

NETWORK LOAD BALANCING

By Jonathan Held

LOAD BALANCING

Over the last decade-and-a-half, Internet use has dramatically increased, placing an extraordinarily high level of demand on underlying hardware. In order to keep up with the increase in user requests and preclude saturation of hardware resources, the hardware itself has become much more powerful and capable. However, the key to successfully serving a customer base that continues to grow is recognition that the solution cannot be achieved merely by investing large sums of money in the latest and greatest hardware. Rather, the answer lies in an understanding of how the network can be used to your advantage and how you can distribute requests to many servers within a cluster that can then process them in an expeditious manner. This concept, aptly called load balancing, is neither complex nor novel, and appropriately used, it can help ensure that no server becomes so overburdened with requests that it ends up failing to properly function. Load balancing has been around for years. Some load balancing implementations have been hardware-based while others only required installation of special software.

In this article, we'll explore Microsoft's Network Load Balancing (NLB), a software-based solution that allows you to effortlessly cluster multiple machines.¹ There are a number of significant benefits to deploying a NLB solution, including the following:

1. NLB is protocol agnostic. It works with any TCP or UDP application-based protocol. This means that you can configure a variety of NLB clusters within your organization, and each cluster can have its own specific function. For example, one cluster may be dedicated to handling all Internet-originated HTTP traffic while another may be used to serve all intranet requests. If your employees have a need for transferring files, you can centralize file storage and closely monitor both uploads and downloads by creating a FTP cluster. Lastly, if there's a requirement for secure remote access to the corporate network, NLB supports the PPTP protocol which can be used by employees to establish a Virtual Private Network (VPN) connection. NLB support for PPTP is significantly less costly than many other alternative VPN solutions.

Separating cluster functionality as previously described results in a number of distinct advantages:

- a. You'll avoid saturating cluster machines with too many requests. Such a strategy should result in a noticeable increase in the available uptime of invaluable resources.
- b. When problems occur, your technical support staff should be able to troubleshoot and restore the system in a much more timely fashion. In the extreme case where an entire cluster needs to be brought down for maintenance, other clusters will remain unaffected and can continue servicing requests. The converse is not true when one cluster is multi-purposed.
- c. Functional isolation of cluster resources allows you to architect your network in a way that is conducive to securing your most valuable corporate resources. By placing a firewall between each cluster and those who use it, you can allow communication to occur over only the ports that are required by that particular protocol. While locking down the network in this fashion doesn't completely preclude the possibility of a

security vulnerability from occurring, it can provide damage containment if an exploit is limited to a particular piece of software (e.g., if the FTP application you were running was vulnerable, only the FTP cluster would need to be repaired).

- d. By configuring each machine in the cluster to run only the software it needs in order to perform its function, it's easier for system administrators to maintain the cluster.
2. Another major benefit of using NLB over other load balancing solutions is that it has a fault tolerance capability. Many other load balancing implementations, such as Round Robin DNS (RRDNS), continue to send requests to servers that have "died" until system administrators pick up on the fact that there is a problem and then manually perform a configuration change. NLB, however, can automatically pull a server from the cluster if it fails to respond. It can handle scenarios where a power supply has burnt out, a network card has gone bad, the primary hard disk has crashed, et cetera.
3. Just as servers can be pulled from a cluster when problems arise, they can also be added. NLB provides a simple, command-line utility that allows you to effortlessly add a new machine to a cluster. So, as cluster activity increases and server performance begins to degrade due to the large volume of requests, you can scale your cluster to an appropriate size to decrease server load. If you ever find that you have more machines than you need in your cluster, you can pull them out and dedicate them to other organizational tasks.
4. By far, one of the biggest advantages of NLB is its ease of use. NLB installs only a networking driver component – absolutely no special hardware is required. Not only does this facilitate the deployment of a load balancing solution, but it also significantly reduces costs.

Load balancing is extremely important and it is fundamental to the operational success of some of the most recognized, high-traffic Websites visited today. Hotmail, MSN, Yahoo, Google, CNN, and USATODAY all have content that they're providing to millions of Internet users. In order to effectively deliver this content, these Websites have been deployed on a large number of servers that are clustered together. In order to determine how load balancing can benefit your organization, you'll need to take some time to perform capacity planning. Evaluate how large your current customer base is, project growth rates, and use this research to build a configuration that can effectively respond to requests. And remember: load balancing, while primarily applied to web servers, can be used with a variety of other services.

In the sections that follow, we'll take an in depth look at Microsoft's NLB solution. We'll not only see how it works, but also what affect it has on the network, how to properly configure it, and what types of tests you can perform in order to ensure that a cluster is operational.

ROUND ROBIN DNS (RRDNS)

Prior to the advent of NLB, Round Robin DNS was used to manage server congestion. With Round Robin, a DNS server contains multiple "A" records for a single host, e.g., the Internet resource www.auerbach-publications.com might correspond to three Internet Protocol (IP) addresses: 208.254.79.10, .11, and .12. The machines with these IP addresses are all identically configured – each is running a web server that has a complete copy of the Auerbach Publications Website, so no matter which server a request is directed to, the same response is provided.

This elementary "load balancing" mechanism works as soon as a DNS query is made. When a client attempts to access the Website, a local DNS lookup is performed to determine what the corresponding IP address is. The first time this query is made, the remote DNS server returns all the address records it has. The local DNS server then determines what address record to return to the client. If all records are returned, the client will take the first one that it is given. With each request, the Round Robin algorithm rotates the order in which the address records are returned,

so each DNS query will result in a client using a different IP address. When the fourth query is made, the address records are returned in the same order as the first.

This process effectively distributes the load across all servers. It is extremely simple to implement and scales quite well. However, there are a number of significant disadvantages in using it. First and foremost, some clients will cache the lookups they've performed in order to improve performance.² Successive queries may not be performed because the address resolution has already been performed. The end result is that the same IP address record will be returned to multiple clients. This caching all but breaks this load balancing scheme. Secondly, RRDNS doesn't deal well with machines that are non-responsive. The DNS server has absolutely no means for monitoring the health of individual hosts. Consequently, a DNS server using the Round Robin algorithm could very well return the IP address of a server that has been turned off or one that's on but has had its web service crash. Lastly, there are occasions when session state is important and you need to tie a client to the same server, which is something that cannot be done using Round Robin DNS.

THE BASICS BEHIND MICROSOFT'S NLB

Microsoft's NLB solution is oftentimes misinterpreted as being a dedicated box that mediates requests to other servers that are part of a cluster. This assumption is grossly incorrect. Rather, it is a software-based implementation that runs on every cluster node.³ NLB works by using a hashing algorithm that takes the IP address or IP address and port of an incoming request and determining a priori what node (host) in the cluster will process that request.⁴ Every node within the cluster receives every packet of traffic, but only one node will ultimately end up servicing a request. The determination as to what node is responsible for responding is made by applying a filter to each packet.

Of note is that as new machines join the cluster, or, conversely, a node is removed, the algorithm must be re-executed so that the load is distributed correctly among all active nodes. This re-evaluation process is extremely important and is called convergence. It is also important to realize that at no time is NLB ever aware of what the load is on any particular cluster node. NLB cannot determine whether a node's CPU usage is extremely high or that a node has little to no available memory to process a request. Should a node in the cluster ever experience such resource saturation, NLB will continue sending it requests until the node stops sending heartbeat messages⁵, at which time the node is automatically pulled from the cluster. However, if the node continues sending heartbeat messages, it will remain in the cluster, albeit in an unusable state.

In order to monitor the state of the cluster, all nodes broadcast a heartbeat message of 1500 bytes per second to all other nodes. In this manner, any single node in the cluster can easily determine when a convergence operation is required. The frequency that these heartbeat messages are broadcast can be explicitly set by changing the value of the *AliveMsgPeriod* registry key (Exhibit 1). If you decide to change this value (1000 milliseconds), it must be changed on every node in the cluster to preclude NLB functionality problems. There are a host of other NLB-related settings that can be configured only by using the registry.

While 1500 bytes per second per node may seem to impose a huge bottleneck on network bandwidth, it is in reality only a very minor impediment. On a 10 Mbps Ethernet backbone with 10 nodes part of a cluster, a quick calculation reveals that heartbeat messages account for only 1.2% of the total available bandwidth:

$$\frac{10 * \frac{(1500 \text{ bytes} + 26 \text{ bytes}) * \frac{8 \text{ bits}}{\text{byte}}}{s}}{10000000 \frac{\text{bits}}{s}} = \frac{122080}{10000000} = 0.012208 = 1.2208 \%$$

If the cluster increased in size to 32 nodes, the portion of bandwidth utilization consumed by the heartbeat messages simply multiplies by a factor of 3.2, becoming 3.90656%. Hence, NLB is extremely efficient in the manner in which it communicates cluster status, and it will not adversely affect your existing network topology or require major restructuring in order to accommodate it.

Name	Type	Data
(Default)	REG_SZ	(value not set)
AliveMsgPeriod	REG_DWORD	0x000003e8 (1000)
AliveMsgTolerance	REG_DWORD	0x00000005 (5)
ClusterIPAddress	REG_SZ	0.0.0.0
ClusterModeOnStart	REG_DWORD	0x00000001 (1)
ClusterName	REG_SZ	cluster.domain.com
ClusterNetworkA...	REG_SZ	02-bf-00-00-00-00
ClusterNetworkMask	REG_SZ	0.0.0.0
ClusterNICName	REG_SZ	PCI\VEN_10b7&DEV_9200&SUBSYS_00b41028&REV_7...
ConnectionClean...	REG_DWORD	0x000493e0 (300000)
DedicatedIPAddress	REG_SZ	0.0.0.0
DedicatedNetwor...	REG_SZ	0.0.0.0
DescriptorsPerAlloc	REG_DWORD	0x00000200 (512)
HostPriority	REG_DWORD	0x00000001 (1)
InstallDate	REG_DWORD	0x3d13b312 (1024701202)
IPChangeDelay	REG_DWORD	0x0000ea60 (60000)
IPToMACEnable	REG_DWORD	0x00000001 (1)
LicenseKey	REG_SZ	
MaskSourceMAC	REG_DWORD	0x00000001 (1)
MaxDescriptorAllocs	REG_DWORD	0x00000200 (512)
MulticastARPEnable	REG_DWORD	0x00000001 (1)
MulticastSupportE...	REG_DWORD	0x00000000 (0)
NBTSupportEnable	REG_DWORD	0x00000001 (1)
NetmonAliveMsgs	REG_DWORD	0x00000000 (0)
NumActions	REG_DWORD	0x00000064 (100)
NumAliveMsgs	REG_DWORD	0x00000042 (66)
NumberOfRules	REG_DWORD	0x00000001 (1)
NumPackets	REG_DWORD	0x000000c8 (200)
ParametersVersion	REG_DWORD	0x00000004 (4)
PortRules	REG_BINARY	00 00 00 00 ff ff 00 00 00 f0 ff 6f 02 00 00 00 03 00 0...
RemoteControlCode	REG_DWORD	0x00000000 (0)
RemoteControlEn...	REG_DWORD	0x00000000 (0)
RemoteControlJD...	REG_DWORD	0x000009c8 (2504)
RemoteMaintena...	REG_DWORD	0x00000000 (0)
ScaleSingleClient	REG_DWORD	0x00000000 (0)
VerifyDate	REG_DWORD	0x97b70951 (2545355089)
VirtualNICName	REG_SZ	{Device}\{26AA2A75-6AA1-4D27-A2A1-C014EC211579}

EXHIBIT 1. NLB-configurable registry settings. These settings are found under *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WLBSP\Parameters*.

CONFIGURING NLB IN SEVEN SIMPLE STEPS

Microsoft's NLB solution is not tremendously difficult to use. The corresponding User Interface (UI) that's provided allows you to quickly and easily configure cluster, host, and port rule settings. In this section of the article, we'll familiarize ourselves with some additional NLB-related terminology, but most importantly, we'll outline the seven simple steps you need to follow in order

to get a host up and running as part of a cluster in a matter of mere minutes.

Step 1. The first step in setting up NLB is perhaps the easiest. NLB, contrary to popular belief, only requires one Network Interface Card (NIC). Each node that's part of the cluster should have a static IP address assigned to the network adapter (or NIC) that will communicate with the other cluster nodes. You can make this IP assignment by editing the properties of the appropriate adapter found in the *Network and Dial-up Connections* Control Panel applet. On the *Properties* dialog that appears when the network adapter is selected, ensure that *Internet Protocol (TCP/IP)* component (Exhibit 2) is installed and, using the left mouse button, double-click it. The dialog (Exhibit 3) that appears gives you two options: You can either dynamically obtain an IP address for the NIC or explicitly set it. Choose the latter option and use an appropriate IP address, subnet mask, and DNS server. After configuring your network adapter's TCP/IP settings, ensure that the Network Load Balancing component is installed (checked) as well.

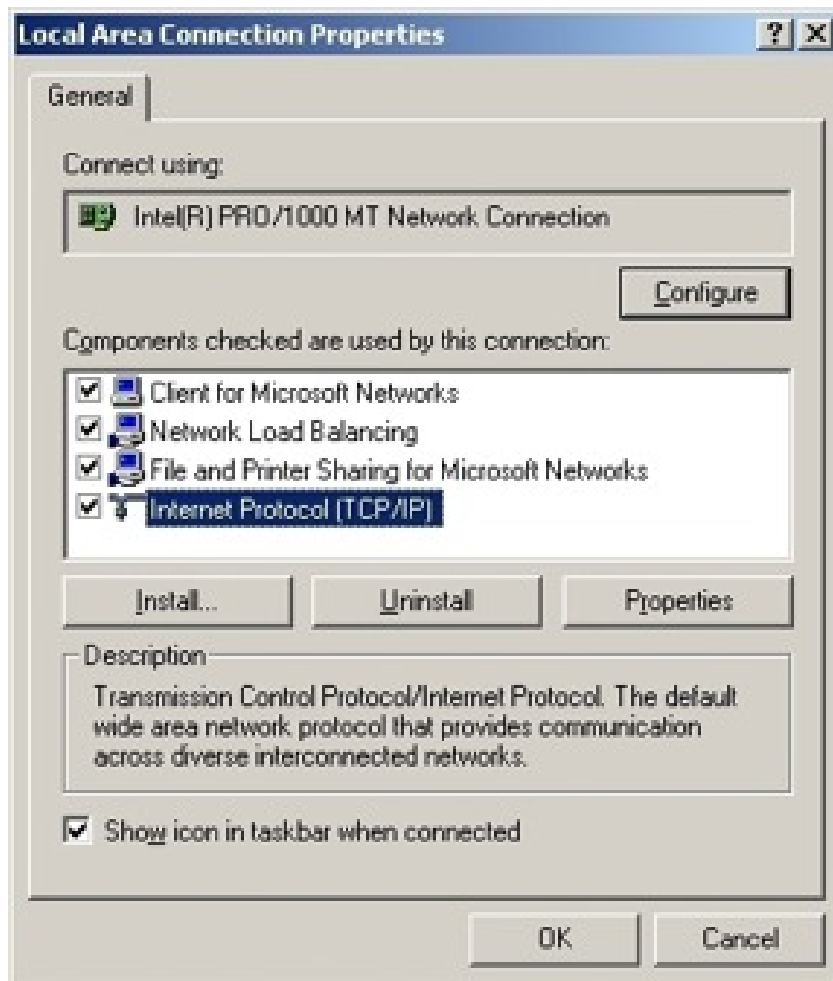


EXHIBIT 2. Modifying the TCP/IP component bound to the network adapter used for cluster communications.

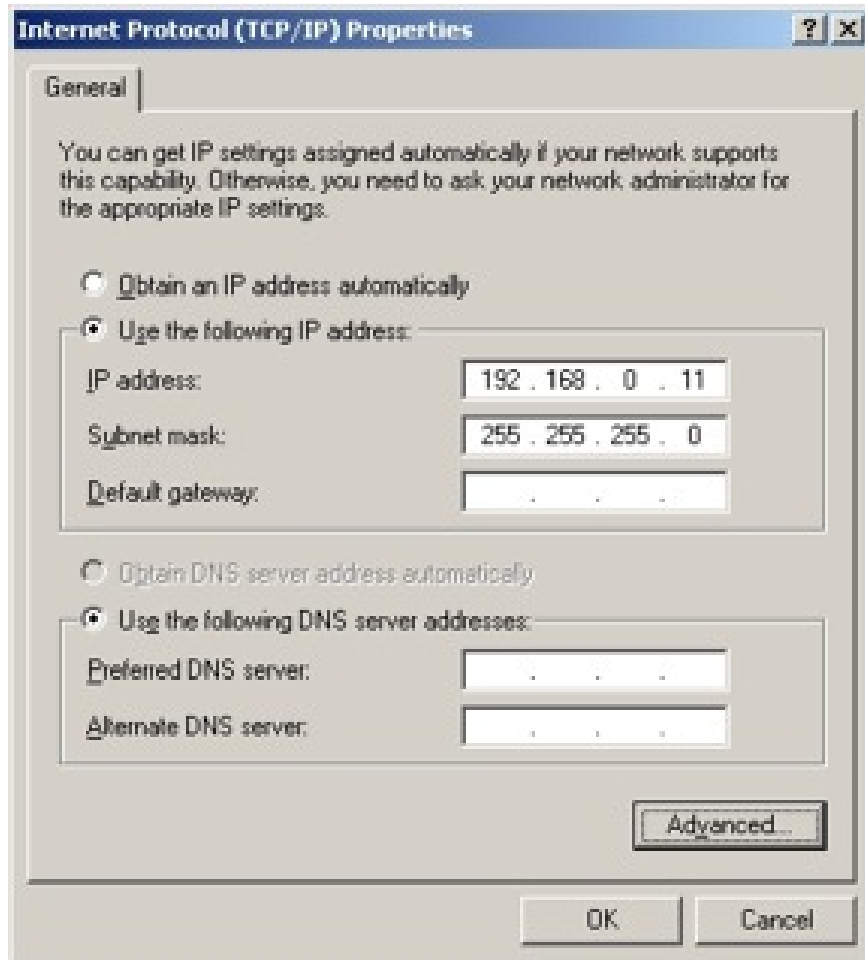


EXHIBIT 3. Assigning a static, Class C IP address to a network adapter.

Step 2. After you've made the static IP address assignment, click the *Advanced* button on the *Internet Protocol (TCP/IP) Properties* page. Use the *Advanced TCP/IP Settings* dialog to add a second IP address to the network adapter. This entry represents a virtual IP address, an address that is shared among all cluster hosts and is used by clients when they make a request. In Exhibit 4, we've designated the virtual IP address as 192.168.0.220. It is extremely important that the virtual IP address you enter uses the same network id as the static IP address, otherwise, NLB will fail to properly function.

Step 3. With the NLB software installed, you can now configure various cluster parameters. Double-click the NLB component shown in Exhibit 2 and a *NLB Properties* page with three tabs is displayed (Exhibit 5). Each tab is used to configure a NLB-specific property. On the first tab, labeled *Cluster Parameters*, you'll need to enter values for at least the first three settings.

The *Primary IP address* is the same as the virtual IP address. This association is required so that NLB can determine which of the two IP addresses you assigned to the network adapter represents the virtual IP address. After entering this IP address, Windows will automatically determine the subnet mask, which should match the value that was previously used.

The *Full Internet name* represents the fully qualified domain name (FQDN) that clients will use to address the cluster. Before deploying your cluster, you must set up a DNS server and add an entry that resolves the FQDN to the virtual IP address of the cluster. Alternatively, if you're only

internally testing cluster functionality, you can avoid the trouble of having to install and configure a DNS server by simply modifying your client's *hosts*⁶ file as illustrated in Exhibit 6.

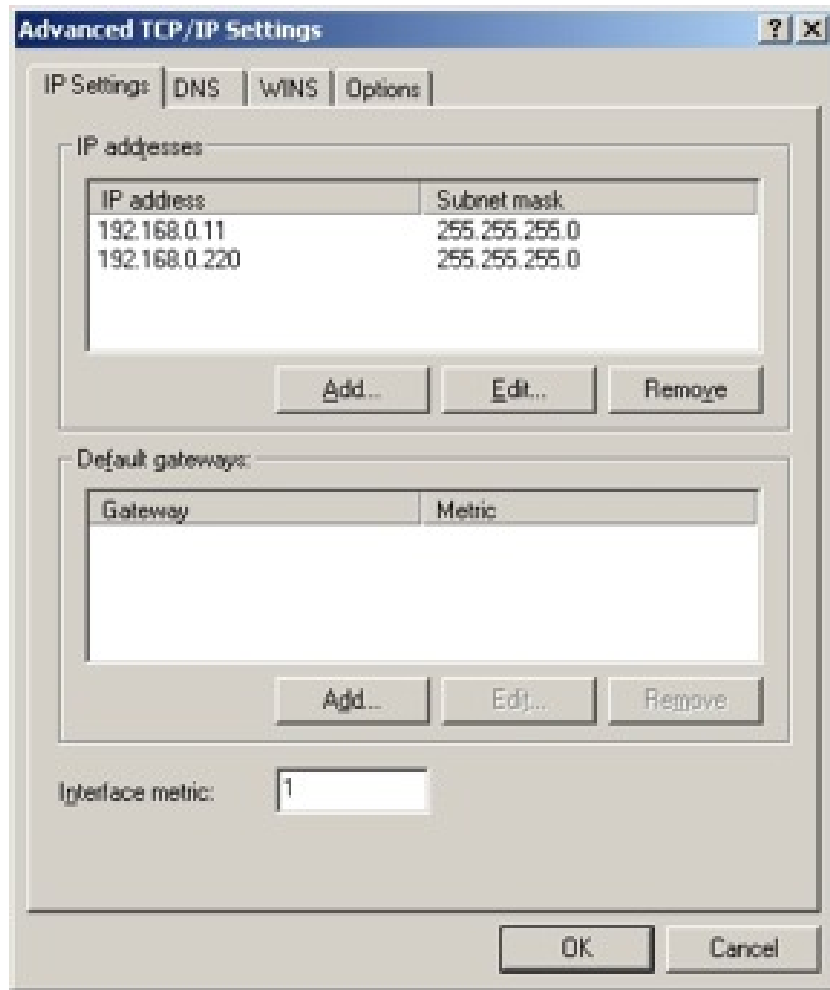


EXHIBIT 4. Adding the virtual IP address 192.168.0.220 to a network adapter using the advanced TCP/IP settings.

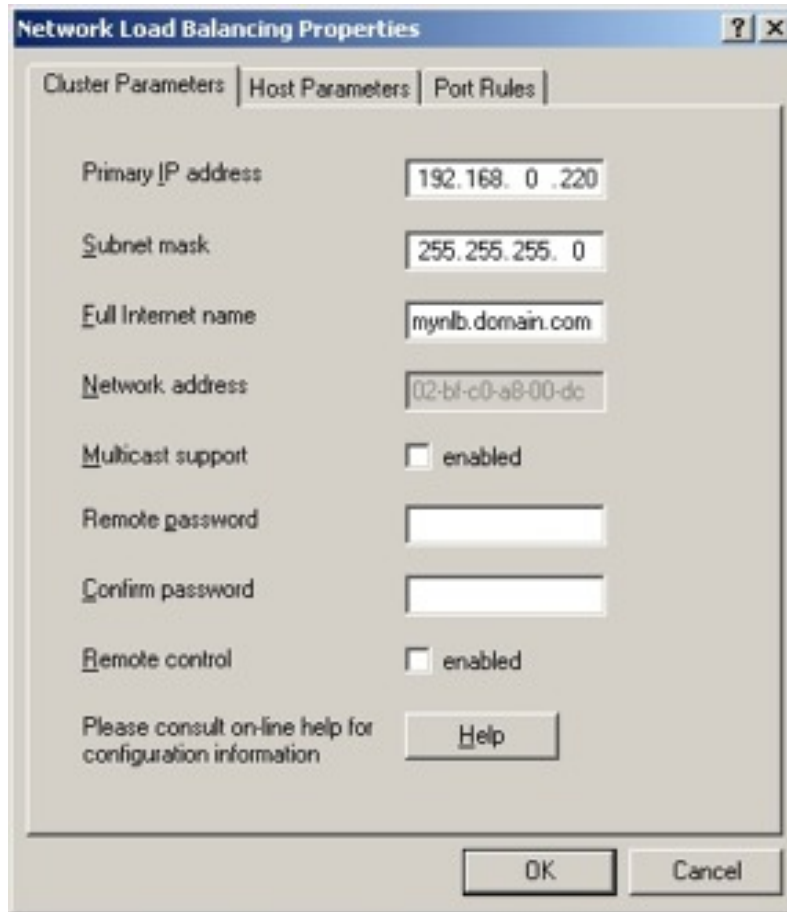


EXHIBIT 5. Configuring NLB cluster parameter settings.

```
# Copyright (c) 1993-1999 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each entry should be
# kept on an individual line. The IP address should be placed in the first column
# followed by the corresponding host name. The IP address and the host name should be
# separated by at least one space.
#
# Additionally, comments (such as these) may be inserted on individual lines or following
# the machine name denoted by a # '#' symbol.
#
# For example:
#
#       102.54.94.97      rhino.acme.com          # source server
#       38.25.63.10     x.acme.com             # x client host

127.0.0.1      localhost
192.168.0.220  mynlb.domain.com
```

EXHIBIT 6. Modifying the hosts file by adding an entry for the cluster's virtual IP address. Using the URL <http://mynlb.domain.com>, clients can reach the cluster without having to perform a DNS lookup.

Step 4. After configuring the cluster parameters, take a look at the four host parameter settings (Exhibit 7). The *Priority* or *Unique host ID* value is nothing more than an integer value from 1 to

32 assigned to the host. This setting specifies the host's priority for handling all network traffic that is not explicitly covered by the cluster's port rules. A lower number indicates a higher priority, and the host with the highest priority is called the default host. As implied by the label associated with this setting, each server in the cluster must have a unique number assigned to it. If unique numbers are not used, the WLBS service will log an error to the *System* event log similar to the following:

WLBS: duplicate host priority 1 was discovered on the network. Please check WLBS Setup dialog on all hosts that belong to the cluster and make sure that they all contain unique host priorities between 1 and 32.

The second host setting is a checkbox labeled *active*. This checkbox allows you to automatically have NLB add, or converge, the host into the cluster once you've completed configuring the various NLB properties. You can always uncheck this option and use the equivalent *wlbs* command later to accomplish the same task. The last two host settings – *Dedicated IP address* and *Subnet mask* correspond to the static IP address assignments that were previously made.

Step 5. The last step in configuring NLB is to define appropriate port rules. In Exhibit 8, rules have been explicitly defined for only two ports: 80 (HTTP) and 443 (HTTPS). A rule consists of six components: a port range, the protocol that is supported, a filtering mode, a handling priority (used only when the filtering mode is set to single host), a load rule, and an affinity setting. To set a rule for handling HTTP traffic, set both the start and end port range to 80. While both protocols can be used, click the radio button next to TCP, since HTTP is a TCP-based protocol. The filtering mode should be left with the default value, *Multiple hosts*, since several cluster hosts will have identical port rules. By contrast, you can use the *single host* setting when enumerating a port rule that only one host in the cluster will handle, and the *disabled* filtering option can be used as an improvisational firewall since it allows you to explicitly block network traffic.

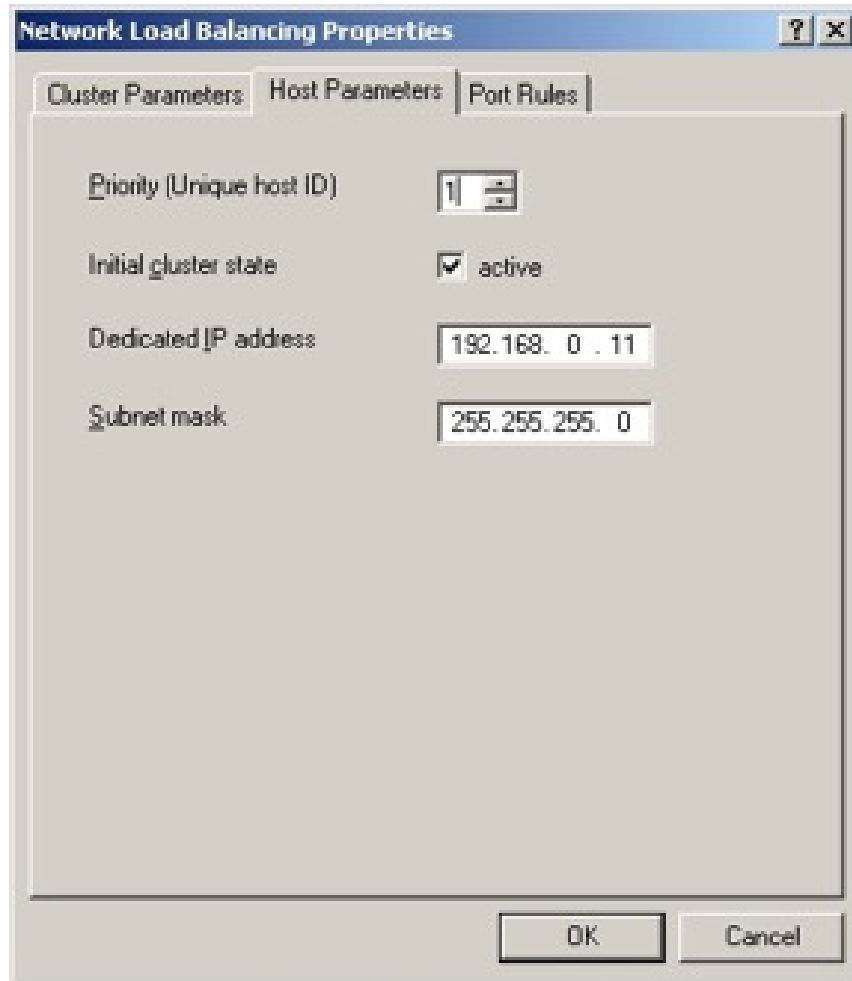


EXHIBIT 7. Configuring NLB host parameter settings.

The most interesting setting, *affinity*, is used to specify how you want to direct network traffic from a particular client. When *affinity* is set to none, multiple requests from the same client can go to any host in the cluster. This setting provides the best performance. However, it may adversely affect clients that are reliant on session-specific data because subsequent requests may be handled by other cluster hosts that do not have this information. In particular, when a web application relies on session state, you should use either *single* or *Class C* affinity because they ensure that every request from a client is directed to the same member of the cluster. You should also use *single* or *Class C* affinity when using HTTPS in order to ensure that client connections are always handled by the same server that established the Secure Socket Layer (SSL) session. The cost of negotiating a session over SSL results in performance degradation, so it is optimal to minimize the number of times that this process has to be done. By selecting one of these affinity options, you can perform that optimization.

Once all port settings have been selected, click the *Add* button to enable the port rule. The rules that are defined for the NLB host are displayed at the bottom of the dialog.

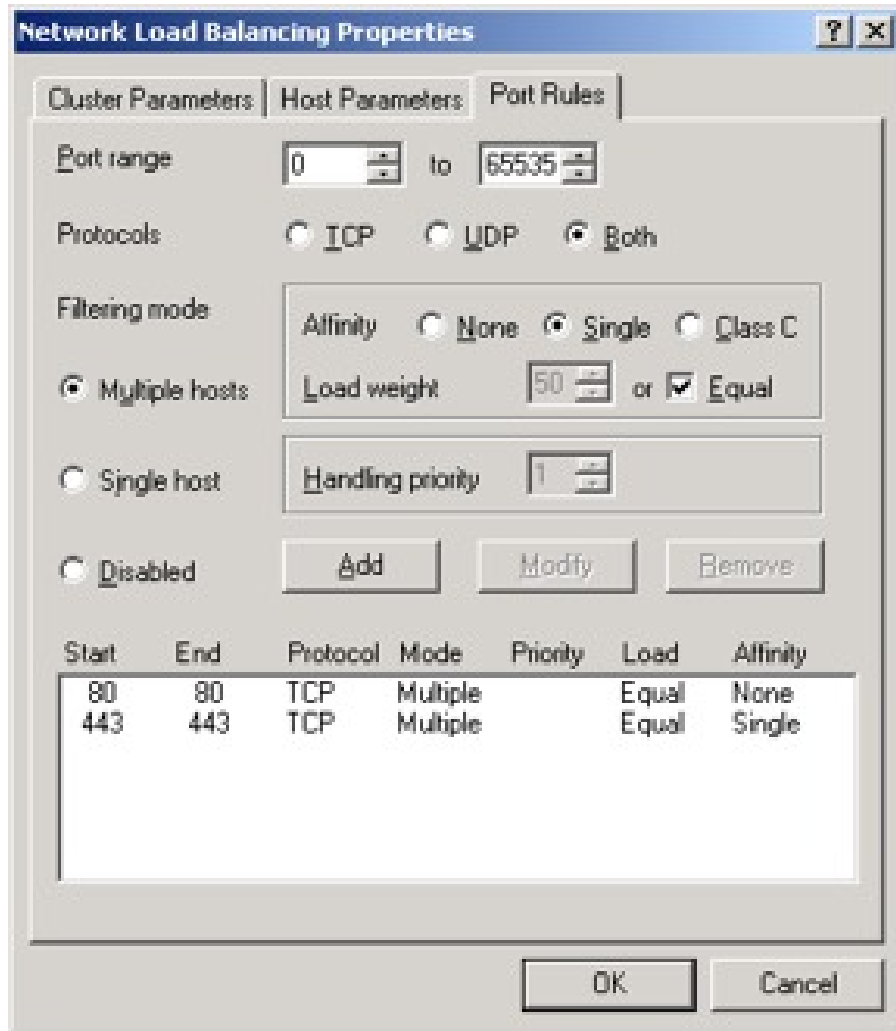


EXHIBIT 8. Defining port rules.

Step 6. If you didn't mark the initial cluster state as active, you'll need to add the node to the cluster by using the *wlbs start* command.

```

WLBS Cluster Control Utility V2.3. (c) 1997-99 Microsoft Corporation

Usage: WLBS <command> [<cluster>[:<host>]]

        [ /PASSW [<password>] ] [ /PORT <port> ] ]

<command>
  help                - displays on-line help
  ip2mac <cluster>    - converts cluster IP to cluster MAC address
  reload              - reloads parameters from the registry
  query               - queries which hosts are currently part of the cluster
  display             - displays configuration parameters, current
                      status, and last several event log messages
  suspend            - suspend control over cluster operations
  resume             - resume control over cluster operations
  start               - starts cluster operations
  stop                - stops cluster operations
  drainstop          - finishes all existing connections and
                      stops cluster operations
  enable <port> | ALL - enables traffic for <port> rule or ALL ports

```

disable <port>	ALL - disables ALL traffic for <port> rule or ALL ports
drain <port>	ALL - disables NEW traffic for <port> rule or ALL ports

EXHIBIT 9. The various WLBS commands.

Step 7. Repeat these steps for each node in the cluster.

TESTING THE NLB CLUSTER

Once you have added all nodes to the cluster, some simple testing can quickly reveal whether or not NLB is properly configured. First and foremost, you can issue a *wlbs query* command. This command will list all host id's that have successfully converged as part of the cluster. If the output of this command is not helpful, you can use *wlbs display* to view the last ten NLB-related event messages, which may provide more insight and assistance in troubleshooting. You should also consult the *System* event log and look for informational, warning, or error messages where the source of the log entry is WLBS.

Assuming that your web cluster is properly configured, you can easily test NLB functionality by creating a simple Active Server Page (ASP) and placing it in the root virtual directory of each web server in the cluster. This ASP page has only two lines of code:

```
<% Set obj=CreateObject("Wscript.Network") %>  
<%= obj.ComputerName %>
```

This code does absolutely nothing more than output the name of the computer on which the ASP page is executed. Once you load this page in your browser, a result similar to that shown in Exhibit 10 should be seen. Now that you know which server the client has hit within the cluster, go to that server and issue a *wlbs stop* command. This action will cause the host to become inactive and it will be pulled from the cluster. With the browser you previously used to hit the cluster, press F5 to refresh the page. After a short delay, you'll see the same test page produce a dramatically different result (Exhibit 11), which is a clear indication that NLB is functioning as it is supposed to.

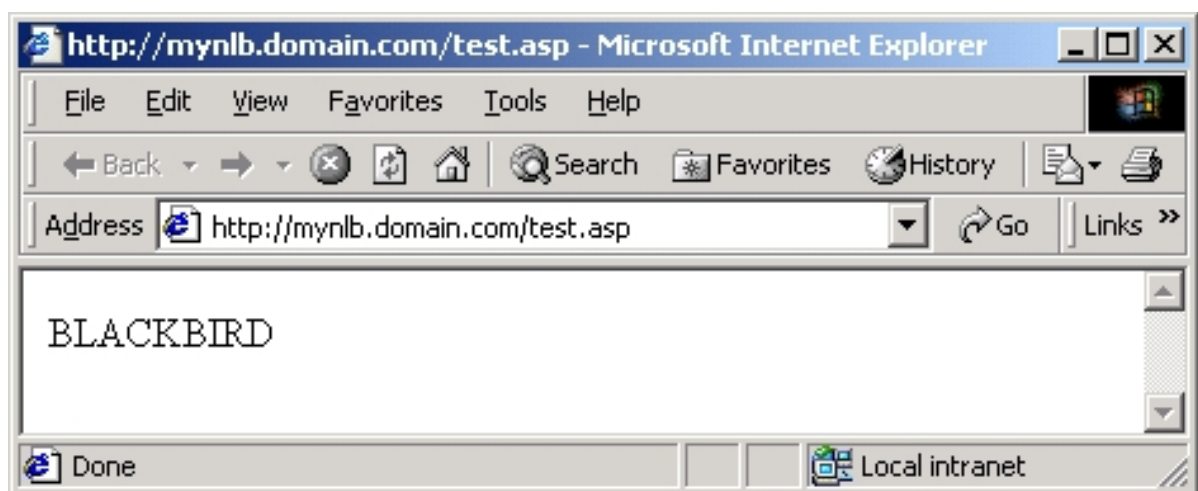


EXHIBIT 10. The *test.asp* page shows the client what NLB cluster host was "hit".

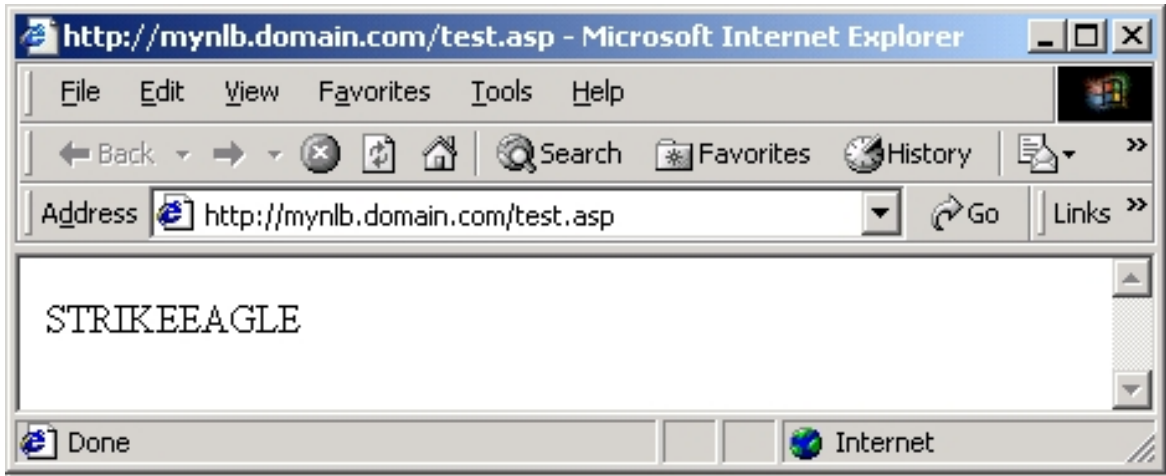


EXHIBIT 11. Result of hitting the test.asp page after the BLACKBIRD host was pulled from the cluster.

CONCLUSION

Popular Websites draw large audiences and millions of requests every day. While one web server could not possibly service all these requests, an appropriately configured cluster of web servers could through load balancing. Load balancing allows requests to be distributed evenly across all cluster hosts. While primitive solutions such as RRDNS have been around for a while, they typically have only addressed scalability problems. If a server in a RRDNS configuration dies, there is no knowledge on part of the load balancing algorithm and requests are sent to the server until the problem is identified and corrected by network operations personnel. Microsoft's NLB, however, is not only a scalable load balancing solution, but it is also one that provides high availability by automatically detecting the failure of a server and repartitioning client traffic among the remaining servers through the convergence process. Additionally, it allows network traffic to be distributed evenly across all servers or according to customized load rules that have been set. Microsoft's NLB is easy to configure and provides a robust, software-implemented load balancing solution that can be used with any TCP- or UDP-based application.

¹ In August 1998, Microsoft Corporation acquired Valence Research's Convoy Cluster software. The product was renamed Microsoft Windows NT Load Balancing Service (WLBS) and has subsequently become known simply as NLB. NLB can be found in the Windows 2000 Advanced and DataCenter Server editions as well as the recently released Windows Server 2003.

² Windows Operating Systems (OS) routinely cache DNS lookups that have been performed. To view the contents of the DNS client cache, use the command *ipconfig /displaydns*. You can flush the DNS cache by using the command *ipconfig /flushdns*.

³ Anywhere from 2 to 32 machines can be part of a NLB cluster. The 32 machine limitation is merely theoretically. The total number of machines that can effectively function as part of a cluster depends on many other factors, such as the application being load balanced, network topology, and the underlying hardware.

⁴ When NLB only uses the IP address to determine what cluster node should handle a request, single affinity is being used. With single affinity, requests from the same IP address will always go to the same node within the cluster. When NLB uses both the IP address and port to determine what node should handle a request, no affinity is being used. In this case, NLB may direct subsequent requests from the same client to different nodes within the cluster.

⁵ By default, a node is removed from the cluster when it misses 5 heartbeat messages (this value is specified by the *AliveMsgTolerance* registry key).

⁶ The *hosts* file can be found in %SYSTEMROOT%\system32\drivers\etc.

⁷ Class C and single affinity are identical in that they both direct requests from the same client to the same host within the cluster, however the latter affinity type is tailored for the intranet environment whereas Class C is optimal for clusters serving Internet clients.