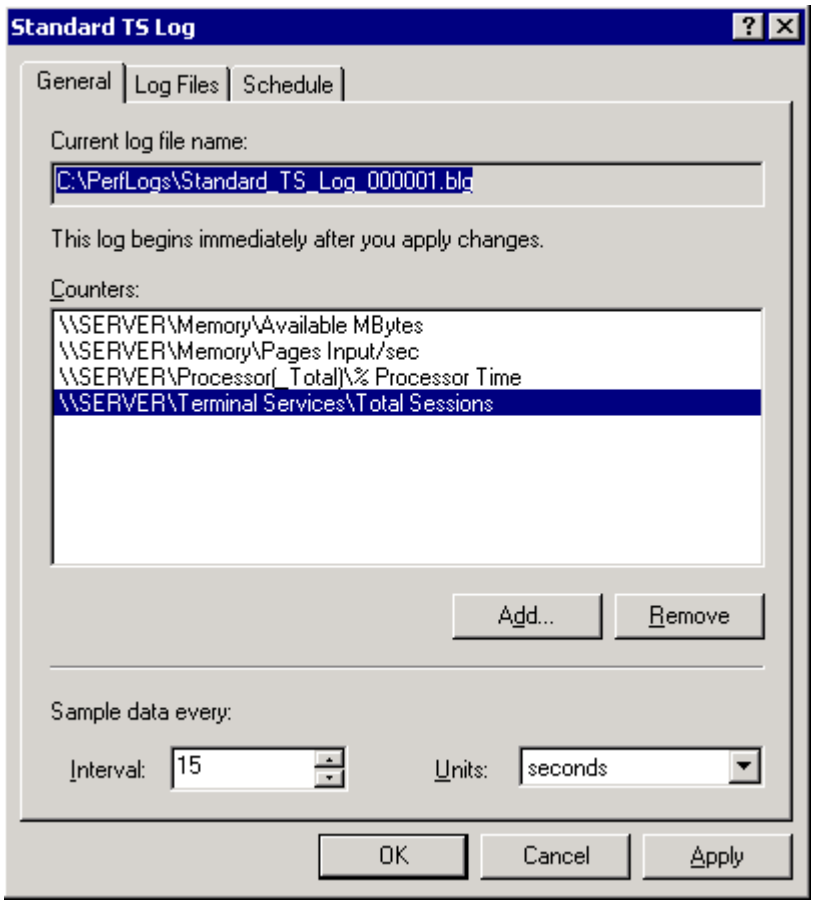

Figure 5.6: Adding a log to Performance Monitor.

The other two tabs in this window allow you to change the default filename and location for the log and control start and stop times for the log. On the Log Files tab, you can also select the format for the log file.

Also in the left-pane of the main PerfMon window, you can add administrative alerts. These alerts are used to trigger specific actions when a defined value for a counter is reached. Figure 5.7 shows the Add Alert dialog box.

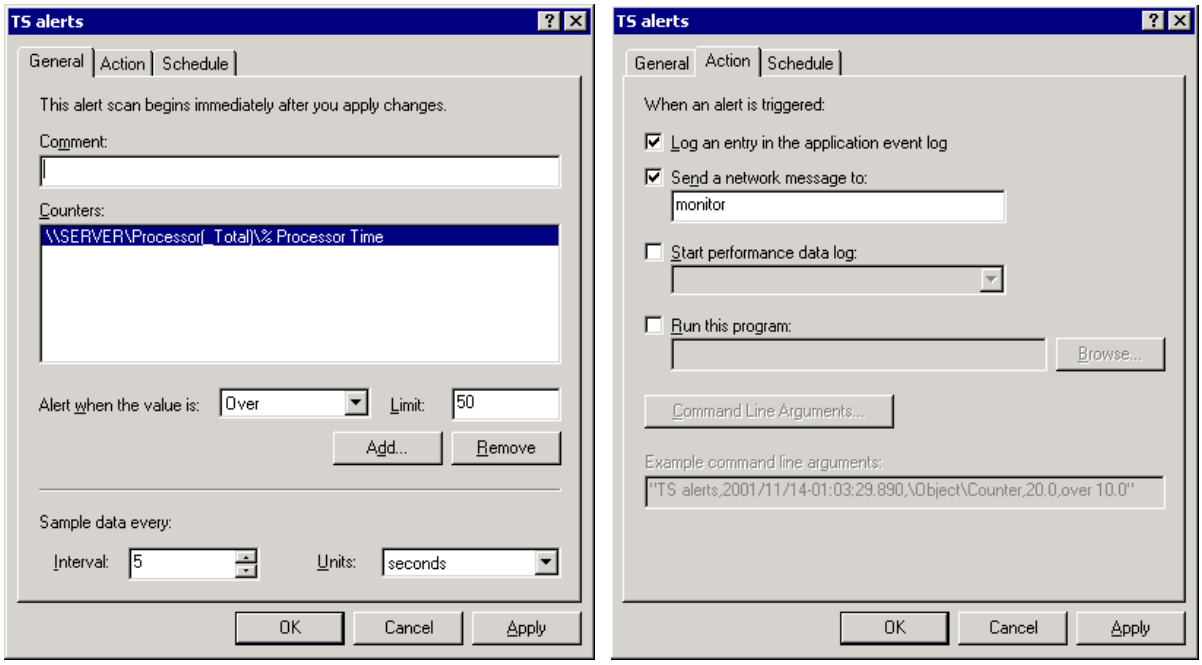

Figure 5.7: Setting up an alert in Performance Monitor.

In Figure 5.7, an alert is triggered when % Processor Time reaches 50 percent; an entry is added to the application event log, and a network message is generated to the computer named Monitor. You can also use events to start a specific PerfMon log or trigger a specific program to run on the server.

Task Manager

Task Manager is one of the most common tools used in Windows and is often considered a user tool; it offers a great deal of information in a very condensed format. On a terminal server, you can use Task Manager to display processes in either only your session or all sessions on the server. From here you can kill a hung process or quickly spot a leaky one.

In Figure 5.8, you can see the *Show processes from all sessions* check box. Once this check box is selected, you should select Select Columns from the View menu, and enable the Username and/or SessionName column. This information will easily identify which users are running a specific process.

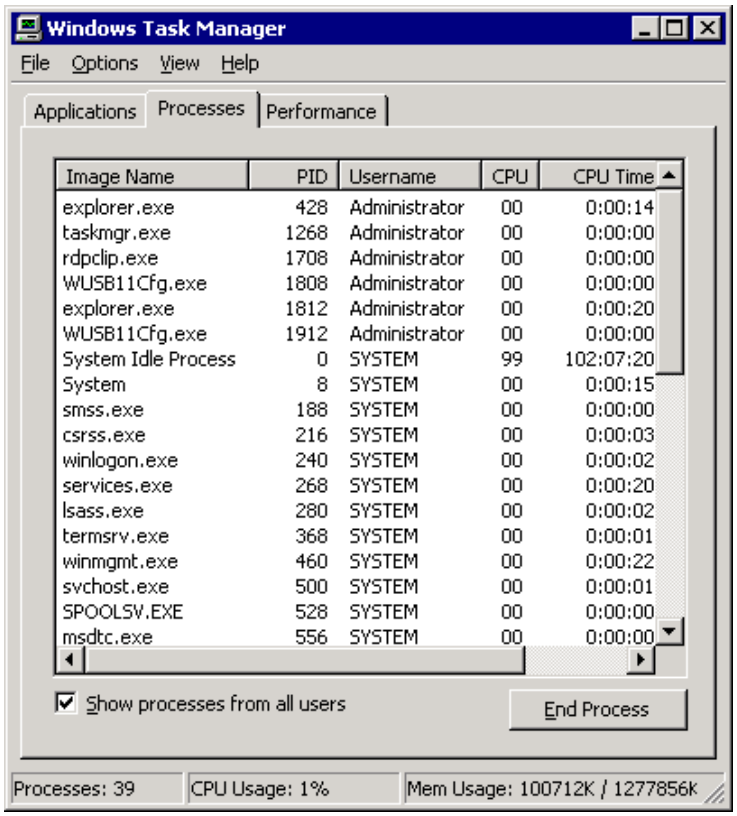

Figure 5.8: The Task Manager GUI.

Load Balancing

The Windows Load Balancing Service is the most powerful feature in a terminal server environment. With it, you can build server farms that can handle far greater user load than an individual server. The Windows Load Balancing Service also creates a layer of fault tolerance in your design—if one server goes down, users can connect to other servers in the cluster. Granted, any users logged onto the crashed system will be impacted, but they will be able to log back onto the terminal server without having to go looking for another hostname.

At a very basic level, the Windows Load Balancing Service creates a Virtual IP address (VIP) and hostname for a group of servers. When a request or connection is made to the VIP or hostname, the load-balancing service decides to which physical server to route the request based on the number of connections on each server.

> Don't confuse the Windows Load Balancing Service with Windows Cluster Service. Both services are only available on Advanced Server and Datacenter Server. The difference between the two services is that while the Windows Load Balancing Service is like a traffic cop directing sessions to one server in the group, Windows Cluster Service would create parallel connections to multiple servers and run all requests and applications on all servers, therefore creating a truly fault-tolerant design. Unfortunately, only Windows Load Balancing Service can be used with Terminal Services. Although you will often see the term "cluster" applied to either type of server group, I will use the term farm to describe a group of load-balanced servers.

Installing Load Balancing

To install Windows Load Balancing Service, go to the Network Control Panel applet and enable Network Load Balancing in the properties of the network connection definition. If Network Load Balancing is not listed, you can add it by clicking Install, then selecting Service. Figure 5.9 shows Network Load Balancing enabled.

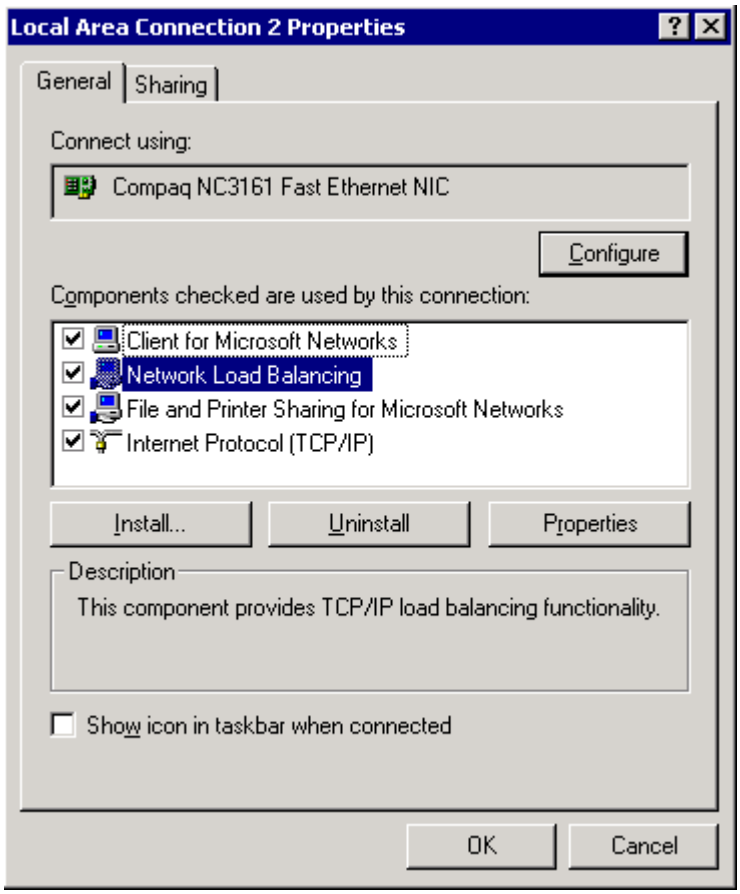

Figure 5.9: Enabling Network Load Balancing.

Before you can configure Windows Load Balancing Service, you will need to add an additional IP address to the TCP/IP protocol. To do so, highlight the Internet Protocol (TCP/IP) option, and click Properties. In the resulting Advanced TCP/IP Settings window, which Figure 5.10 shows, you will see the IP address for the server. The first address in the list should always be the actual physical IP address of the server. Click Add, and enter the VIP address you want to use for the farm. You will need to repeat this addition on all servers in the farm. The VIP you are adding will be the same for all servers in the group.

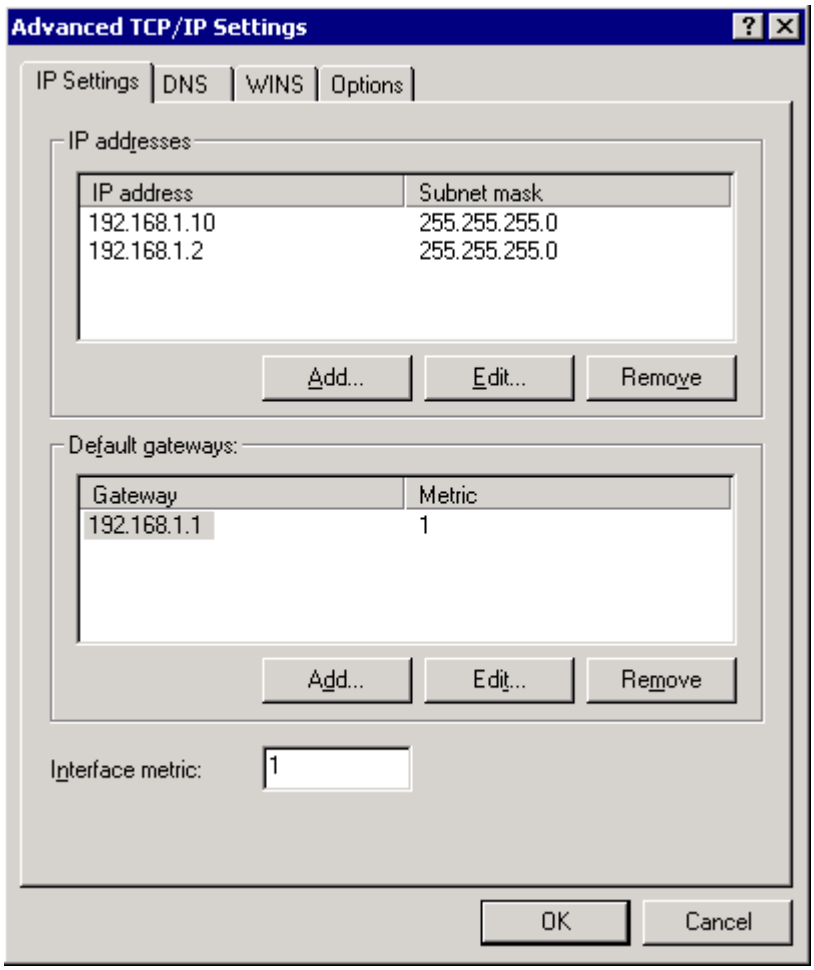

Figure 5.10: Adding the cluster's IP address to each server.

Once you have added the cluster IP to each server, you are ready to configure the Network Load Balancing Service. You do so by going to the properties of the service in the Network Control Panel applet of each server. On the Cluster Parameters tab, which Figure 5.11 shows, you enter information about the farm—the VIP address and hostname and a password if you want to be able to manage the cluster service remotely through a command-line utility. This page should be the same on all servers in the farm.

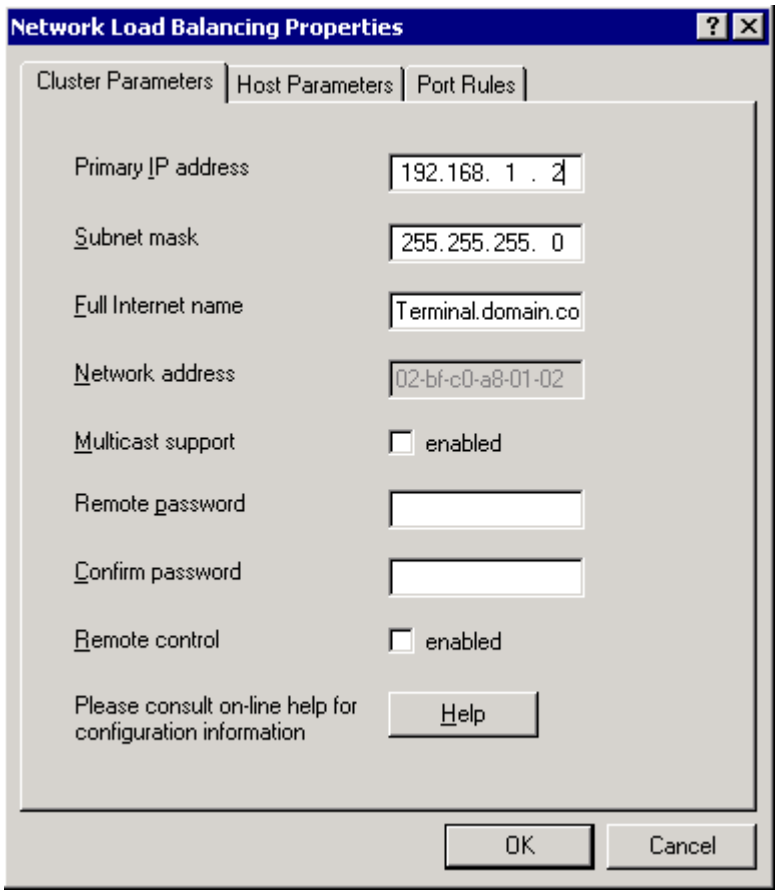

Figure 5.11: Cluster parameters on the load balancing properties sheet.

Figure 5.12 shows you the Host Parameters tab, which describes the individual host that you are configuring. Here you enter the physical IP address of the server and give the server a priority number that must be unique for each server. The priority setting comes into play when a member of the cluster is not available. The service will then use the priority setting to select which server should take over new requests to the cluster: the lower the number, the higher the priority.

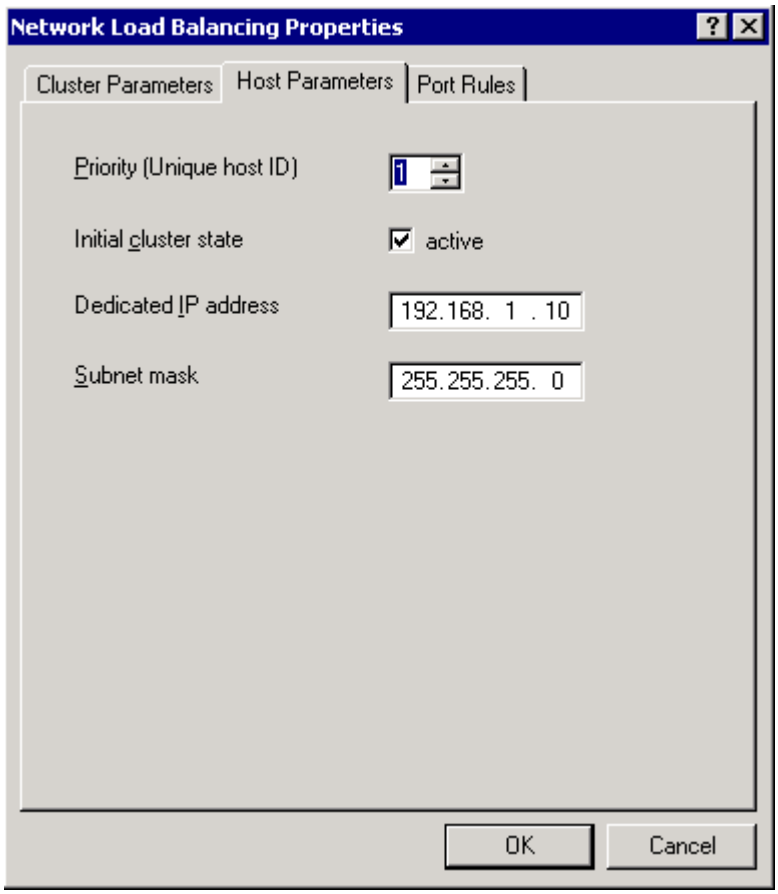

Figure 5.12: Host parameters of the load-balancing service.

The last tab in the properties sheet is the Port Rules tab. When using the load-balancing service for multiple network or database applications, this tab can be very complex. But if you are only worried about running Terminal Services, configuring port rules is quite simple. By default, the entire range of TCP ports is set to balance among the servers; although this setting will work for Terminal Services, you are better off restricting the load balancing to just RDP. This restriction will keep the service from having to evaluate every network request that is directed at the VIP. RDP uses port 3389, so edit the default port rule to filter only this TCP port number. Leave the filtering mode set to multiple hosts, and the affinity mode set to single.

If all servers in the cluster are running equivalent hardware, you can leave the load weight set to equal, but if you have some servers that are more powerful than others, you can specify a percentage of the requests to be handled by each host. Figure 5.13 shows these settings.

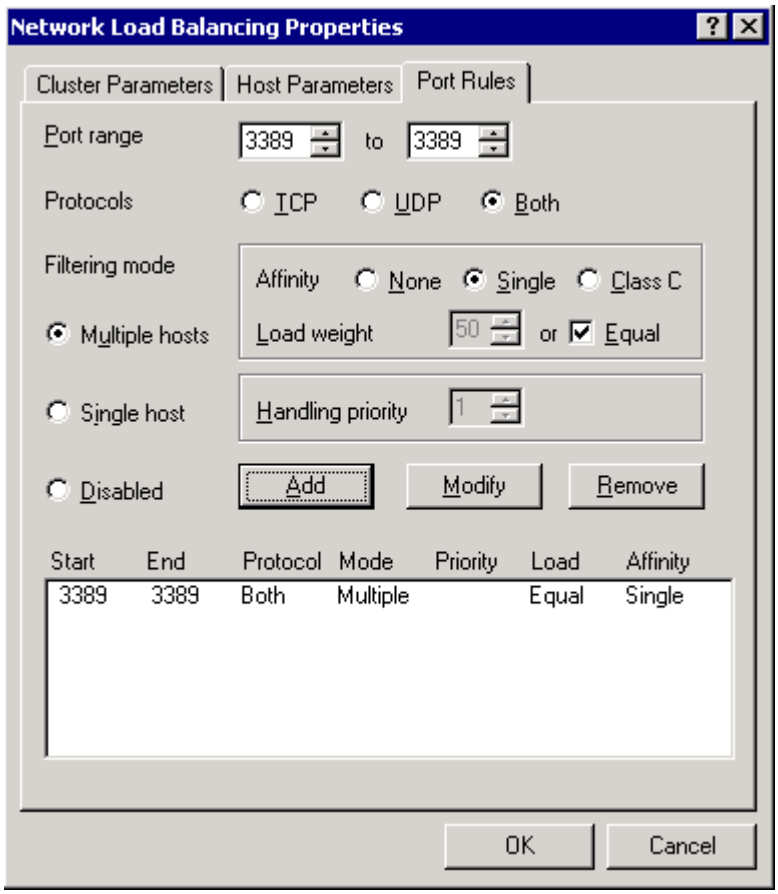

Figure 5.13: The Port Rules tab of a Terminal Services cluster.

Affinity Mode

Affinity mode is very important in a farm of terminal servers. Affinity remembers which server accepted the last request from a specific client IP address. This way, if a user loses his or her connection to the server, the user can reconnect to the same server in the cluster and re-establish the same session (assuming you are allowing a disconnected session to stay alive). One problem that this behavior might create, however, is if a user tries to reconnect from a different client device, the client's IP address will be different, and therefore affinity will not apply. Keep this functionality in mind when considering whether to use load-balancing service, especially for remote access, with which a user's IP can change often.

Controlling Load Balancing

Once you have installed and configured load balancing, you can control the farm from a command line. The Windows Load Balancing Service commands are used to control the service. With them, you can start, stop, enable, disable, or drain the server. The most common command is used to drain a server:

```
WLBS DRAIN ALL
```

This command tells the server to stop accepting new requests but doesn't affect existing connections. Once the last user on the server has logged off, you can perform maintenance or reboot the server without impacting users.

> The Help file for load balancing is very thorough and informative. To access it, click Help in the load balancing properties sheet or type WLBS HELP from a command line.

Figure 5.14 shows an example of a Terminal Services load-balancing group.

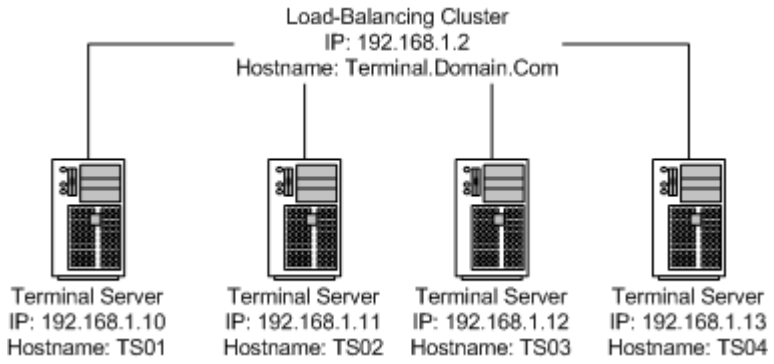

Figure 5.14: Example of a Terminal Services load-balancing cluster.

Application-Level Load Balancing

The major shortcoming of using Windows Load Balancing Service in a terminal server environment is that every server in the farm must have the same applications loaded on it. If you have an application that only a small number of users need, you will still need to load it on every server in the farm.

By using third-party products such as New Moon's Canaveral IQ or Citrix's MetaFrame, you will be able to perform load balancing on a per-application basis instead of at a per-server level. This way, you can publish your core applications from many servers, but offer lesser-used applications from only a few servers. The other major advantage to using a third-party add-on for load balancing is that these products do not require the additional expense of Win2K Advanced Server—you can implement them on the standard Win2K Server OS. Of course, these products offer many more advantages than just application-level load balancing, so be sure to consider them in your Terminal Services design.

Terminal Server Maintenance

In a perfect world, Windows systems would always perform as they are supposed to. Users would be able to log on, log off, open and close applications, and recover from hung applications without causing problems to the system or other users. In reality, however, most Terminal Services administrators have found that applications will occasionally cause memory leaks and user profiles will sometimes become locked on the terminal server. Although your goal should be to find the cause of such problems and correct them, a very helpful task is to implement a maintenance schedule for you terminal servers.

> We all accept that we have to reboot a workstation occasionally to clear up a problem. Microsoft has made great strides with Win2K to reduce the frequency of these types of problems, but when you consider that a terminal server is actually many virtual workstations running in one box, you realize that weekly reboots are more for preventative maintenance than an indication of a problem. Please do not take this suggestion as an indication that all Windows systems require regular reboots—doing so is a terminal server-specific practice.

Under WTS, a common practice was to have servers with heavy utilization perform maintenance reboots nightly. Win2K is more robust, and most administrators have found that running weekly maintenance scripts is sufficient. You can perform any number of actions during a maintenance script, but the most common actions are to

1. Disable new logons.
2. Warn any users that are currently logged on that the system will be rebooting so that they can save their work and log out.
3. Force any remaining users to log out.
4. Delete any locally cached user profiles that were not deleted when the user logged off (you should only do so if you are using roaming Terminal Services profiles).
5. Clean the print spooler.
6. Reboot the server.

You can write your maintenance script in any scripting language that you are comfortable with, but because many of the commands you will use are native to Windows, Shell Script is perhaps the easiest to use. Here are a number of commands and utilities that are useful in writing maintenance scripts (these commands are not case sensitive):

- MSG—Native on Terminal Services, this command sends a popup message to a user on the terminal server. When followed by an asterisk (*), this command sends the message to all users on the server.
- SLEEP—This command is a resource kit utility. You should copy this utility to the \winnt directory of you server, as it is very useful in scripting. SLEEP pauses a batch file for a specified number of seconds.
- DELPROF—Native on Win2K, this command deletes local copies of user profiles. When followed by /I /Q, it will ignore errors and not wait for you to OK each deletion.
- CHANGE LOGON /DISABLE—Native on Terminal Services, this command disables new logons to the server.
- LOGOFF RDP-TCP < YES.TXT—Native on Terminal Services, this command logs off all users connected through RDP. Because there is no quiet switch, you have to pipe a yes command to LOGOFF from a text file containing a Y and a carriage return.
- TSSHUTDN /reboot—Native on Terminal Services, this command reboots the terminal server.
- Net Stop Spooler—Native on Win2K, this command stops the print spooler so that you can delete any orphaned spool files.

Listing 5.1 shows a sample maintenance script.

```
msg * Please save your work and log off. Weekly reboot in 10 minutes.

sleep 600

msg * Save your work now and log off. Nightly reboot in 5 minutes

sleep 300

msg * Nightly reboot in progress. You will be logged off in 30 seconds

sleep 30

logoff rdp-tcp < yes.txt

delprof /i /q

net stop spooler
del %systemroot%\system32\spool\printers\*.* /q

tsshutdn /REBOOT
```

Listing 5.1: An example maintenance script.

> Occasionally, a user's profile will become locked in the registry of the terminal server. In this case, the DELPROF command will not be able to clear it until you either manually unload the user hive by using regedt32 or rebooting the server. There is a shareware utility floating around the Internet called WTS-REGUNLD that will unload all user hives so that you can remove them all with DELPROF.

Once you write and test your maintenance scripts, you can schedule them to run by using the Windows Schedule service (the AT command). If you are using load balancing, be sure to schedule each server at a different time or on a different day so that at least one member of the cluster is available at all times. To schedule the script to run every Monday at 2:00AM, you would use the following command:

```
AT 2AM /every:mon script_name
```

When implementing scheduled reboots, you must be careful to not leave a session logged on at the console of the terminal server. Certain system messages at the console will prevent the server from rebooting.

> Be careful when using the TSSCHUTDN command. If you type this command with no switches, the default behavior is to shut down the server without rebooting. In the appendix, I will give a VBS script that you can use as a wrapper for TSSHUTDN to prevent inadvertent shutdowns.

Summary

In this chapter, I went into detail about the capacity-planning tools that Microsoft provides. I also examined the Performance Monitor counters that will assist you in watching the health and capacity of your terminal servers. We also looked at the Windows Load Balancing Service—one of the most powerful options available for high-availability Terminal Services designs. Finally, we went over maintenance scripting and the common commands you will use in maintenance

scripts. All the tools and procedures available to you will help you design and maintain a stable, fault-tolerant Terminal Services infrastructure.

In Chapter 6, I will take you through how to secure your Terminal Services design. We will look at the network ports required for terminal server access and set up Terminal Services access through a firewall. I will also go over some common pitfalls that can occur when untrained users and administrators are given Admin rights on a terminal server.

Copyright Statement

© 2001 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.

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