Microsoft Windows Server 2003 TCP/IP Implementation Details

Microsoft Corporation
Published: June 2003
Updated: December 2007

Abstract

This white paper describes the implementation of the TCP/IP protocol stack in the Microsoft® Windows Server® 2003 family and is a supplement to the Windows Server 2003 Help and Support Center and Technical Reference documentation. This white paper contains an overview of TCP/IP in Windows Server 2003 features and capabilities, a discussion of protocol architecture, and detailed discussions of the core components, network application interfaces, and critical client components and services. The intended audience for this paper is network engineers and support professionals who are already familiar with TCP/IP. Except where noted, the TCP/IP implementation for Windows® XP is the same as that for Windows Server 2003.

Contents

Introduction

Microsoft has adopted TCP/IP as the strategic enterprise network transport for its platforms. In the early 1990s, Microsoft started an ambitious project to create a TCP/IP stack and services that would greatly improve the scalability of Microsoft networking. With the release of the Microsoft Windows NT® 3.5 operating system, Microsoft introduced a completely rewritten TCP/IP stack. This new stack was designed to incorporate many of the advances in performance and ease of administration that were developed over the past decade. The stack was a high-performance implementation of the industry-standard TCP/IP protocol. It has evolved with each version of Windows based on the Windows NT code base to include new features and services that enhance performance, security, and reliability.

The goals in designing the TCP/IP stack were to make it:

- Standards-compliant and interoperable
- Portable
- Scalable and fast
- Versatile
- Self-tuning and easy to administer

This paper examines the Windows Server 2003 TCP/IP protocol suite from the bottom up. Throughout the paper, network traces are used to illustrate key concepts. These traces were gathered and formatted using Microsoft Network Monitor 2.0, a software-based protocol tracing and analysis tool included in the Microsoft Systems Management Server product. Windows 2000 Server and Windows Server 2003 include a limited functionality version of Network Monitor. The primary difference between this version and the Systems Management Server version is that the limited version can only capture frames that would normally be seen by the computer that it is installed on, rather than all frames that pass over the network (which requires the network interface card to be in promiscuous mode). It also does not support connecting to remote Network Monitor Agents.

Capabilities and Functionality

Overview

Windows Server 2003 TCP/IP was designed to make it easy to integrate Microsoft systems into large-scale corporate, government, and public networks, and to provide the ability to operate over those networks in a secure manner. The Windows Server 2003 TCP/IP protocol is installed by default and, unlike previous versions of Windows, cannot be uninstalled. However, you can reset the TCP/IP configuration to a default state with the **netsh interface ip reset** command.

Support for Standard Features

Windows Server 2003 TCP/IP supports the following standard features:

- Ability to bind to multiple network adapters with different media types
- Logical and physical multihoming
- Internal IP routing capability
- Internet Group Management Protocol (IGMP) version 3 (IP multicasting)
- Duplicate IP address detection
- Multiple default gateways
- Dead gateway detection
- Automatic Path Maximum Transmission Unit (PMTU) discovery
- Internet Protocol security (IPsec)
- Quality of Service (QoS)
- ATM Services
- Virtual Private Networks (VPNs) with the Point-to-Point Tunneling Protocol (PPTP) and the Layer Two Tunneling Protocol with IPsec (L2TP/IPsec)

Performance Enhancements

In addition, Windows Server 2003 TCP/IP has the following performance enhancements:

- Protocol stack tuning, including increased default window sizes and new algorithms for high-delay and high-loss links, which increases throughput
- TCP-scalable window sizes (described in RFC 1323)
- Selective acknowledgments (SACK) (described in RFC 2018)
- TCP fast retransmit and fast recovery (described in RFC 2581)
- Round Trip Time (RTT) and Retransmission Timeout (RTO) calculation improvements
- Improved performance for management of large numbers of connections
- Hardware task offload mechanisms including checksum offload and large send offload (LSO)

Services Available

The Windows Server 2003 operating system provides the following TCP/IP-related services:

- Dynamic Host Configuration Protocol (DHCP) client and server and DHCP Relay Agent (with the Routing and Remote Access service)
- In the absence of a DHCP server, Automatic Private IP Addressing (APIPA) is used
- Windows Internet Name Service (WINS), a NetBIOS name client and server
- Domain Name System (DNS) client and server, including support for DNS dynamic updates
- Dial-up support using the Point-to-Point Protocol (client and server) and Serial Line Internet Protocol (client only)
- PPTP and L2TP/IPsec, used for remote access and site-to-site VPN connections
- TCP/IP network printing (client only with the Lpr.exe and Lpq.exe tools)
- SNMP agent
- NetBIOS interface
- Network Location Service
- Windows Sockets version 2 (Winsock2) interface
- Remote Procedure Call (RPC) support
- Network Dynamic Data Exchange (NetDDE)
- Computer browsing (My Network Places) across IP routers
- Reliable multicast with the Pragmatic General Multicast (PGM) protocol
- Basic TCP/IP connectivity utilities, including: finger, ftp, rcp, rexec, rsh, telnet, and tftp
- Server and client software for simple network protocols, including: Character Generator, Daytime, Discard, Echo, and Quote of the Day
- Routing Information Protocol (RIP) listener (for Windows XP Professional) and RIP and Open Shortest Path First (OSPF) (with the Routing and Remote Access service)
- Network Address Translator (NAT) capabilities using either the Internet Connection Sharing (ICS) or the NAT/Basic Firewall routing protocol component of the Routing and Remote Access service
- Stateful firewalling capabilities using either the Internet Connection Firewall (for Windows Server 2003 with no service packs installed), Windows Firewall (for Windows Server 2003 Service Pack 1), or the NAT/Basic Firewall routing protocol component of the Routing and Remote Access service
- Multicast forwarding and IGMP router and proxy capabilities with the Routing and Remote Access service
- TCP/IP management and diagnostic tools, including: arp, ipconfig, nbtstat, netsh, netstat, ping, pathping, route, nslookup, and tracert

New Features for Windows Server 2003 TCP/IP

The features and improvements of TCP/IP that are new for Windows Server 2003 include the following:

- Windows Server 2003, Windows XP with Service Pack 1, and Windows XP with Service Pack 2 now include a production-quality IPv6 protocol stack. For more information about IPv6, see Windows Server 2003 Help and Support Center or the Microsoft IPv6 Web site (http://www.microsoft.com/ipv6).
- Auto-negotiation of RFC 1323 options (window scaling and TCP timestamps).
- Default support of network interface cards (NICs) providing large send offload (LSO) and checksum offload.
- IGMP version 3.
- Reliable multicast with PGM.
- Alternate configuration.
- Automatic determination of the interface-related and default route metrics.

The table below lists features and the operating system versions that they are present in as a reference. Features are described in more detail throughout this paper.

Table 1. Feature Comparison Table for Windows TCP/IP Versions

Product	Windows 98	Windows 98 SE	Windows NT 4.0 SP5	Windows 2000	Windows Server 2003
Dead gateway detection	Y	Y	Y	Y	Y
Fast retransmit/recovery	Y	Y	Y	Y	Y
APIPA	Y	Y	N	Y	Y
Selective ACK (SACK)	Y	Y	N	Y	Y
Jumbo frame support	Y	Y	Y	Y	Y
Large windows	D	D	N	D	D
DNS dynamic update	N	N	N	Y	Y
Media sense	N	N	N	Y	Y
Wake on LAN	N	N	N	Y	Y
IP forwarding	N	D	D	D	D
NAT	N	D	N	D	D
Kerberos v5	N	N	N	Y	Y
IPsec	N	N	N	Y	Y
PPTP	Y	Y	Y	Y	Y
L2TP/IPsec	N	N	N	Y	Y
IP Helper API	Y	Y	Y	Y	Y
Winsock2 API	Y	Y	Y	Y	Y
GQoS API	Y	Y	N	Y	Y
IP Filtering API	N	N	N	Y	Y
Firewall hook	N	N	N	Y	Y

Packet scheduler	N	N	N	D	D
Network location	N	N	N	N	Y
ISSLOW	Y	Y	N	Y	Y
Personal firewall	N	N	N	N	D
Block source routing	N	Y	Y	Y	Y
ICMP Router Discovery	Y	Y	D	D	D
IPsec offload	N	N	N	Y	Y
IGMP v3	N	N	N	N	Y
Reliable multicast (PGM)	N	N	N	N	Y
Alternate configuration	N	N	N	N	Y
Auto-determination of routing metrics	N	N	N	N	Y
Checksum offload	N	N	N	N	Y
Large send offload	N	N	N	N	Y

N=No, Y=Yes, and D=Disabled by Default

Internet RFCs Supported by Windows Server 2003 TCP/IP

Requests for Comments (RFCs) are a constantly evolving series of reports, proposals for protocols, and protocol standards used by the Internet community. You can obtain RFCs from http://www.ietf.org/rfc.html.

Table 2. RFCs supported by Windows Server 2003 TCP/IP

RFC	Title
768	User Datagram Protocol (UDP)
783	Trivial File Transfer Protocol (TFTP)
791	Internet Protocol (IP)
792	Internet Control Message Protocol (ICMP)
793	Transmission Control Protocol (TCP)
816	Fault Isolation and Recovery
826	Address Resolution Protocol (ARP)
854	Telnet Protocol (TELNET)
862	Echo Protocol (ECHO)
863	Discard Protocol (DISCARD)
864	Character Generator Protocol (CHARGEN)
865	Quote of the Day Protocol (QUOTE)
867	Daytime Protocol (DAYTIME)
894	IP over Ethernet
919, 922	IP Broadcast Datagrams (broadcasting with subnets)
950	Internet Standard Subnetting Procedure
959	File Transfer Protocol (FTP)
1001, 1002	NetBIOS Service Protocols

1065, 1035, 1123, 1886	Domain Name System (DNS)
1042	A Standard for the Transmission of IP Datagrams over IEEE 802 Networks
1055	Transmission of IP over Serial Lines (IP-SLIP)
1112	Internet Group Management Protocol (IGMP)
1122, 1123	Host Requirements (communications and applications)
1144	Compressing TCP/IP Headers for Low-Speed Serial Links
1157	Simple Network Management Protocol (SNMP)
1179	Line Printer Daemon Protocol
1188	IP over FDDI
1191	Path MTU Discovery
1201	IP over ARCNET
1256	ICMP Router Discovery Messages
1323	TCP Extensions for High Performance
1332	PPP Internet Protocol Control Protocol (IPCP)
1518	Architecture for IP Address Allocation with CIDR
1519	Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy
1534	Interoperation Between DHCP and BOOTP
1542	Clarifications and Extensions for the Bootstrap Protocol
1552	PPP Internetwork Packet Exchange Control Protocol (IPXCP)
1661	The Point-to-Point Protocol (PPP)
1662	PPP in HDLC-like Framing
1748	IEEE 802.5 MIB using SMIv2
1749	IEEE 802.5 Station Source Routing MIB using SMIv2
1812	Requirements for IP Version 4 Routers
1828	IP Authentication using Keyed MD5
1829	ESP DES-CBC Transform
1851	ESP Triple DES-CBC Transform
1852	IP Authentication using Keyed SHA
1994	PPP Challenge Handshake Authentication Protocol (CHAP)
1995	Incremental Zone Transfer in DNS
1996	A Mechanism for Prompt DNS Notification of Zone Changes
2018	TCP Selective Acknowledgment Options
2085	HMAC-MD5 IP Authentication with Replay Prevention
2104	HMAC: Keyed Hashing for Message Authentication
2131	Dynamic Host Configuration Protocol
2136	Dynamic Updates in the Domain Name System (DNS UPDATE)
2181	Clarifications to the DNS Specification

2236	Internet Group Management Protocol, Version 2
2308	Negative Caching of DNS Queries (DNS NCACHE)
2401	Security Architecture for the Internet Protocol
2402	IP Authentication Header
2406	IP Encapsulating Security Payload (ESP)
2581	TCP Congestion Control
3208	PGM Reliable Transport Protocol Specification
3376	Internet Group Management Protocol, Version 3

New Features for TCP/IP in Windows Server 2003 Service Pack 1

The features and improvements of TCP/IP that are new for Windows Server 2003 Service Pack 1 include the following:

- Windows Firewall
- The Netstat –b option
- Netsh commands for Windows Sockets
- SYN attack protection is enabled by default
- SYN attack notification IP Helper APIs
- Registry value for ICMP host routes
- Smart TCP port allocation
- Registry value for multiple ARP replies

Windows Firewall

Windows Firewall replaces the Internet Connection Firewall provided with Windows Server 2003 with no service packs installed. Windows Firewall is a stateful firewall that drops unsolicited incoming traffic that does not correspond to either traffic sent in response to a request of the computer (solicited traffic) or unsolicited traffic that has been specified as allowed (excepted traffic). Windows Firewall provides a level of protection from malicious users and programs that rely on unsolicited incoming traffic to attack computers.

For more information about Windows Firewall in Windows Server 2003 Service Pack 1, see the Microsoft Windows Server 2003 Windows Firewall TechCenter.

The Netstat –b option

The Netstat tool displays a variety of information about active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, and IPv4 and IPv6 statistics. In Windows Server 2003 Service Pack 1, the Netstat tool supports a new **–b** option that displays the set of components that are listening on each open TCP and UDP port.

With Windows Server 2003 with no service packs installed, you can use the **–o** option to display the set of ports being listened on and the corresponding process ID (PID). You can then lookup the PID in the display of the **tasklist /svc** command to discover the name of the process that owns the port. However,

in some cases, there are multiple services within a single process and it is not possible to determine which service within the process owned the port.

With the **–b** option, Netstat displays the TCP or UDP port, the file names corresponding to the components of the service that owns the port, and the PID. From the file names and PID, you can determine which of the services in the display of the **tasklist /svc** command owns the port.

Netsh Commands for Windows Sockets

There are now Windows Sockets (Winsock) Netsh commands to view the set of installed Winsock Layered Service Providers (LSPs) (the **netsh winsock show catalog** command) and to reset the Winsock LSP catalog to a default configuration (the **netsh winsock reset catalog** command). The **netsh winsock reset catalog** command is useful for restoring the Winsock LSP catalog when it has been corrupted by programs or services that install LSPs. However, you must reinstall the programs or services that use LSPs.

SYN Attack Protection is Enabled by Default

A TCP Synchronize (SYN) attack is a denial-of-service attack that exploits the retransmission and time-out behavior of the Synchronize-Acknowledgement (SYN-ACK) segment during the TCP three-way handshake to create a large number of half-open TCP connections. Depending on the TCP/IP protocol implementation, a large number of half-open TCP connections could do any of the following:

- Use all available memory.
- Use all possible entries in the TCP Transmission Control Block (TCB), an internal table used to track TCP connections. Once the half-open connections use all the entries, further connection attempts are responded to with a TCP connection reset.
- Use all available half-open connections. Once all the half-open connections are used, further connection attempts are responded to with a TCP connection reset.

To create a large number of TCP half-open connections, attackers send a large number of SYN segments, each from a spoofed IP address and TCP port number. Each spoofed IP address and TCP port number are for a process that does not respond to the SYN-ACKs being sent by the attacked host. SYN attacks are typically used to render Internet servers inoperative.

To mitigate the impact on a host experiencing a SYN attack, TCP/IP minimizes the amount of resources devoted to incomplete TCP connections and reduces the amount of time before abandoning the half-open connection. When a SYN attack is detected, TCP/IP in Windows Server 2003 and Windows XP lowers the number of retransmissions of the SYN-ACK segment and does not allocate memory or table entry resources for the connection until the TCP three-way handshake has been completed.

You can control SYN attack protection through the SynAttackProtect registry entry at HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters (type REG_DWORD). You set SynAttackProtect to 0 to disable SYN attack protection and to 1 to enable it.

For TCP/IP in Windows XP (all versions) and Windows Server 2003 with no service packs installed, SynAttackProtect is set to 0 by default. For TCP/IP in Windows Server 2003 Service Pack 1, SynAttackProtect is set to 1 by default.

SYN Attack Notification IP Helper APIs

To allow an application to notify network administrators that a SYN attack is taking place, the IP Helper API supports new SYN attack notification APIs named NotifySecurityHealthChange and CancelSecurityHealthChangeNotify. For information about these new APIs, see the Microsoft Developer Network (MSDN).

Registry Value for ICMP Host Routes

TCP/IP for Windows Server 2003 SP1 supports a new registry value that restricts the number of host routes that can be added to the local IP route table by receiving ICMP Redirect messages. The new registry value is HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services \Tcpip\Parameters\ MaxICMPHostRoutes (REG_DWORD type). MaxICMPHostRoutes is set to 1000 by default. You should not change this value unless the computer needs to be able to add a large number of host routes by receiving ICMP Redirect messages.

The update to Windows Server 2003 SP1 available from Microsoft Knowledge Base article 898060 changes the default value of MaxICMPHostRoutes to 10000.

Smart TCP Port Allocation

When a TCP peer initiates a TCP connection termination and the connection termination completes, the TCP connection enters the TIME-WAIT state. Once the TIME-WAIT state is reached, TCP must wait twice the maximum segment lifetime (MSL) before a connection with the same set of socket addresses can be created. The set of socket addresses consist of the combination of the source and destination IP addresses and source and destination TCP ports. The MSL is the maximum amount of time a TCP segment can exist in an internetwork, and its recommended value is 120 seconds. This delay prevents a new connection's TCP segments that are using the same set of socket addresses from being confused with duplicated TCP segments of the old connection.

The TCP port for a connection in the TIME-WAIT state is considered an available port and can be assigned for use by an application. This can lead to the following situation:

1. An application requests any available TCP port.
2. TCP/IP assigns a TCP port to use for the application socket.
3. The application attempts to open a socket with a specific destination IP address.
4. The application establishes a TCP connection and sends data.
5. The application terminates the TCP connection.
6. TCP/IP places the application's TCP connection in the TIME-WAIT state until twice the MSL has passed.
7. The same application requests another available TCP port.
8. TCP/IP assigns a TCP port to use for the application socket. Because the port for the connection in the TIME-WAIT state is considered open, it can be chosen as the next port to assign to the requesting application.
9. Assuming that TCP/IP assigns the same TCP port number, the application attempts to open a socket with the same destination IP address.

10. Because the connection is using the same set of socket addresses as the connection in the TIME-WAIT state, TCP/IP indicates an error to the application.

You can mitigate this situation by setting the TcpTimedWaitDelay registry entry at HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters (REG_DWORD type) to a lower value. The value of TcpTimedWaitDelay determines the length of time that a connection stays in the TIME-WAIT state. However, lowering the value of TcpTimedWaitDelay is contrary to the original design of TCP and the MSL.

To prevent an application from creating a connection with the same set of socket addresses of a connection that is in a TIME-WAIT state, TCP/IP in Windows Server 2003 Service Pack 1 has implemented a smart TCP port allocation algorithm. When an application requests any available TCP port, TCP/IP first attempts to find an available port that does not correspond to a connection in the TIME-WAIT state. If a port cannot be found, then it picks any available port.

This new behavior makes it much more unlikely that an application will be assigned a TCP port that is in the TIME-WAIT state when connecting to the same destination. You no longer need to modify the TcpTimedWaitDelay registry entry.

Registry Value for Multiple ARP Replies

TCP/IP for Windows Server 2003 SP1 supports a new registry value that determines which MAC address is stored in the ARP cache when multiple ARP Reply messages are received. If there are multiple computers that are using the same IP address on a subnet, when a node sends an ARP Request frame for the IP address, it will receive multiple ARP replies. The new registry value HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\StrictARPUpdate (REG_DWORD type) allows you to specify whether TCP/IP in Windows Server 2003 SP1 will store the MAC address of the last ARP reply received (StrictARPUpdate=0 [default]) or the MAC address of the first ARP reply received (StrictARPUpdate=1).

Architectural Model

Overview

The Windows TCP/IP suite contains *core protocol elements*, *services*, and the *interfaces* between them. The Transport Driver Interface (TDI) and the Network Device Interface Specification (NDIS) are public, and their specifications are available from Microsoft in the Microsoft Windows Driver Kit (WDK), available at http://www.microsoft.com/whdc/devtools/wdk/default.mspx . In addition, there are a number of higher-level interfaces available to user-mode applications. The most commonly used are Windows Sockets, remote procedure call (RPC), and NetBIOS.

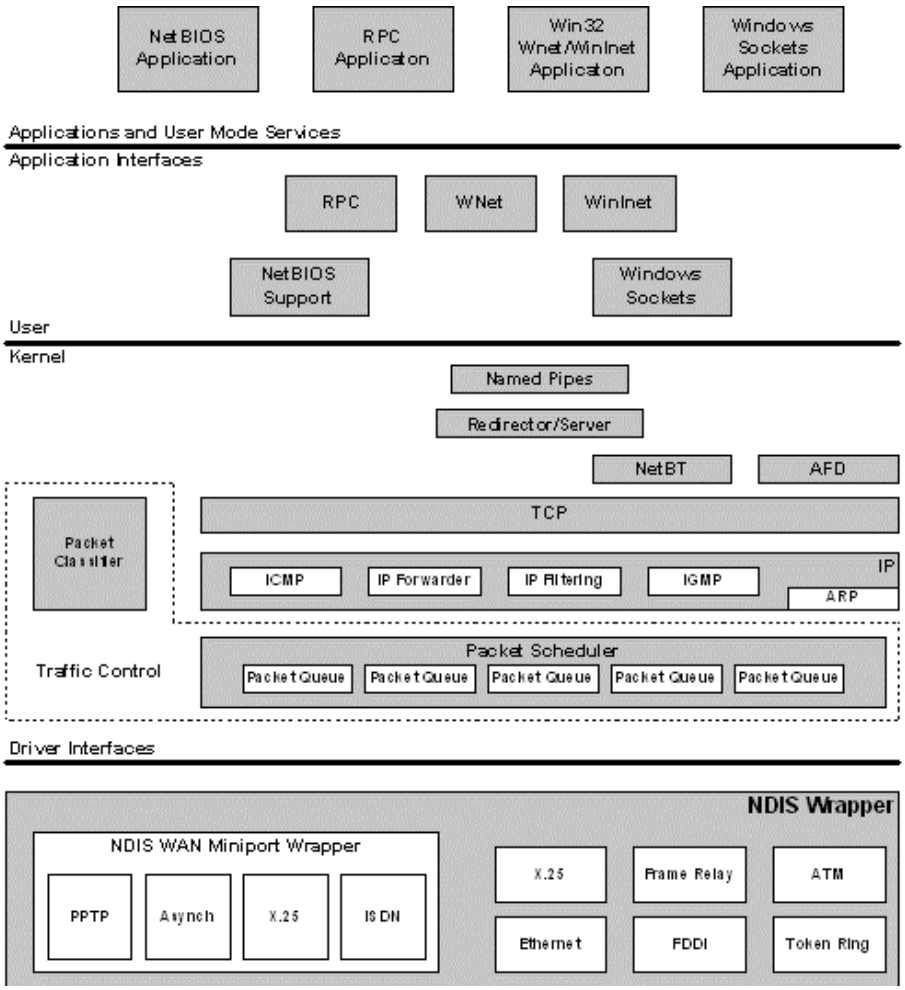

***Figure 1**. The Windows Server 2003 TCP/IP architectural model*

Note Figure 1 does not show the IPsec components.

Sending and Receiving IP Packets

When the source sends an IP packet, the following components analyze or change the packet in the following order:

1. Routing and Remote Access service IP packet filters
2. IPsec

When a router running Windows Server 2003 forwards an IP packet, the following components analyze or change the packet in the following order:

1. Internet Connection Sharing of the Network Connections folder or the NAT/Basic Firewall component of the Routing and Remote Access service
2. Routing and Remote Access service IP packet filters
3. IPsec

When an IP packet is received for forwarding, the following components analyze or change the packet in the following order:

1. IPsec
2. Routing and Remote Access service IP packet filters
3. Internet Connection Sharing of the Network Connections folder or the NAT/Basic Firewall component of the Routing and Remote Access service

When an IP packet is received that is destined for the local host, the following components analyze or change the packet in the following order:

1. IPsec
2. Routing and Remote Access service IP packet filters
3. Internet Connection Firewall (for Windows Server 2003 with no service packs installed), Windows Firewall (for Windows Server 2003 with Service Pack 1), or the NAT/Basic Firewall component of the Routing and Remote Access service
4. TCP/IP filters

This discussion only includes components that are provided with Windows Server 2003. This does not include any Windows Sockets layered service providers or NDIS intermediate miniport drivers. For additional information, see TCP/IP Packet Processing Paths.

Plug and Play

Windows Server 2003 includes support for Plug and Play. Plug and Play has the following capabilities and features:

- Automatic and dynamic recognition of installed hardware. This includes initial system installation, recognition of static hardware changes that may occur between boots, and response to run-time hardware events, such as dock or undock, and insertion or removal of cards.
- Streamlined hardware configuration in response to automatic and dynamic recognition of hardware, including dynamic hardware activation, resource arbitration, device driver loading, drive mounting, and so on.

- Support for particular buses and other hardware standards that facilitate automatic and dynamic recognition of hardware and streamlined hardware configuration, including Plug and Play ISA, PCI, PCMCIA, PC Card/CardBus, USB, and IEEE 1394. This includes promulgation of standards and advice about how hardware should behave.
- An orderly Plug and Play framework in which driver writers can operate. This includes infrastructure, such as device information (INF) interfaces, APIs, kernel-mode notifications, executive interfaces, and so on.
- Mechanisms that allow user-mode code and applications to learn of changes in the hardware environment so that they can take appropriate actions.

Plug and Play operation does not require Plug and Play hardware. To the degree possible, the first two bullets above apply to legacy hardware, as well as Plug and Play hardware. In some cases, orderly enumeration of legacy devices is not possible because the detection methods are destructive or inordinately time-consuming.

The primary impact that Plug and Play support has on protocol stacks is that network interfaces can come and go at any time. Windows Server 2003 TCP/IP and related components have been adapted to support Plug and Play.

The NDIS Interface and Below

Microsoft networking protocols use the Network Device Interface Specification (NDIS) to communicate with network card drivers. Much of the OSI model link layer functionality is implemented in the protocol stack. This makes development of network card drivers much simpler.

Network Driver Interface Specification (3.1 through 5.1)

NDIS 3.1 supports basic services that allow a protocol module to send raw packets over a network device and allow that same module to be notified of incoming packets received by a network device.

NDIS 4.0 added the following new features to NDIS 3.1:

- Out-of-band data support (required for Broadcast PC)
- WirelessWAN Media Extension
- High-speed packet send and receive (a significant performance win)
- Fast IrDA Media Extension
- Media Sense (required for the Designed for Windows logo in PC 97 and later Hardware Design Guide). The Windows Server 2003 TCP/IP stack utilizes media sense information, which is described in the “Automatic Client Configuration” section of this paper.
- All local packet filter (prevents Network Monitor from monopolizing the CPU)
- Numerous new NDIS system functions (required for miniport binary compatibility across Windows 98, Windows Millennium Edition, Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003)

NDIS 5.0 includes all functionality defined in NDIS 4.0, plus the following extensions:

- NDIS power management (required for Network Power Management and Network Wake-up)
- Plug and Play
- Support for Windows Management Instrumentation (WMI), which provides Web-based Enterprise Management (WBEM)–compatible instrumentation of NDIS miniports and their associated adapters
- Support for a single INF format across Windows operating systems. The INF format is based on the Windows 98 INF format.
- Deserialized miniport for improved performance
- Task offload mechanisms, such as TCP and UDP checksum and Fast Packet Forwarding
- TCP segmentation offload, also known as large-send offload (LSO)
- Broadcast Media Extension (needed for Broadcast Services for Windows)
- Connection-oriented NDIS (required to support Asynchronous Transfer Mode [ATM], Asymmetric Digital Subscriber Line [ADSL], and Windows Driver Model–Connection Streaming Architecture [WDM-CSA])
- Support for Quality of Service (QoS)

- Intermediate Driver Support (required for Broadcast PC, Virtual LANs, Packet Scheduling for QoS, and NDIS support of IEEE 1394 network devices)

Note You can disable a NIC's use of task offload capabilities on the **Advanced** tab of the properties of the NIC driver in Control Panel-Network or Device Manager.

NDIS 5.1 enhancements for Windows Server 2003 include:

- **Plug and Play and Power Event Notification** – Enables NIC miniport drivers to be notified of power or Plug and Play events. This results in cleaner system operation during these events.
- **Support for Send Cancellation** – Allows network protocols to avoid having to wait lengthy amounts of time for network packet send requests to complete.
- **Increased Statistics Capacity (64-bit statistic counters)** – This enhancement enables accurate network statistic displays, even on today's high-speed network media.
- **Performance Enhancements** – Several enhancements were made to speed up critical network data paths and avoid unnecessary packet copies.
- **Wake on LAN Change** – A change was made to Wake on LAN to allow you to limit wake up packets to just magic packets (instead of protocol registered packet patterns). This is now configurable on the **Power Management** tab from the properties of a NIC driver.
- **Miscellaneous Changes** – Several additional changes have been made to support common needs or requests from driver developers or to improve driver integrity.

Remote NDIS is also included as part of the Windows Server 2003 family. Remote NDIS enables the support of USB-attached network devices without the installation of third party drivers. Microsoft supplies the drivers required to communicate with the network devices. This results in easier installation and a lessened chance of system failure because of a poorly built or tested driver.

NDIS can power down network adapters when the system requests a power level change. Either the user or the system can initiate this request. For example, the user may want to put the computer in sleep mode, or the system may request a power level change based on keyboard or mouse inactivity. In addition, disconnecting the network cable can initiate a power-down request if the network interface card (NIC) supports this functionality. In this case, the system waits a configurable time period before powering down the NIC because the disconnect could be the result of temporary wiring changes on the network, rather than the disconnection of a cable from the network device itself.

NDIS power management policy is *no network activity*–based. This means that all overlying network components must agree to the request before the NIC can be powered down. If there are any active sessions or open files over the network, the power-down request can be refused by any or all of the components involved.

The computer can also be awakened from a lower power state, based on network events. A wakeup signal can be caused by:

- Detection of a change in the network link state (for example, cable reconnect)
- Receipt of a network wakeup frame
- Receipt of a Magic Packet.

At driver initialization, NDIS queries the capabilities of the miniport to determine if it supports such things as Magic Packet, pattern match, or link change wakeups, and to determine the lowest required power state for each wakeup method. The network protocols then query the miniport capabilities. At run time, the protocol sets the wakeup policy, using object identifiers (OIDs), such as Enable Wakeup, Set Packet Pattern, and Remove Packet Pattern.

Currently, TCP/IP is the only Microsoft protocol stack that supports network power management. It registers the following packet patterns at miniport initialization:

- Directed IP packet
- ARP broadcast for the station's IP address
- NetBIOS over TCP/IP broadcast for the station's assigned computer name

NDIS-compliant drivers are available for a wide variety of NICs from many vendors. The NDIS interface allows multiple protocol drivers of different types to bind to a single NIC driver and allows a single protocol to bind to multiple NIC drivers. The NDIS specification describes the multiplexing mechanism used to accomplish this. Bindings can be viewed or changed as advanced settings from the Network Connections folder.

Windows Server 2003 TCP/IP provides support for the following media types:

- Ethernet (using Ethernet II or IEEE 802.3 Sub-Network Access Protocol [SNAP] encapsulation)
- Fiber Distributed Data Interchange (FDDI)
- Token Ring (IEEE 802.5)
- IEEE 802.11 Wireless LAN
- ATM (using LAN emulation [LANE] and Classical IP [CLIP] over ATM)
- Attached Resource Computing Network (ARCnet)
- Dedicated wide area network (WAN) links such as Dataphone Digital Service (DDS) and T-carrier (Fractional T1, T1, T3, E1, and E3)
- Dial-up or permanent circuit-switched WAN services such as analog phone, Integrated Services Digital Network (ISDN), and Digital Subscriber Line (xDSL)
- Packet-switched WAN services such as X.25, Frame Relay, and ATM

Link Layer Functionality

Link layer functionality is divided between the network interface card/driver combination and the low-level protocol stack driver. The network card/driver combination filters are based on the destination media access control (MAC) address of each frame.

Normally, the hardware filters out all incoming frames except those containing one of the following destination addresses:

- The address of the adapter
- The all ones broadcast address (0xFF-FF-FF-FF-FF-FF)
- Multicast addresses that a protocol driver on this host has registered interest in, using the

NDISRequest() function

Because this first filtering decision is made by the hardware, the NIC discards any frames that do not meet the filter criteria without incurring any CPU processing. All frames (including broadcasts) that pass the hardware filter and Frame Check Sequence (FCS) validation are then passed up to the NIC driver through a hardware interrupt. Most NICs have the ability to be placed into a mode in which the NIC does not perform any address filtering on frames that appear on the media. Instead, it passes every frame upwards that passes the cyclic redundancy check (CRC). This feature is used by some protocol analysis software, such as Microsoft Network Monitor.

The NIC driver is software that runs on the computer, so any frames that make it this far require some CPU time to process. The NIC driver brings the frame into system memory from the interface card. Then the frame is indicated (passed up) to the appropriate bound transport driver(s). The NDIS 5.1 specification provides more detail on this process.

Frames are passed up to all bound transport drivers in the order that they are bound.

As a packet traverses a network or series of networks, the source MAC address is always that of the NIC that placed it on the media, and the destination MAC address is that of the NIC that is intended to pull it off the media. This means that, in a routed network, the source and destination MAC address changes with each hop through a network-layer device (router or Layer 3 switch).

Maximum Transmission Unit (MTU)

Each media type has a maximum frame size that cannot be exceeded. The link layer is responsible for discovering this MTU and reporting it to the protocols above. NDIS drivers may be queried for the local MTU by the protocol stack. Knowledge of the MTU for an interface is used by upper layer protocols, such as TCP, that optimize packet sizes for each media automatically. For details, see the discussion of TCP path Maximum Transmission Unit (PMTU) discovery in the “Transmission Control Protocol (TCP)” section of this paper.

If a NIC driver—such as an ATM driver—uses LAN emulation mode, it may report that it has an MTU that is higher than what is expected for that media type. For example, it may emulate Ethernet but report an MTU of 9180 bytes. Windows Server 2003 TCP/IP accepts and uses the MTU size reported by the adapter, even when it exceeds the normal MTU for a given media type.

Sometimes the MTU reported to the protocol stack may be less than what would be expected for a given media type. For instance, use of the 802.1p standard for QoS over Ethernet often reduces the MTU reported by 4 bytes due to larger link-layer headers (this is hardware dependent).

Core Protocol Stack Components and the TDI Interface

The core protocol stack components are those shown between the NDIS and TDI interfaces in Figure 1. They are implemented in the Windows Server 2003 Tcpip.sys driver. The TCP/IP stack is accessible through the TDI interface and the NDIS interface. The Winsock2 interface also provides some support for direct access to the protocol stack.

Address Resolution Protocol (ARP)

ARP performs IP address-to-MAC address resolution for outgoing packets. As each outgoing IP datagram is encapsulated in a frame, source and destination MAC addresses must be added. Determining the destination MAC address for each frame is the responsibility of ARP.

ARP compares the next-hop IP address on every outbound IP datagram to the ARP cache for the NIC over which the frame will be sent. If there is a matching entry, the MAC address is retrieved from the cache. If not, ARP broadcasts an ARP Request frame on the local subnet, requesting that the owner of the IP address in question reply with its MAC address. If the packet is going through a router, the next-hop address is that of a neighboring router and ARP resolves the MAC address for that next-hop router, rather than the final destination host. When an ARP reply is received, the ARP cache is updated with the new information, and it is used to address the packet at the link layer.

ARP Cache

You can use the ARP utility to view, add, or delete entries in the ARP cache. Examples are shown below. Entries added manually are static and are not automatically removed from the cache, whereas dynamic entries are removed from the cache (see the “ARP Cache Aging” section for more information).

The **arp** command can be used to view the ARP cache, as shown here:

```
C:\>arp -a

Interface: 157.60.137.88 --- 0x10003
  Internet Address      Physical Address      Type
  157.60.136.1          00-0a-42-b0-54-0a     dynamic
  157.60.137.0          00-b0-d0-e9-41-43     dynamic

Interface: 10.0.0.3 --- 0x10004
  Internet Address      Physical Address      Type
  10.0.0.1              08-00-2b-c4-25-b6     dynamic
```

The computer in this example is *multihomed*—has more than one NIC—so there is a separate ARP cache for each interface.

In the following example, the command **arp –s** is used to add a static entry to the ARP cache used by the second interface for the host whose IP address is 10.0.0.32 and whose NIC address is 00-60-8C-0E-6C-6A:

```
C:\>arp -s 10.0.0.32 00-60-8c-0e-6c-6a 10.0.0.3

C:\>arp -a

Interface: 157.60.137.88 --- 0x10003
  Internet Address      Physical Address      Type
  157.60.136.1          00-0a-42-b0-54-0a     dynamic
  157.60.137.0          00-b0-d0-e9-41-43     dynamic

Interface: 10.0.0.3 --- 0x10004
  Internet Address      Physical Address      Type
  10.0.0.1              08-00-2b-c4-25-b6     dynamic
  10.0.0.32             00-60-8c-0e-6c-6a     static
```

ARP Cache Aging

Windows Server 2003 adjusts the size of the ARP cache automatically to meet the needs of the system. If an entry is not used by any outgoing datagram for two minutes, the entry is removed from the ARP cache. Entries that are being referenced are removed from the ARP cache after ten minutes. Entries added manually are not removed from the cache automatically. The registry parameter *ArpCacheLife*, described in Appendix A, allows more administrative control over aging.

Use the command **arp –d** to delete entries from the cache, as shown below:

```
C:\>arp -d 10.0.0.32

C:\>arp -a

Interface: 157.60.137.88 --- 0x10003
  Internet Address      Physical Address      Type
  157.60.136.1          00-0a-42-b0-54-0a     dynamic
  157.60.137.0          00-b0-d0-e9-41-43     dynamic

Interface: 10.0.0.3 --- 0x10004
  Internet Address      Physical Address      Type
  10.0.0.1              08-00-2b-c4-25-b6     dynamic
```

ARP queues only one outbound IP datagram for a specified destination address while that IP address is being resolved to a MAC address. If a UDP-based application sends multiple IP datagrams to a single destination address without any pauses between them, some of the datagrams may be dropped if there is no ARP cache entry already present. An application can compensate for this by calling the Iphlpapi.dll routine SendArp() to establish an ARP cache entry, before sending the stream of packets. See INFO: IP Helper APIs Add Net Config and Stat Info to Win32 Apps or the MSDN for IP Helper API details.

Internet Protocol (IP)

IP is the mailroom of the TCP/IP stack, where packet sorting and delivery take place. At this layer, each incoming or outgoing packet is referred to as a datagram. Each IP datagram bears the source IP address of the sender and the destination IP address of the intended recipient. Unlike MAC addresses, the IP addresses in a datagram remain the same throughout a packet's journey across an internetwork, unless you are using source routing. IP layer functions are described below.

Routing

Routing is a primary function of IP. Datagrams are handed to IP from UDP and TCP above, and from the NIC(s) below. Each datagram is labeled with a source and destination IP address. IP examines the destination address on each datagram, compares it to a locally maintained route table, and decides what action to take. There are three possibilities for each datagram:

- It can be passed up to a protocol layer above IP on the local host.
- It can be forwarded using one of the locally attached NICs.
- It can be discarded.

The route table maintains four different types of routes:

1. Host route (a route to a single, specific destination IP address)
2. Subnet route (a route to a subnet)
3. Network route (a route to an entire network)
4. Default route (used when there is no other match)

To determine the single route to use to forward an IP datagram, IP uses the following process:

1. For each route in the route table, IP performs a bit-wise logical AND between the destination IP address and the netmask. IP compares the result with the network destination for a match. If they match, IP marks the route as one that matches the destination IP address.
2. From the list of matching routes, IP determines the route that has the most bits in the netmask. This is the route that matches the most bits to the destination IP address and is therefore the most specific route for the IP datagram. This is known as finding the longest or closest matching route.
3. If multiple closest matching routes are found, IP uses the route with the lowest metric. If multiple closest matching routes with the lowest metric are found, IP can choose to use any of those routes. For Windows Server 2003, IP uses the route corresponding to the adapter that is the highest in the binding order. You can view and modify the binding order from the **Adapters and Bindings** tab in the **Advanced Settings** dialog box for the Network Connections folder.

You can use the **route print** command to view the route table from the command prompt, as shown here:

```
C:\>route print

IPv4 Route Table
===========================================================================
Interface List
```

```
0x1 ........................... MS TCP Loopback interface
0x10002 ...00 53 45 00 00 00 ...... WAN (PPP/SLIP) Interface
0x10003 ...00 04 5a 56 10 06 ...... Linksys LNE100TX Fast Ethernet
                                    Adapter(LNE100TX v4)
===========================================================================
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0     157.60.136.1    157.60.137.88     20
     157.60.136.0    255.255.252.0    157.60.137.88    157.60.137.88     20
    157.60.137.88  255.255.255.255        127.0.0.1        127.0.0.1     20
   157.60.255.255  255.255.255.255    157.60.137.88    157.60.137.88     20
        127.0.0.0        255.0.0.0        127.0.0.1        127.0.0.1      1
        224.0.0.0        240.0.0.0    157.60.137.88    157.60.137.88     20
  255.255.255.255  255.255.255.255    157.60.137.88    157.60.137.88      1
Default Gateway:      157.60.136.1
===========================================================================
Persistent Routes:
  None
```

If the IPv6 protocol is installed, the display of the **route print** command also lists IPv6 routes.

The route table above is for a computer with the IP address of 157.60.137.88, the subnet mask of 255.255.252.0, and the default gateway of 157.60.136.1. It contains the following entries:

- The first entry, to destination 0.0.0.0, is the default route.
- The second entry is for the subnet 157.60.136.0, on which this computer resides.
- The third entry, to destination 157.60.137.88, is a host route for the local host. It specifies the loopback address, which makes sense because a datagram bound for the local host should be looped back internally.
- The fourth entry is for the all-subnets-directed broadcast address corresponding to the original Class B network ID 157.60.0.0.
- The fifth entry is for the loopback network, 127.0.0.0.
- The sixth entry is for IP multicasting, which is discussed later in this paper.
- The final entry is for the limited broadcast (all ones) address.

The *Default Gateway* is the currently active default gateway. This is useful to know when multiple default gateways are configured.

On this host, if a packet is sent to 157.60.138.49, the closest matching route is the local subnet route (157.60.136.0 with the mask of 255.255.252.0). The packet is sent via the local interface that is assigned the IP address 157.60.137.88. If a packet is sent to 10.200.1.1, the closest matching route is the default route. In this case, the packet is forwarded to the default gateway at 157.60.136.1.

The route table is maintained automatically in most cases. When a host initializes, entries for the local network(s), loopback, multicast, and configured default gateway are added. More routes may appear in

the table as the IP layer learns of them. For instance, the default gateway for a host may advise it of a better route to a specific address using ICMP redirect, which is explained later in this paper. Routes also may be added manually using the **route** command, or by a routing protocol. The **-p** (persistent) switch can be used with the route command to specify permanent routes. Persistent routes are stored in the registry under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\PersistentRoutes.

Windows Server 2003 TCP/IP introduces a new **Automatic metric** configuration option for interface-based and default gateway routing metrics. If selected for the interface, automatic metric configuration determines the metric for the routes associated with the interface configuration, such as subnet routes and host routes, based on the speed (bit rate) of the interface. The higher the speed, the lower the metric. For example, routes associated with 10 Mbps Ethernet interfaces have a metric of 30 and routes associated with 100 Mbps Ethernet interfaces have a metric of 20. If selected for the default gateway, automatic metric configuration determines the metric for the default route assigned to the interface, which is also based on the speed of the interface. Automatic metric configuration for both interface metrics and default routes are enabled by default and can be modified from the advanced configuration properties of the TCP/IP protocol for a connection in Network Connections. For more information, see An explanation of the Automatic Metric feature for Internet Protocol routes.

DHCP servers can also provide a base metric and a list of default gateways. If a DHCP server provides a base of 100, and a list of three default gateways, the gateways will be configured with metrics of 100, 101, and 102 respectively. A DHCP-provided base metric does not apply to statically configured default gateways.

By default, Windows-based systems do not behave as routers and do not forward IP datagrams between interfaces. However, the Routing and Remote Access service is included in Windows Server 2003 and can be enabled and configured to provide dynamic IP routing services using RIP and OSPF. Windows XP Professional includes support for silent RIP.

When running multiple logical subnets on the same physical network, the following command can be used to have IP treat all subnets as local (all destinations are on the local link):

```
route add 0.0.0.0 MASK 0.0.0.0 <my local ip address>
```

Thus, packets destined for non-local subnets are transmitted directly onto the local media instead of being sent to a router. In essence, the local interface card can be designated as the default gateway. This can be useful where several class C networks are used on one physical network with no router to the outside world, or in a proxy-ARP environment.

Duplicate IP Address Detection

Duplicate address detection is an important feature. When the stack is first initialized or when a new IP address is added, gratuitous ARP requests are broadcast for the IP addresses of the local host. The number of ARP requests to send is controlled by the *ArpRetryCount* registry parameter described in Appendix A, which defaults to 3. If another host replies to any of these ARP requests, the IP address is already in use. When this happens for an interface with a single manually-configured address, the Windows-based computer still boots; however, the interface containing the offending address is disabled, a system log entry is generated, and an error message is displayed. If the host that is defending the address is also a Windows-based computer, a system log entry is generated, and an error message is displayed on that computer. In order to update the ARP caches on other computers,

the offending computer re-broadcasts another ARP request, spoofing the MAC address of the defending computer, to restore the proper values in the ARP caches of the other computers.

A computer using a duplicate IP address can be started when it is not attached to the network, in which case no conflict would be detected. However, if it is then plugged into the network, the first time that it sends an ARP request for another IP address, any computer running a version of Windows with the Windows NT codebase with a conflicting address detects the conflict. The computer detecting the conflict displays an error message and logs a detailed event in the system log. A sample event log entry is shown below:

```
The system detected an address conflict for IP address 10.199.40.123 with the
system having network hardware address 00:DD:01:0F:7A:B5. Network operations on
this system may be disrupted as a result.
```

DHCP-enabled clients inform the DHCP server with a DHCPDecline message when an IP address conflict is detected and, instead of disabling the TCP/IP protocol, they request a new address from the DHCP server and request that the server mark the conflicting address as bad.

Multihoming

When a computer is configured with more than one IP address, it is referred to as a *multihomed* system. Multihoming is supported in three different ways:

- **Multiple IP addresses per NIC**
  - To add addresses for an interface, obtain properties of the **Internet Protocol (TCP/IP)** protocol in Network Connection, and then click **Advanced**. In the **Advanced Settings** dialog box, click **Add** on the **IP Settings** tab to add IP addresses.
  - NetBIOS over TCP/IP (NetBT) binds to only one IP address per interface card. When a NetBIOS name registration is sent out, only one IP address is registered per interface. This registration occurs over the IP address that is listed first on the **IP Settings** tab.
- **Multiple NICs per physical network.** There are no restrictions, other than hardware limitations on the number of NICs.
- **Multiple networks and media types.** There are no restrictions, other than hardware and media support. See the “The NDIS Interface and Below” section for supported media types.

When an IP datagram is sent from a multihomed host, it is passed to the interface with the best apparent route to the destination. Accordingly, the datagram may contain the source IP address of one interface in the multihomed host, yet be placed on the media by a different interface. The source MAC address on the frame is that of the interface that actually transmitted the frame to the media, and the source IP address is the one that the sending application sourced it from, not necessarily one of the IP addresses assigned to the sending interface.

When a computer is multihomed with NICs attached to disjoint networks (networks that are separate from and unaware of each other, such as a private network using private addressing and the Internet), routing problems may arise. It is often necessary to set up static routes to the private networks in this situation.

When configuring a computer to be multihomed on two disjoint networks, the best practice is to configure the default gateway on the interface connected to the largest and least-known network, in which the default route summarizes the most destinations. Then, either add static routes or use a

routing protocol to provide connectivity to the hosts on the smaller or better-known network. Avoid configuring a different default gateway on each side; this can result in unpredictable behavior and loss of connectivity. For more information, see Default Gateway Behavior for Windows TCP/IP.

Note There can only be one active default gateway for a computer at any moment in time.

More details on name registration, resolution, and choice of NIC on outbound datagrams with multihomed computers are provided in the "Transmission Control Protocol (TCP)," "NetBIOS over TCP/IP," and "Windows Sockets" sections of this paper.

Classless Inter-Domain Routing (CIDR)

CIDR, described in RFCs 1518 and 1519, removes the concept of address classes from the IP address assignment and management process. In place of predefined, well-known boundaries, CIDR allocates addresses defined by a network prefix, which makes more efficient use of available space. The network prefix defines the portion of the address that is fixed. For example, an assignment from an ISP to a corporate client might be expressed as 157.60.1.128/25. In this prefix, the first 25 bits are fixed and the last 7 bits can be used for address assignment. This would result in a 128-address block for local use, with the upper 25 bits being the network identifier part of the address. A legacy, class-full prefix would be expressed as *w*.0.0.0/8, *w.x*.0.0/16, or *w.x.y*.0/24. As these are reclaimed, they will be reallocated using classless CIDR techniques.

Given the installed base of classful systems, the initial implementation of CIDR was to summarize portions of the Class C address space. This process was called *supernetting*. Supernetting can be used to consolidate several network IDs into one prefix. For example, the network IDs 131.107.4.0, 131.107.5.0, 131.107.6.0, and 131.107.7.0 can be summarized with the network ID 131.107.4.0 with a subnet mask of 255.255.252.0 (131.107.4.0/22). For example:

```
NET     131.107.4.0   (1100 0111.1100 0111.0000 0100.0000 0000)
NET     131.107.5.0   (1100 0111.1100 0111.0000 0101.0000 0000)
NET     131.107.6.0   (1100 0111.1100 0111.0000 0110.0000 0000)
NET     131.107.7.0   (1100 0111.1100 0111.0000 0111.0000 0000)
MASK  255.255.252.0   (1111 1111.1111 1111.1111 1100.0000 0000)
```

All four of the network IDs share the same high-order 22 bits (underlined).

When routing decisions are made, only the bits covered by the subnet mask are used, thus making all these addresses appear to be part of the same network for routing purposes. Any routers in use must also support CIDR and may require special configuration.

Windows Server 2003 TCP/IP includes support for the all-0's and all-1's subnets as described in RFC 1878.

IP Multicasting

IP multicasting is used to provide efficient multicast services to clients that may not be located on the same network segment. Windows Sockets applications can join a multicast group to participate in a wide-area conference, for instance.

Windows Server 2003 TCP/IP is level 2-compliant with RFC 1112 (send and receive). IGMP is the protocol used to track multicast membership on a subnet, which is described later in this paper.

IP over ATM

Windows Server 2003 includes support for IP over ATM. RFC 1577 (and successors) define the basic operation of an IP over ATM network known as Classical IP over ATM (CLIP), which defines a Logical IP Subnet (LIS). A LIS is a set of IP hosts that can communicate directly with each other. Two hosts belonging to different LISs can communicate only through an IP router that is a member of both subnets. Windows Server 2003 also includes support for ATM LAN Emulation (LANE), which supports broadcasting.

ATM Address Resolution

Because an ATM network is non-broadcast, ARP broadcasts (as used by Ethernet or Token Ring) are not a suitable solution. Instead, a dedicated ARP server is used to provide IP-to-ATM address resolution.

One of the stations in a LIS is designated as an ARP server, and the ARP server software is loaded on it. Stations that use the services of the ARP server are referred to as *ARP clients*. All IP stations within a LIS are ARP clients. Each ARP client is configured with the ATM address of the ARP server. When an ARP client starts up, it makes an ATM connection to the ARP server, and sends a packet to the server that contains the client's IP and ATM addresses. The ARP server builds a table of IP-address-to-ATM-address mappings. When a client has an IP packet to be sent to another client (whose IP address is known but whose ATM address is unknown), it first queries the ARP server for the ATM address of the desired client. When it receives a reply that contains the desired ATM address, the client establishes a direct ATM connection to the target client and sends IP packets for that client on this connection.

The clients close any ATM connection, including the connection to the server, if the connections are inactive. All clients refresh their IP and ATM address information with the server periodically (the default is 15 minutes). The server purges an entry that is not refreshed after 20 minutes (by default). The ATM ARP client and ARP server both support a number of adjustable registry parameters, which are listed in Appendix A.

Internet Control Message Protocol (ICMP)

ICMP is a maintenance protocol specified in RFC 792 and is normally considered part of the Internet layer. ICMP messages are encapsulated within IP datagrams, so that they can be routed throughout an internetwork. Windows Server 2003 TCP/IP uses ICMP to:

- Report delivery problems encountered by routers or destination hosts.
- Build and maintain route tables.
- Perform router discovery.
- Assist in Path Maximum Transmission Unit (PMTU) discovery.
- Diagnose reachability problems (the ping, tracert, and pathping tools).
- Adjust flow control to prevent link or router saturation.

ICMP Router Discovery

Windows Server 2003 TCP/IP can perform router discovery as specified in RFC 1256. Router discovery provides an improved method of configuring and detecting default gateways. Instead of using manually- or DHCP-configured default gateways, hosts can dynamically discover routers on their subnet. If the primary router fails or the network administrators change router preferences, hosts can automatically switch to a backup router.

When a host that supports router discovery initializes, it joins the all-systems IP multicast group (224.0.0.1), and then listens for the router advertisements that routers send to that group. Hosts can also send router-solicitation messages to the all-routers IP multicast address (224.0.0.2) when an interface initializes to avoid any delay in being configured. Windows Server 2003 TCP/IP sends a maximum of three solicitations at intervals of approximately 600 milliseconds.

The use of router discovery is controlled by the *PerformRouterDiscovery* and *SolicitationAddressBCast* registry parameters, and it defaults to DHCP controlled in Windows Server 2003. Setting *SolicitationAddressBCast* to 1 causes router solicitations to be broadcast, instead of multicast, as described in RFC 1256.

Maintaining Route Tables

When a Windows-based computer is initialized, the route table normally contains only a few entries. One of those entries specifies a default gateway. Datagrams that have a destination IP address with no better match in the route table are sent to the default gateway. However, because routers share information about network topology, the default gateway may know a better route to a given address. When this is the case, then upon receiving a datagram that could take the better path, the router forwards the datagram normally. It then advises the sender of the better route, using an ICMP Redirect message. These messages typically specify redirection for a specific destination address. When a Windows-based computer receives an ICMP redirect, a validity check is performed to be sure that it came from the first-hop gateway in the current route, and that the gateway is on a directly connected network. If so, a host route with a 10-minute lifetime is added to the route table for that destination IP address. If the ICMP redirect did not come from the first-hop gateway in the current route, or if that gateway is not on a directly connected network, the ICMP redirect is ignored.

In Windows Server 2003 Service Pack 1, the new MaxICMPHostRoutes registry value defines the maximum number of host routes that can be added through the receipt of ICMP Redirect messages. For more information, see Appendix A.

Path Maximum Transmission Unit (PMTU) Discovery

TCP employs Path Maximum Transmission Unit (PMTU) discovery, as described in the “Transmission Control Protocol (TCP)” section of this paper. The mechanism relies on ICMP Destination Unreachable messages.

Use of ICMP to Diagnose Problems

The ping command-line utility is used to send ICMP echo requests to an IP address and wait for ICMP echo responses. Ping reports on the number of responses received and the time interval between sending the request and receiving the response. There are many different options that can be used with the ping utility.

Tracert is a route-tracing utility that can be very useful. Tracert works by sending ICMP echo requests to an IP address, while incrementing the Time to Live (TTL) field in the IP header, starting at 1, and analyzing the ICMP errors that are returned. Each succeeding echo request should get one hop further into the network before the TTL field reaches 0 and the router attempting to forward it returns an ICMP Time Exceeded-TTL Exceeded in Transit error message. Tracert prints out an ordered list of the routers in the path that returned these error messages, including the name and the IP address the nearside interface of each router. If the **-d** (do not do a DNS reverse query on each IP address) switch is used, only the IP address is reported. The example below illustrates using tracert to find the route from a computer dialed in over Point-to-Point Protocol (PPP) to an Internet service provider in Seattle to www.whitehouse.gov.

```
C:\>tracert www.whitehouse.gov

Tracing route to www.whitehouse.gov [128.102.252.1]
over a maximum of 30 hops:

 1  300 ms  281 ms  280 ms roto.seanet.com [199.181.164.100]
 2  300 ms  301 ms  310 ms sl-stk-1-S12-T1.sprintlink.net [144.228.192.65]
 3  300 ms  311 ms  320 ms sl-stk-5-F0/0.sprintlink.net [144.228.40.5]
 4  380 ms  311 ms  340 ms icm-fix-w-H2/0-T3.icp.net [144.228.10.22]
 5  310 ms  301 ms  320 ms arc-nas-gw.arc.nasa.gov [192.203.230.3]
 6  300 ms  321 ms  320 ms n254-ed-cisco7010.arc.nasa.gov [128.102.64.254]
 7  360 ms  361 ms  371 ms www.whitehouse.gov [128.102.252.1]
```

Pathping is a command-line utility that combines the functionality of ping and tracert as well as introducing some new features. Along with the tracing functionality of tracert, pathping will ping each hop along the route multiple times and display delay and packet loss information per hop, which can help you determine if there is a high-loss link or router in the path.

Internet Protocol Security (IPsec)

Windows Server 2003 includes support for Internet Protocol security (IPsec). IPsec features and implementation details are very complex and are described in a series of RFCs, Internet drafts, and in other Microsoft technical papers. For more information, see the IPsec Web page.

IPsec uses cryptography-based security to provide data integrity, data origin authentication, replay protection, data confidentiality (encryption), and limited traffic-flow confidentiality. Because IPsec is provided at the IP layer, its services are available to the upper-layer protocols in the stack and, transparently, to existing applications.

IPsec enables a system to select security protocols, decide which algorithm(s) to use for the service(s), and establish and maintain cryptographic keys for each security relationship. IPsec can protect paths between hosts, between security gateways, or between hosts and security gateways. The services available and required for traffic are configured using IPsec policy. IPsec policy may be configured locally on a computer or can be assigned through Group Policy mechanisms using the Active Directory™ directory service. When using Active Directory, hosts detect policy assignment at startup, retrieve the policy, and then periodically check for policy updates. The IPsec policy specifies how computers trust each other. IPsec can use certificates, Kerberos, or preshared keys as authentication methods. The easiest trust to use is the Windows Server 2003 domain trust based on Kerberos.

Each IP datagram processed at the IP layer is compared to a set of filters that are provided by the security policy, which is maintained by an administrator for a computer that belongs to a domain. IP can do one of three things with any datagram:

- Apply IPsec protections to it.
- Allow it to pass unmodified.
- Discard it.

An IPsec policy contains one or more rules, each of which contain a filter, a filter action, authentication methods, a tunnel setting, and a connection type. For example, two stand-alone computers can be configured to use IPsec between them and activate the secure server policy. If the two computers are not members of the same or a trusted domain, trust must be configured using a certificate or preshared key in a secure server mode by:

- Setting up a filter that specifies all traffic between the two hosts
- Choosing an authentication method
- Selecting a negotiation policy (secure server in this case, indicating that all traffic matching the filter(s) must use IPsec)
- Specifying a connection type (LAN, dial-up, or all)

Once the policy has been put in place, traffic that matches the filters uses the services provided by IPsec. When a host directs IP traffic to another host (including something as simple as ping traffic), a Security Association (SA) is established through an Internet Key Exchange (IKE) negotiation using UDP port 500, and then the traffic begins to flow. The following network trace illustrates setting up a TCP connection between two such IPsec-enabled hosts. The only parts of the IP datagram that are unencrypted and visible to Network Monitor after the SA is established are the MAC and IP headers:

```
Source IP      Dest IP       Prot  Description

192.168.10.47  10.197.14.91  UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 216 (0xD8)

10.197.14.91   192.168.10.47 UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 216 (0xD8)

192.168.10.47  10.197.14.91  UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 128 (0x80)

10.197.14.91   192.168.10.47 UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 128 (0x80)

192.168.10.47  10.197.14.91  UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 76 (0x4C)

10.197.14.91   192.168.10.47 UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 76 (0x4C)

192.168.10.47  10.197.14.91  UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 212 (0xD4)

10.197.14.91   192.168.10.47 UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 172 (0xAC)

192.168.10.47  10.197.14.91  UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 84 (0x54)

10.197.14.91   192.168.10.47 UDP   Src Port: ISAKMP, (500); Dst Port: ISAKMP (500);
Length = 92 (0x5C)

192.168.10.47  10.197.14.91  IP    ID = 0xC906; Proto = 0x32; Len: 96

10.197.14.91   192.168.10.47 IP    ID = 0xA202; Proto = 0x32; Len: 96
```

```
192.168.10.47  10.197.14.91  IP    ID = 0xCA06; Proto = 0x32; Len: 88
```

Opening one of the IP datagrams sent after the SA is established reveals very little of what is actually in the datagram (a TCP SYN, or connection request). The only clear parts of the packet are the Ethernet and IP headers. Even the TCP header is encrypted and cannot be parsed by Network Monitor if ESP is used.

```
Src IP         Dest IP        Protoc  Description
================================================

192.168.10.47  10.197.14.91   IP      ID = 0xC906; Proto = 0x32;
                                       Len: 96

+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
  IP: ID = 0xC906; Proto = 0x32; Len: 96
      IP: Version = 4 (0x4)
      IP: Header Length = 20 (0x14)
      IP: Precedence = Routine
      IP: Type of Service = Normal Service
      IP: Total Length = 96 (0x60)
      IP: Identification = 51462 (0xC906)
    + IP: Flags Summary = 2 (0x2)
      IP: Fragment Offset = 0 (0x0) bytes
      IP: Time to Live = 128 (0x80)
      IP: Protocol = 0x32
      IP: Checksum = 0xD55A
      IP: Source Address = 192.168.10.47
      IP: Destination Address = 10.197.14.91
      IP: Data: Number of data bytes remaining = 76 (0x004C)
```

Using a secure server policy also restricts all other types of traffic from reaching destinations that do not understand IPsec or are not part of the same trusted group. A secure initiator policy provides settings that apply best to servers; traffic security is attempted, but if the client does not understand IPsec, the negotiation falls back to sending clear text packets.

When IPsec is used to encrypt data, network performance generally drops, due to the processing overhead of encryption. One possible method for reducing the impact of this overhead is to offload the processing to a hardware device. Because NDIS 5.1 supports task offloading, it is feasible to include encryption hardware on NICs. NICs supporting IPsec hardware offload are available from several vendors.

IPsec promises to be popular for protecting both public network traffic and internal corporate or government traffic that requires confidentiality. One common deployment is to apply *secure server* IPsec policies only to specific servers that are used to store and/or serve confidential information.

Internet Group Management Protocol (IGMP)

Windows Server 2003 TCP/IP provides level 2 (full) support for IP multicasting and IGMP versions 1 through 3, as described in RFCs 1112, 2236, and 3376. The introduction to RFC 1112 provides a good overall summary of IP multicasting. The text reads:

> "IP multicasting is the transmission of an IP datagram to a *host group*—a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same 'best-effort' reliability as regular unicast IP datagrams; that is, the datagram is not guaranteed to arrive intact to all members of the destination group or in the same order relative to other datagrams.
>
> "The membership of a host group is dynamic; that is, hosts may join and leave groups at any time. There is no restriction on the location or number of members in a host group. A host may be a member of more than one group at a time. A host need not be a member of a group to send datagrams to it.
>
> "A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address—not the membership of the group—that is permanent; at any time a permanent group may have any number of members, even zero. Those IP multicast addresses that are not reserved for permanent groups are available for dynamic assignment to transient groups that exist only as long as they have members.
>
> "Internetwork forwarding of IP multicast datagrams is handled by multicast routers that may be co-resident with, or separate from, Internet gateways. A host transmits an IP multicast datagram as a local network multicast that reaches all immediately-neighboring members of the destination host group. If the datagram has an IP time-to-live greater than 1, the multicast router(s) attached to the local network take responsibility for forwarding it towards all other networks that have members of the destination group. On those other member networks that are reachable within the IP time-to-live, an attached multicast router completes delivery by transmitting the datagram as a local multicast."

IP/ARP Extensions for IP Multicasting

To support IP multicasting, an additional route is defined on the host. The route (added by default) specifies that if a datagram is being sent to a multicast host group, it should be sent to the IP address of the host group through the local NIC, and not forwarded to the default gateway. The following route (which you can discover using the **route print** command) illustrates this:

```
Network Address  Netmask    Gateway Address  Interface     Metric
224.0.0.0        240.0.0.0  157.60.137.88    157.60.137.88     20
```

IP multicast addresses are easily identified, as they are from the class D range, 224.0.0.0 to 239.255.255.255 (224.0.0.0/4). These IP addresses all have 1110 as their high-order bits.

To send a packet to a host group, using the local interface, the IP address must be resolved to a media access control address. As stated in RFC 1112:

> "An IP host group address is mapped to an Ethernet multicast address by placing the low-order 23 bits of the IP address into the low-order 23 bits of the Ethernet multicast address 01-00-5E-00-00-00 (hex). Because there are 28 significant bits in an IP host group address, more than one host group address may map to the same Ethernet multicast address."

For example, a datagram addressed to the multicast address 225.0.0.5 would be sent to the (Ethernet) MAC address 01-00-5E-00-00-05. This MAC address is formed by the junction of 01-00-5E and the 23 low-order bits of 225.0.0.5 (00-00-05).

Because more than one host group address can map to the same Ethernet multicast address, the interface may indicate hand-up multicasts for a host group for which no local applications have a registered interest. These extra multicasts are discarded by the TCP/IP protocol.

Multicast Extensions to Windows Sockets

Internet Protocol multicasting is currently supported only on AF_INET sockets of type SOCK_DGRAM and SOCK_RAW. By default, IP multicast datagrams are sent with a Time to Live (TTL) of 1. Applications can use the setsockopt() function to specify a TTL. By convention, multicast routers use TTL thresholds to determine how far to forward datagrams. These TTL thresholds are defined as follows:

- Multicast datagrams with initial TTL 0 are restricted to the same host.
- Multicast datagrams with initial TTL 1 are restricted to the same subnet.
- Multicast datagrams with initial TTL 32 are restricted to the same site.
- Multicast datagrams with initial TTL 64 are restricted to the same region.
- Multicast datagrams with initial TTL 128 are restricted to the same continent.
- Multicast datagrams with initial TTL 255 are unrestricted in scope.

Use of Multicast and IGMP by Windows Components

Some Windows Server 2003 components use IP multicast traffic. For example, router discovery uses multicasts, by default. WINS servers use multicasting when attempting to locate replication partners. The Windows Server 2003 Routing and Remote Access service supports multicast forwarding and the configuration of interfaces to operate in IGMP router or IGMP proxy mode.

Transmission Control Protocol (TCP)

TCP provides a connection-based, reliable byte-stream service to applications. Microsoft networking relies upon TCP for logon, file and print sharing, replication of information between domain controllers, transfer of browse lists, and other common functions. It can only be used for one-to-one communications.

TCP uses a checksum on both the TCP header and payload of each segment to reduce the chance that network corruption will go undetected. NDIS 5.1 provides support for task offloading, and Windows Server 2003 TCP/IP takes advantage of this by allowing the NIC to perform the TCP checksum calculations if the NIC driver offers support for this function. Offloading the checksum calculations to hardware can result in performance improvements in very high-throughput environments.

Windows Server 2003 TCP/IP has also been strengthened against a variety of attacks that were published over the past couple of years and has been subject to an internal security review intended to reduce susceptibility to future attacks. For instance, the initial sequence number (ISN) algorithm has been modified so that ISNs increase in random increments, using an RC4-based random number generator initialized with a 2048-bit random key upon system startup.

TCP Receive Window Size Calculation and Window Scaling (RFC 1323)

The TCP receive window size is the amount of receive data (in bytes) that can be buffered at one time on a connection. The sending host can send only that amount of data before waiting for an acknowledgment and window update from the receiving host. The Windows Server 2003 TCP/IP stack was designed to tune itself in most environments and uses larger default window sizes than earlier versions. Instead of using a hard-coded default receive window size, TCP adjusts to even increments of the maximum segment size (MSS) negotiated during connection setup. Matching the receive window to even increments of the MSS increases the percentage of full-sized TCP segments used during bulk data transmission.

The default advertised TCP receive-window size in Windows Server 2003 depends on the following, in order of precedence:

1. The SO_RCVBUF WinSock option for the connection.
2. The per-interface *TcpWindowSize* registry parameter.
3. The global *TcpWindowSize* registry parameter.
4. Autodetermination based on the NDIS-reported bit rate of the media. stack also tunes itself based on the auto-negotiated media speed reported by the network adapter:
    - Below 1 Mbps: 8 KB
    - From 1 Mbps – below 100 Mbps: 17 KB
    - 100 Mbps or above: 64 KB

For 10 Mbps Ethernet, the window is normally set to 17,520 bytes (17 KB rounded up to twelve 1460-byte segments.) For 100 Mbps Ethernet, the window is normally set to [0]64 KB.

There are two methods for setting the receive window size to specific values:

- The *TcpWindowSize* registry parameter (see Appendix A)
- The *setsockopt*() Windows Sockets function (on a per-socket basis)

To improve performance on high-bandwidth, high-delay networks, Windows Server 2003 TCP/IP supports scalable windows support as described in RFC 1323. This RFC details a method for supporting scalable windows by allowing TCP to negotiate a scaling factor for the window size at connection establishment. This allows for an actual receive window of up to 1 gigabyte (GB). Section 2.2 of RFC 1323 provides a good description:

> "The three-byte Window Scale option may be sent in a SYN segment by a TCP. It has two purposes: 1. indicate that the TCP is prepared to do both send and receive window scaling, and 2. communicate a scale factor to be applied to its receive window. Thus, a TCP that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is limited to a power of two and encoded logarithmically, so it may be implemented by binary shift operations.
>
> "This option is an offer, not a promise; both sides must send Window Scale options in their SYN segments to enable window scaling in either direction. If window scaling is enabled, then the TCP that sent this option will right-shift its true receive-window values by 'shift.cnt' bits for

transmission in SEG.WND. The value *shift.cnt* may be zero (offering to scale, while applying a scale factor of 1 to the receive window).

"This option may be sent in an initial <SYN> segment (in other words, a segment with the SYN bit on and the ACK bit off). It may also be sent in a <SYN,ACK> segment, but only if a Window Scale option was received in the initial <SYN> segment. A Window Scale option in a segment without a SYN bit should be ignored.

"The Window field in a SYN (in other words, a <SYN> or <SYN,ACK>) segment itself is never scaled."

When you read network traces of a connection that was established by two computers that support scalable windows, keep in mind that the window sizes advertised in the TCP headers must be scaled by the negotiated scale factor. The scale factor can be observed in the connection establishment (three-way handshake) packets, as illustrated in the following Network Monitor capture:

```
*************************************************************
Src Addr   Dst Addr  Protocol  Description
10.0.0.1   10.0.0.9  TCP       ....S., len:0, seq:725163-725163,
                               ack:0, win:65535, src:1217 dst:139


+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet
                                             Protocol
+ IP: ID = 0xB908; Proto = TCP; Len: 64
  TCP: ....S., len:0, seq:725163-725163, ack:0, win:65535,
               src:1217  dst:139 (NBT Session)
      TCP: Source Port = 0x04C1
      TCP: Destination Port = NETBIOS Session Service
      TCP: Sequence Number = 725163 (0xB10AB)
      TCP: Acknowledgement Number = 0 (0x0)
      TCP: Data Offset = 44 (0x2C)
      TCP: Reserved = 0 (0x0000)
    + TCP: Flags = 0x02 : ....S.
      TCP: Window = 65535 (0xFFFF)
      TCP: Checksum = 0x8565
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
        + TCP: Maximum Segment Size Option
          TCP: Option Nop = 1 (0x1)
          TCP: Window Scale Option
              TCP: Option Type = Window Scale
              TCP: Option Length = 3 (0x3)
              TCP: Window Scale = 5 (0x5)
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
        + TCP: Timestamps Option
          TCP: Option Nop = 1 (0x1)
```

```
          TCP: Option Nop = 1 (0x1)
        + TCP: SACK Permitted Option
************************************************************
```

The computer sending the packet above is offering the Window Scale option, with a scaling factor of 5. If the target computer responds, accepting the Window Scale option in the SYN-ACK, then it is understood that any TCP window advertised by this computer needs to be left-shifted 5 bits from this point onward (the SYN itself is not scaled). For example, if the computer advertised a 32 KB window in its first send of data, this value would need to be left-shifted (shifting in 0's from the right) 5 bits as shown below:

```
32 KB = 0x7fff =    111 1111 1111 1111
Left-shift 5 bits = 1111 1111 1111 1110 0000 = 0xfffe0 (1,048,544 bytes)
```

As a check, left-shifting a number 5 bits is equivalent to multiplying it by 2^5, or 32. Performing the calculation in decimal, we get 32767 * 32 = 1,048,544.

The scale factor is not necessarily symmetrical, so it may be different for each direction of data flow.

TCP window scaling is negotiated on-demand in Windows Server 2003, based on the value set for the SO_RCVBUF WinSock option when a connection is initiated. Additionally, the Window Scale option is used by default on a connection if the received SYN segment for a connection initiated by a TCP peer contains the Window Scale option. You can modify this default behavior with the *Tcp1323Opts* registry parameter (see Appendix A).

Delayed Acknowledgments

As specified in RFC 1122, TCP uses delayed acknowledgments (ACKs) to reduce the number of packets sent on the media. The Windows Server 2003 TCP/IP stack takes a common approach to implementing delayed ACKs. As data is received by TCP on a connection, it only sends an acknowledgment back if one of the following conditions is met:

- No ACK was sent for the previous segment received.
- A segment is received, but no other segment arrives within 200 milliseconds for that connection.

In summary, normally an ACK is sent for every other TCP segment received on a connection, unless the delayed ACK timer (200 milliseconds) expires. The delayed ACK timer can be adjusted through the *TcpDelAckTicks* registry parameter.

TCP Selective Acknowledgment (RFC 2018)

Windows Server 2003 TCP/IP includes support for an important performance feature known as Selective Acknowledgement (SACK). SACK is especially important for connections using large TCP window sizes. Prior to SACK, a receiver could only acknowledge the latest sequence number of contiguous data that had been received, or the left edge of the receive window. When SACK is enabled, the receiver continues to use the ACK number to acknowledge the left edge of the receive window, but it can also acknowledge other non-contiguous blocks of received data individually. SACK uses two different TCP header options:

- The Sack-Permitted option is used to indicate that the sender can receive and interpret the SACK option. This option is sent in TCP segments in which the SYN flag is set.

- The SACK option is used to convey extended acknowledgment information from the receiver to the sender over an established TCP connection. Within the SACK option are up to four pairs of TCP sequence numbers acknowledging up to four non-continuous blocks. Each TCP sequence pair contains the left edge (the sequence number of the first byte in the block) and the right edge (the sequence number of the byte immediately following the last byte in the block) of the block being acknowledged.

When SACK is enabled (the default), a packet or series of packets can be dropped, and the receiver can inform the sender of exactly which data has been received, and where the holes in the data are. The sender can then selectively retransmit the missing data without needing to retransmit blocks of data that have already been received successfully. SACK is controlled by the *SackOpts* registry parameter and is enabled by default. The Network Monitor capture below illustrates a host acknowledging all data up to sequence number 54857340, plus the data from sequence number 54858789-54861684.

```
+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet
                                             Protocol
+ IP: ID = 0x1A0D; Proto = TCP; Len: 64
  TCP: .A...., len:0, seq:925104-925104, ack:54857341, win:32722,
               src:1242  dst:139
      TCP: Source Port = 0x04DA
      TCP: Destination Port = NETBIOS Session Service
      TCP: Sequence Number = 925104 (0xE1DB0)
      TCP: Acknowledgement Number = 54857341 (0x3450E7D)
      TCP: Data Offset = 44 (0x2C)
      TCP: Reserved = 0 (0x0000)
    + TCP: Flags = 0x10 : .A....
      TCP: Window = 32722 (0x7FD2)
      TCP: Checksum = 0x4A72
      TCP: Urgent Pointer = 0 (0x0)
      TCP: Options
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
        + TCP: Timestamps Option
          TCP: Option Nop = 1 (0x1)
          TCP: Option Nop = 1 (0x1)
          TCP: SACK Option
              TCP: Option Type = 0x05
              TCP: Option Length = 10 (0xA)
              TCP: Left Edge of Block  = 54858789 (0x3451425)
              TCP: Right Edge of Block = 54861685 (0x3451F75)
```

TCP Timestamps (RFC 1323)

Windows Server 2003 TCP/IP supports TCP time stamps, as described in RFC 1323. Like SACK, time stamps are important for connections using large window sizes. Time stamps were conceived to assist TCP in accurately measuring round-trip time (RTT) to adjust retransmission time-outs.

The Timestamps option carries two four-byte time stamp fields. The time-stamp value field (TSval) contains the current value of the time-stamp clock of the TCP sending the option. The Timestamp Echo Reply field (TSecr) is only valid if the ACK bit is set in the TCP header; if it is valid, it echoes a timestamp value that was sent by the remote TCP peer in the TSval field of a Timestamps option. When TSecr is not valid, its value must be zero. The TSecr value will generally be from the most recent Timestamp option that was received; however, there are exceptions that are explained below.

A TCP peer may send the Timestamps option (TSopt) in an initial SYN segment (i.e., segment containing a SYN bit and no ACK bit), and may send a TSopt in other segments only if it received a TSopt in the initial SYN segment for the connection.

The Timestamps option field can be viewed in a Network Monitor capture by expanding the TCP options field, as shown below:

```
TCP: Timestamps Option
    TCP: Option Type = Timestamps
    TCP: Option Length = 10 (0xA)
    TCP: Timestamp = 2525186 (0x268802)
    TCP: Reply Timestamp = 1823192 (0x1BD1D8)
```

By default, the Timestamp option is only used on a connection if the received SYN segment for a connection initiated by a TCP peer contains the Timestamp option. This default behavior can be modified with the *Tcp1323Opts* registry parameter (see Appendix A).

Path Maximum Transmission Unit (PMTU) Discovery

PMTU discovery is described in RFC 1191. When a connection is established, the two hosts involved exchange their TCP maximum segment size (MSS) values. The smaller of the two MSS values is used for the connection. Historically, the MSS for a host has been the MTU at the link layer minus 40 bytes for the IP and TCP headers. However, support for additional TCP options, such as time stamps, has increased the typical TCP+IP header to 52 or more bytes. Figure 2 shows the difference between MTU and MSS.

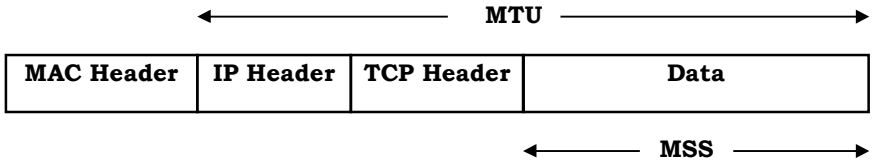

Figure 2. MTU versus MSS

When TCP segments are destined to a non-local network, the Don't Fragment (DF) flag is set in the IP header. Any link along the path can have an MTU that differs from that of the two hosts. If a link has an MTU that is too small for the IP datagram being routed, the router attempts to fragment the datagram. However, the Don't Fragment bit is set in the IP header. At this point, the router should inform the sending host that the datagram couldn't be forwarded further without fragmentation. This is done with

an ICMP Destination Unreachable-Fragmentation Needed and DF Set message. Most routers also specify the MTU for the next hop by putting the value for it in the low-order 16 bits of the ICMP header field that is unused in RFC 792. See RFC 1191, section 4, for the format of this message. Upon receiving this ICMP error message, TCP adjusts its MSS for the connection to the specified MTU minus the TCP and IP header size so that any further packets sent on the connection are no larger than the maximum size that can traverse the path without fragmentation.

Note The minimum MTU permitted is 88 bytes, and Windows Server 2003 TCP/IP enforces this limit.

Some noncompliant routers may silently drop IP datagrams that cannot be fragmented or may not correctly report their next-hop MTU. If this occurs, it may be necessary to make a configuration change to the PMTU detection algorithm. There are two registry changes that can be made to Windows Server 2003 TCP/IP stack to work around these problematic devices:

- *EnablePMTUBHDetect*—Adjusts the PMTU discovery algorithm to attempt to detect PMTU black hole routers. PMTU black hole detection is disabled by default.
- *EnablePMTUDiscovery*—Completely enables or disables the PMTU discovery mechanism. When PMTU discovery is disabled, an MSS of 536 bytes is used for all non-local destination addresses. PMTU discovery is enabled by default (this is the recommended setting).

These registry entries are described in more detail in Appendix A.

The PMTU between two hosts can be discovered manually using the ping tool with the **-f** (don't fragment) switch, as follows:

ping -f -n *<number of pings>* **-l** *<size>* *<destination ip address>*

As shown in the example below, the *size* parameter can be varied until the MTU is found. The *size* parameter used by ping is the size of the data buffer to send, not including headers. The ICMP header consumes 8 bytes, and the IP header is normally 20 bytes. In the case below (Ethernet), the link layer MTU is the maximum-sized ping buffer plus 28, or 1500 bytes:

```
C:\>ping -f -n 1 -l 1472 10.99.99.10
Pinging 10.99.99.10 with 1472 bytes of data:
Reply from 10.99.99.10: bytes=1472 time<10ms TTL=128
Ping statistics for 10.99.99.10:
    Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms

C:\>ping -f -n 1 -l 1473 10.99.99.10
Pinging 10.99.99.10 with 1473 bytes of data:
Packet needs to be fragmented but DF set.
Ping statistics for 10.99.99.10:
    Packets: Sent = 1, Received = 0, Lost = 1 (100% loss),
Approximate round trip times in milliseconds:
    Minimum = 0ms, Maximum =  0ms, Average =  0ms
```

In the example shown above, the IP layer returned an ICMP error message that ping interpreted. If the router had been a PMTU black hole router, ping would simply not be answered once its size exceeded the MTU that the router could handle. Ping can be used in this manner to detect such a router.

A sample ICMP Destination Unreachable-Fragmentation Needed and DF Set message is shown here:

```
*************************************************************
Src Addr       Dst Addr     Protocol  Description
10.99.99.10    10.99.99.9   ICMP      Destination Unreachable:
                                      10.99.99.10   See frame 3
+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP:  DOD Internet Protocol
+ IP: ID = 0x4401; Proto = ICMP; Len: 56
  ICMP: Destination Unreachable: 10.99.99.10   See frame 3
       ICMP: Packet Type = Destination Unreachable
       ICMP: Unreachable Code = Fragmentation Needed, DF Flag Set
       ICMP: Checksum = 0xA05B
       ICMP: Next Hop MTU = 576 (0x240)
       ICMP: Data: Number of data bytes remaining = 28 (0x001C)
       ICMP: Description of original IP frame
```

This error was generated by using **ping -f –n 1 -l 1000** on an Ethernet-based host to send a large datagram across a router interface that only supports an MTU of 576 bytes. When the router tried to place the large frame onto the network with the smaller MTU, it found that fragmentation was not allowed. Therefore, it returned the error message indicating that the largest datagram that could be forwarded is 576 bytes.

Dead Gateway Detection

Dead gateway detection is used to allow TCP to detect failure of the default gateway and to adjust the IP route table to use another default gateway. Windows Server 2003 TCP/IP uses the triggered reselection method described in RFC 816, with slight modifications based upon customer experience and feedback.

When a TCP connection routed through the default gateway attempts to send a TCP packet to the destination a number of times (equal to one-half of the registry value *TcpMaxDataRetransmissions*) without receiving a response, the algorithm changes the Route Cache Entry (RCE) for that remote IP address to use the next default gateway in the list as its next-hop address. When 25 percent of the TCP connections have moved to the next default gateway, the algorithm advises IP to change the computer's default gateway to the one that the connections are now using.

For example, assume that there are currently TCP connections to 11 different IP addresses that are being routed through the default gateway. Now assume that the default gateway fails, that there is a second default gateway configured, and that the value for *TcpMaxDataRetransmissions* is at the default of 5.

When the first TCP connection tries to send data, it does not receive any acknowledgments. After the third retransmission, the RCE for that remote IP address is switched to the next default gateway in the list. At this point, any TCP connections to that one remote IP address have switched over, but the remaining connections still try to use the original default gateway.

When the second TCP connection tries to send data, the same thing happens. Now, two of the 11 RCEs point to the new gateway.

When the third TCP connection tries to send data, after the third retransmission, three of 11 RCEs have been switched to the second default gateway. Because, at this point, over 25 percent of the RCEs have been moved, the default gateway for the whole computer is moved to the new one.

That default gateway remains the primary one for the computer until it experiences problems (causing the dead gateway algorithm to try the next one in the list again) or until the computer is restarted.

When the search reaches the last default gateway, it returns to the beginning of the list.

TCP Retransmission Behavior

TCP starts a retransmission timer when each outbound segment is handed down to IP. If no acknowledgment has been received for the data in a given segment before the timer expires, the segment is retransmitted. For new connection requests, the retransmission timer is initialized to 3 seconds (controllable using the *TcpInitialRtt* per-adapter registry parameter), and the request (SYN) is resent up to the value specified in *TcpMaxConnectRetransmissions* (the default for Windows Server 2003 is 2 times). On existing connections, the number of retransmissions is controlled by the *TcpMaxDataRetransmissions* registry parameter (5 by default). The retransmission time-out is adjusted on the fly to match the characteristics of the connection, using Smoothed Round Trip Time (SRTT) calculations as described in Van Jacobson's paper named "Congestion Avoidance and Control." The timer for a given segment is doubled after each retransmission of that segment. Using this algorithm, TCP tunes itself to the normal delay of a connection. TCP connections over high-delay links take much longer to time-out than those over low-delay links. Adding 1 to the registry parameter *TcpMaxDataRetransmissions* or *TcpMaxConnectRetransmissions* approximately doubles the total retransmission time-out period**.** If it is necessary to configure longer time-outs, these parameters should be increased very gradually.

The following Network Monitor capture shows the retransmission algorithm for two hosts that are connected over Ethernet on the same subnet. An FTP file transfer was in progress when the receiving host was disconnected from the network. Because the SRTT for this connection was very small, the first retransmission was sent after about one-half second. The timer was then doubled for each of the retransmissions that followed. After the fifth retransmission, the timer was once again doubled. If no acknowledgment was received before it expired, the connection was aborted.

```
Delta  Source IP    Dest IP      Pro Flags Description
-------------------------------------------------------
0.000  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
0.521  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
1.001  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
2.003  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
4.007  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
8.130  10.57.10.32  10.57.9.138  TCP .A.., len: 1460, seq: 8043781, ack: 8153124, win:
8760
```

There are some circumstances under which TCP retransmits data prior to the time that the retransmission timer expires. The most common of these occurs due to a feature known as *fast retransmit*. When a receiver that supports fast retransmit receives data with a sequence number beyond the current expected one, it assumes that some data was dropped. To help make the sender aware of this event, the receiver immediately sends an ACK, with the ACK number set to the sequence number that it was expecting. It continues to do this for each additional TCP segment that arrives containing data subsequent to the missing data in the incoming stream. When the sender starts to receive a stream of ACKs that are acknowledging the same sequence number and that sequence number is earlier than the current sequence number being sent, it can infer that a segment (or more) must have been dropped. Senders that support the fast retransmit algorithm immediately resend the segment that the receiver is expecting to fill in the gap in the data, without waiting for the retransmission timer to expire for that segment. This optimization greatly improves performance in a busy network environment.

With fast retransmit, the sender retransmits the TCP segments to fill in the gap in the data before their retransmission timers expire. Because the retransmission timers did not expire for the missing TCP segments, the sender can more quickly send additional segments to the receiver. This is known as *fast recovery*. Fast retransmit and fast recovery are described in RFC 2581.

By default, Windows Server 2003 resends a segment if it receives three ACKs for the same sequence number and that sequence number lags the current one. This is controllable with the *TcpMaxDupAcks* registry parameter. See also the “TCP Selective Acknowledgment (RFC 2018)” section in this paper.

TCP Keep-Alive Messages

A TCP keep-alive packet is simply an ACK with the sequence number set to one less than the current sequence number for the connection. A host receiving one of these ACKs responds with an ACK for the current sequence number. Keep-alives can be used to verify that the computer at the remote end of a connection is still available. TCP keep-alives can be sent once every *KeepAliveTime* (defaults to 7,200,000 milliseconds or two hours) if no other data or higher-level keep-alives have been carried over the TCP connection. If there is no response to a keep-alive, it is repeated once every *KeepAliveInterval* seconds. *KeepAliveInterval* defaults to 1 second. NetBT connections, such as those used by other Microsoft networking components, send NetBIOS keep-alives more frequently, so normally no TCP keep-alives are sent on a NetBIOS connection. TCP keep-alives are disabled by default, but Windows Sockets applications can use the *setsockopt*() function to enable them.

Slow Start Algorithm and Congestion Avoidance

When a connection is established, TCP starts slowly at first to assess the bandwidth of the connection, and to avoid overflowing the receiving host or any other devices or links in the path. The send window is set to two TCP segments, and if that is acknowledged, it is incremented to three segments. Instead of sending one TCP segment when starting out, Windows Server 2003 TCP starts with two. This avoids the need to wait for the delayed ACK timer to expire on the first send to the target computer, which improves performance for some applications.

If those are acknowledged, it is incremented again, and so on until the amount of data being sent per burst reaches the size of the receive window on the remote host. At that point, the slow start algorithm is no longer in use, and flow control is governed by the receive window. However, congestion could still occur on a connection at any time during transmission. If this happens (evidenced by the need to

retransmit), a congestion-avoidance algorithm is used to reduce the send window size temporarily and to grow it back towards the receive window size. Slow start and congestion avoidance are described in RFCs 2581.

Silly Window Syndrome (SWS)

Silly Window Syndrome is described in RFC 1122 as follows:

> "In brief, SWS is caused by the receiver advancing the right window edge whenever it has any new buffer space available to receive data and by the sender using any incremental window, no matter how small, to send more data [TCP:5]. The result can be a stable pattern of sending tiny data segments, even though both sender and receiver have a large total buffer space for the connection."

Windows Server 2003 TCP/IP implements SWS avoidance, as specified in RFC 1122, by not sending more data until there is a sufficient window size advertised by the receiving end to send a full TCP segment. It also implements SWS avoidance on the receive end of a connection by not opening the receive window in increments of less than a TCP segment.

Nagle Algorithm

Windows Server 2003 TCP/IP implements the Nagle algorithm described in RFC 896. The purpose of this algorithm is to reduce the number of very small segments sent, especially on high-delay (remote) links. The Nagle algorithm allows only one small segment to be outstanding at a time without acknowledgment. If more small segments are generated while awaiting the ACK for the first one, these segments are coalesced into one larger segment. Any full-sized segment is always transmitted immediately, on the assumption that there is a sufficient receive window available. The Nagle algorithm is effective in reducing the number of packets sent by interactive applications, such as Telnet, especially over slow links.

The Nagle algorithm can be observed in the following Network Monitor capture. The trace was captured by using PPP to dial up an Internet service provider. A Telnet (character-mode) session was established, and then the **Y** key was held down on the computer. At all times, one segment was sent, and further **Y** characters were held by the stack until an acknowledgment was received for the previous segment. In this example, three to four **Y** characters were buffered each time and sent together in one segment. The Nagle algorithm resulted in a huge savings in the number of packets sent—the number of packets was reduced by a factor of about three.

```
Delta Source IP     Dest IP       Prot   Description
-------------------------------------------------------
0.644 204.182.66.83 199.181.164.4 TELNET To Server Port = 1901
0.144 199.181.164.4 204.182.66.83 TELNET To Client Port = 1901
0.000 204.182.66.83 199.181.164.4 TELNET To Server Port = 1901
0.145 199.181.164.4 204.182.66.83 TELNET To Client Port = 1901
0.000 204.182.66.83 199.181.164.4 TELNET To Server Port = 1901
0.144 199.181.164.4 204.182.66.83 TELNET To Client Port = 1901
. . .
```

Each segment contained several of the "y" characters. The first segment is shown more fully parsed below, and the data portion is pointed out in the hexadecimal display at the bottom.

```
************************************************************
Time  Source IP      Dest IP        Prot    Description
0.644 204.182.66.83  199.181.164.4  TELNET  To Server Port = 1901
+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ IP: ID = 0xEA83; Proto = TCP; Len: 43
+ TCP: .AP..., len: 3, seq:1032660278, ack: 353339017, win: 7766, src: 1901 dst: 23
(TELNET)
 TELNET: To Server From Port = 1901
   TELNET: Telnet Data

D2 41 53 48 00 00 52 41 53 48 00 00 08 00 45 00  .ASH..RASH....E.
00 2B EA 83 40 00 20 06 F5 85 CC B6 42 53 C7 B5  .+..@. .....BS..
A4 04 07 6D 00 17 3D 8D 25 36 15 0F 86 89 50 18  ...m..=.%6....P.
1E 56 1E 56 00 00 79 79 79                       .V.V..yyy
                                                       ^^^
                                                       data
```

Windows Sockets applications can disable the Nagle algorithm for their connections by setting the TCP_NODELAY socket option. However, this practice should be avoided unless it is absolutely necessary because it increases network utilization. Some network applications may not perform well if their design does not take into account the effects of transmitting large numbers of small packets and the Nagle algorithm. The Nagle algorithm is not applied to loopback TCP connections for performance reasons. Windows Server 2003 NetBT disables Nagling for NetBIOS over TCP connections as well as direct-hosted redirector/server connections, which can improve performance for applications issuing numerous small file manipulation commands. An example is an application that uses file locking/unlocking frequently.

TCP TIME-WAIT Delay

When a TCP connection is closed, the socket-pair is placed into a state known as TIME-WAIT. This is done so that a new connection does not use the same protocol, source IP address, destination IP address, source port, and destination port until enough time has passed to ensure that any segments that may have been misrouted or delayed are not delivered unexpectedly. RFC 793 specifies the length of time that the socket-pair should not be reused as two maximum segment lifetimes (2 MSL), or four minutes. This is the default setting for Windows Server 2003 TCP/IP. However, with this default setting, some network applications that perform many outbound connections in a short time may use up all available ports before the ports can be recycled.

Windows Server 2003 TCP/IP offers two methods of controlling this behavior. First, the *TcpTimedWaitDelay* registry parameter can be used to alter this value. Windows Server 2003 TCP/IP allows it to be set as low as 30 seconds, which should not cause problems in most environments. Second, the number of user-accessible ephemeral ports that can be used to source outbound connections is configurable using the *MaxUserPort* registry parameter. By default, when an application requests any socket from the system to use for an outbound call, a port between the values of 1024 and 5000 is supplied. The *MaxUserPort* parameter can be used to set the value of the uppermost port that the administrator chooses to allow for outbound connections. For instance, setting this value to

10,000 (decimal) would make approximately 9000 user ports available for outbound connections. For more details on this concept, see RFC 793. See also the *MaxFreeTcbs* and *MaxHashTableSize* registry parameters in Appendix A.

TCP/IP for Windows Server 2003 Service Pack 1 has implemented a smart TCP port allocation algorithm. When an application requests any available TCP port, TCP/IP first attempts to find an available port that does not correspond to a connection in the TIME-WAIT state. If a port cannot be found, then it picks any available port. For more information, see "Smart TCP Port Allocation" in this paper.

TCP Connections to and from Multihomed Computers

When TCP connections are made to a multihomed host, both the WINS client and the Domain Name Resolver (DNR) attempt to determine whether any of the destination IP addresses provided by the name server are on the same subnet as any of the interfaces in the local computer. If so, these addresses are sorted to the top of the list so that the application can try them prior to trying addresses that are not on the same subnet. If none of the addresses is on a common subnet with the local computer, behavior is different depending upon the name space. The *PrioritizeRecordData* TCP/IP registry parameter can be used to prevent the DNR component from sorting local subnet addresses to the top of the list.

In the WINS name space, the client is responsible for randomizing or load balancing between the provided addresses. The WINS server always returns the list of addresses in the same order, and the WINS client randomly picks one of them for each connection.

In the DNS name space, the DNS server is usually configured to provide the addresses in a round robin fashion. The DNR does not attempt to further randomize the addresses. In some situations, it is desirable to connect to a specific interface on a multihomed computer. The best way to accomplish this is to provide the interface with its own DNS entry. For example, a computer named *raincity* could have one DNS entry listing both IP addresses (actually two separate records in the DNS with the same name), and also records in the DNS for raincity1 and raincity2, each associated with just one of the IP addresses assigned to the computer.

When TCP connections are made from a multihomed host, things get a bit more complicated. If the connection is a Winsock connection using the DNS name space, once the target IP address for the connection is known, TCP attempts to connect from the best source IP address available. Again, the route table is used to make this determination. If there is an interface in the local computer that is on the same subnet as the target IP address, its IP address is used as the source in the connection request. If there is no best source IP address to use, the system chooses one randomly.

If the connection is a NetBIOS-based connection using the redirector, little routing information is available at the application level. The NetBIOS interface supports connections over various protocols and has no knowledge of IP. Instead, the redirector places calls on all of the transports that are bound to it. If there are two interfaces in the computer and one protocol installed, there are two transports available to the redirector. Calls are placed on both, and NetBT submits connection requests to the stack, using an IP address from each interface. It is possible that both calls succeed. If so, the redirector cancels one of them. The choice of which one to cancel depends upon the redirector *ObeyBindingOrder* registry value. See the Microsoft Knowledge Base at http://search.support.microsoft.com/search/?adv=1 for redirector registry parameters. If this is set to 0

(the default value), the primary transport (determined by binding order) is the preferred one, and the redirector waits for the primary transport to time out before accepting the connection on the secondary transport. If this value is set to 1, the binding order is ignored, and the redirector accepts the first connection that succeeds and cancels the other(s).

Throughput Considerations

TCP was designed to provide optimum performance over varying link conditions, and Windows Server 2003 TCP/IP contains improvements such as those supporting RFC 1323. Actual throughput for a link depends on a number of variables, but the most important factors are:

- Link speed (bits-per-second that can be transmitted)
- Propagation delay
- Window size (amount of unacknowledged data that may be outstanding on a TCP connection)
- Link reliability
- Network and intermediate device congestion
- Path MTU

TCP throughput calculation is discussed in detail in Chapters 20–24 of *TCP/IP Illustrated,* by W. Richard Stevens (Stevens, Richard. *TCP/IP Illustrated*, *Volume 1: The Protocols*. Reading, MA: Addison-Wesley Publishing Co., 1993). Some key considerations are listed below:

- The capacity of a pipe is bandwidth multiplied by round-trip time. This is known as the *bandwidth-delay product*. If the link is reliable, for best performance the window size should be greater than or equal to the capacity of the pipe so that the sending stack can fill it. The largest window size that can be specified, due to its 16-bit field in the TCP header, is 65535, but larger windows can be negotiated by using window scaling as described earlier in this paper. See *TcpWindowSize* in Appendix A.
- Throughput can never exceed window size divided by round-trip time.
- If the link is unreliable or badly congested and packets are being dropped, using a larger window size may not improve throughput. Along with scaling windows support, Windows Server 2003 TCP/IP supports Selective Acknowledgments (SACK; described in RFC 2018) to improve performance in environments that are experiencing packet loss. It also includes support for timestamps (described in RFC 1323) for improved RTT estimation.
- Propagation delay is dependent upon the speed of light, latencies in transmission equipment, and so on.
- Transmission delay depends on the speed of the media.
- For a specified path, propagation delay is fixed, but transmission delay depends upon the packet size.
- At low speeds, transmission delay is the limiting factor. At high speeds, propagation delay may become the limiting factor.

To summarize, Windows Server 2003 TCP/IP can adapt to most network conditions and can dynamically provide the best throughput and reliability possible on a per-connection basis. Attempts at

manual tuning are often counter-productive unless a qualified network engineer first performs a careful study of data flow.

User Datagram Protocol (UDP)

UDP provides a connectionless, unreliable transport service. It is often used for communications that use broadcast or multicast IP datagrams. Since delivery of UDP datagrams is not guaranteed, applications using UDP must supply their own mechanisms for reliability, if needed. Microsoft networking components use UDP for logon, browsing, and name resolution. UDP can also be used to carry IP multicast streams.

UDP and Name Resolution

UDP is used for NetBIOS name resolution by unicast to a NetBIOS name server or subnet broadcasts, and for DNS host name to IP address resolution. NetBIOS name resolution is accomplished over UDP port 137. DNS queries use UDP port 53. Because UDP itself does not guarantee delivery of datagrams, both of these services use their own retransmission schemes if they receive no answer to queries. Broadcast UDP datagrams are not usually forwarded over IP routers, so NetBIOS name resolution in a routed environment requires a name server of some type, or the use of static database files.

Mailslots over UDP

Many NetBIOS applications use *mailslot* messaging. A second-class mailslot is a simple mechanism for sending a message from one NetBIOS name to another over UDP. Mailslot messages can be broadcast on a subnet or directed to the remote host. To direct a mailslot message to another host, there must be some method of NetBIOS name resolution available. Microsoft provides WINS for this purpose.

NetBIOS over TCP/IP (NetBT)

The Windows Server 2003 implementation of NetBIOS over TCP/IP is referred to as *NetBT*. NetBT uses the following TCP and UDP ports:

- UDP port 137 (name services)
- UDP port 138 (datagram services)
- TCP port 139 (session services)

NetBIOS over TCP/IP is specified by RFC 1001 and RFC 1002. The Netbt.sys driver is a kernel-mode component that supports the Transport Driver Interface (TDI) interface. Services such as Workstation (redirector) and Server (file server) use the TDI interface directly, but traditional NetBIOS applications have their calls mapped to TDI calls by the Netbios.sys driver. Using TDI to make calls to NetBT is a more difficult programming task, but can provide higher performance and freedom from historical NetBIOS limitations. NetBIOS concepts are discussed further in the “Network Application Interfaces” section of this paper.

Transport Driver Interface (TDI)

Microsoft developed the Transport Driver Interface (TDI) to provide greater flexibility and functionality than is provided by existing interfaces, such as NetBIOS and Windows Sockets. All Windows transport providers expose TDI. The TDI specification describes the set of primitive functions by which transport

drivers and TDI clients communicate and the call mechanisms used for accessing them. Currently, TDI is kernel-mode only.

The Windows Server 2003 redirector and server both use TDI directly, rather than going through the NetBIOS mapping layer. By doing so, they are not subject to many of the restrictions imposed by NetBIOS, such as the legacy 254-session limit.

TDI Features

TDI may be the most difficult to use of all Windows network APIs. It is a simple conduit, so the programmer must determine the format and meaning of messages.

TDI includes the following features:

- Most Windows Server 2003 transports support TDI
- An open naming and addressing scheme
- Message and stream-mode data transfer
- Asynchronous operation
- Support for unsolicited indication of events
- Extensibility—clients can submit private requests to a transport driver that understands them.
- Support for limited use of standard kernel-mode I/O functions to send and receive data
- 32-bit addressing and values
- Support for Access Control Lists (ACLs, used for security) on TDI address objects

More information about TDI is available from the Windows DDK.

Security Considerations

Network security is a serious consideration for administrators with computers exposed to public networks. Microsoft's TCP/IP stack has been strengthened against many attacks and in its default state handles most of the common attacks. Some additional protection against popular Denial of Service attacks can be added by setting the value of the *SynAttackProtect* parameter in the registry. This key allows the administrator to choose several levels of protection against SYN attacks.

Here are general guidelines that can lower your exposure to attack:

- Disable unnecessary or optional services (for instance, the Client for Microsoft Networks and the File and Printer Sharing for Microsoft Networks components on the network connections of an IIS server).
- Enable TCP/IP filtering and restrict access to only the ports that are necessary for the server to function. See the Windows NT, Terminal Server, and Microsoft Exchange Services Use TCP/IP Ports for a list of ports that Windows services use.
- Disable NetBIOS over TCP/IP on network connections where it is not needed.
- Configure static IP addresses and parameters for network adapters connected to the Internet.
- Configure registry parameters for maximum protection (see Appendix D).

Consult the Microsoft Security Web site regularly for security bulletins.

Network Application Interfaces

There are a number of ways that network applications can communicate using the TCP/IP protocol stack. Some of them, such as named pipes, go through the network redirector, which is part of the Workstation service. Many older applications were written to the NetBIOS interface, which is supported by NetBIOS over TCP/IP.

The Windows Sockets interface is the most popular API for writing Windows network applications today. A quick overview of the Windows Sockets Interface and the NetBIOS Interface is presented here.

Windows Sockets

Windows Sockets specifies a programming interface that was originally based on the familiar socket interface from the University of California at Berkeley. It includes a set of extensions designed to take advantage of the message-driven nature of Microsoft Windows. Version 1.1 of the specification was released in January 1993, and version 2.2.0 was published in May of 1996. Windows Server 2003 supports version 2.2, commonly referred to as Winsock2.

Applications

There are many Windows Sockets applications available. A number of the utilities that ship with Windows Server 2003 are based on Windows Sockets, including the FTP and DHCP clients and servers, Telnet client, and so on. There are also higher-level programming interfaces that rely on Winsock, such as the Windows Internet API (WinInet) used by Internet Explorer.

Name and Address Resolution

Traditionally, Windows Sockets applications use the gethostbyname() function to resolve a host name to an IP address and the gethostbyaddr() function to resolve an IP address to a host name. The gethostbyname() and gethostbyaddr() functions are specific to IPv4. To accommodate the use of either IPv4 or IPv6, these functions have been deprecated in RFC 2553 and replaced by the new functions getaddrinfo() and getnameinfo(). Both getaddrinfo() and getnameinfo() are independent of whether IPv4, IPv6, or both are installed on the computer.

To allow applications to run over both IPv4 and IPv6, application developers must modify their applications to use getaddrinfo() and getnameinfo() and make other modifications. For more information about modifying applications to run over both IPv4 and IPv6, see the IPv6 Guide for Windows Sockets Applications at http://technet.microsoft.com/en-us/network/bb531150.aspx.

The gethostbyname() or getaddrinfo() functions use the following (default) name look-up sequence:

1. Check the local host name for a matching name.
2. Check the Hosts file for a matching name entry.
3. If a DNS server is configured, query it.
4. If no match is found, attempt NetBIOS name-resolution.

The gethostbyaddr() or getnameinfo() functions use the following (default) address look-up sequence:

1. Check the Hosts file for a matching address entry.
2. If a DNS server is configured, query it.
3. Send a NetBIOS Adapter Status Request to the IP address being queried. If it responds with a list of NetBIOS names registered for the adapter, parse it for the computer name.

Support for IP Multicasting

Winsock2 provides support for IP multicasting. Multicasting is described in the Windows Sockets 2.0 specification and in the IGMP section of this paper. IP multicasting is currently supported only on AF_INET and AF_INET6 sockets of the types SOCK_DGRAM and SOCK_RAW.

Winsock2 and Windows Server 2003 also support IP multicasting on SOCK_RDM sockets, which provide reliable multicast delivery using the Pragmatic General Multicast (PGM) protocol. Installing the Reliable Multicast Protocol through Network Connections enables PGM and reliable multicast support. The Reliable Multicast Protocol installs a driver, Rmcast.sys, which communicates directly with the TCP/IP protocol to provide its own transport layer. Windows XP does not include the Reliable Multicast Protocol. PGM is described in RFC 3208.

Backlog Parameter

Windows Sockets server applications generally create a socket, and then use the listen() function on it to listen for connection requests. One of the parameters passed when calling listen() is the backlog of connection requests that the application would like Windows Sockets to queue for it. This value controls the number of unaccepted connections that can be queued. Once an application *accepts* a connection, it is moved out of the connection request backlog and no longer counts. The Windows Sockets 1.1 specification indicates that the maximum allowable value for a backlog is 5; however, Windows NT 4.0, Windows 2000 Server, and Windows Server 2003 accept a backlog of 200. Windows XP Professional accepts a backlog of 5.

Push Bit Interpretation

By default, Windows Server 2003 TCP/IP completes a recv() call when one of the following conditions is met:

- Data arrives with the PUSH bit set
- The user recv buffer is full
- 0.5 seconds have elapsed since any data has arrived

If a client application is run on a computer with a TCP/IP implementation that does not set the push bit on send operations, response delays may result. It is best to correct this on the client; however, a configuration parameter (*IgnorePushBitOnReceives*) was added to Afd.sys to force it to treat all arriving packets as though the push bit were set.

ConnectEx/TransmitPackets and TCP/IP

The following two functions are Microsoft-specific extensions to the Windows Sockets 2 specification:

- The Windows Sockets ConnectEx() function establishes a connection to another socket application and optionally sends the block of data after the connection is established.

- The Windows Sockets TransmitPackets() function transmits in memory and/or file data over a connected socket (either datagram or stream). The operating system's cache manager is used to retrieve file data and locks memory for the minimum necessary time required to transmit it. This provides high performance and efficiency for file and memory data transfer over sockets.

Windows Sockets Direct Path for System Area Networks

The Windows Server 2003 family contains substantial performance improvements to Windows Sockets Direct (WSDP) for System Area Networks (SANs). WSD enables Windows Sockets applications written for SOCK_STREAM to obtain the performance benefits of SANs without having to make application modifications. The fundamental component of this technology is the WinSock switch that emulates TCP/IP semantics over native SAN service providers. For the Windows 2000 Server family, WSD support was only available for Windows 2000 Advanced Server and Windows 2000 Datacenter Server. WSD support is included for all members of the Windows Server 2003 family.

NetBIOS over TCP/IP

NetBIOS defines a software interface and a naming convention, not a protocol. Early versions of Microsoft networking products provided only the NetBIOS Extended User Interface (NetBEUI) protocol with a NetBIOS application-programming interface. NetBEUI is a small, fast protocol with no networking layer; thus, it is not routable and is not suitable for internetworks. NetBEUI relies on multicasts for name resolution and location of services. NetBIOS over TCP/IP provides the NetBIOS programming interface over the TCP/IP protocol, extending the reach of NetBIOS client and server programs to the WAN, and providing interoperability with various other operating systems.

The Workstation, Server, Browser, Messenger, and NetLogon services are all (direct) NetBT clients. They use TDI (described earlier in this paper) to communicate with NetBT. Windows Server 2003 also includes a NetBIOS emulator. The emulator takes standard NetBIOS requests from NetBIOS applications and translates them to equivalent TDI primitives.

Windows Server 2003 still uses NetBIOS over TCP/IP to communicate with prior versions of Windows NT and other clients, such as Windows 98. However, the Windows Server 2003 redirector and server components also support *direct hosting* to communicate with other computers capable of direct hosting. Direct hosting uses DNS for name resolution. No NetBIOS name resolution (WINS or broadcast) is used, and the protocol is simpler. Direct hosting uses TCP port 445, instead of the NetBIOS TCP port 139.

By default, both NetBIOS and direct hosting are enabled, and both are tried in parallel when a new connection is established. The first to succeed in connecting is used for any attempt. NetBIOS support can be disabled to force all traffic to use direct hosting.

To disable NetBIOS support

1. On the **Start** menu, point to **Control Panel**, and then double-click **Network Connections**. Right-click **Local Area Connection** (or the name of your LAN connection) and then click **Properties**.
2. Select **Internet Protocol (TCP/IP)** in the list of components, and click **Properties**.
3. Click **Advanced**.
4. On the **WINS** tab, select **Disable NetBIOS over TCP/IP**.

Applications and services that depend on NetBIOS no longer function after this is done, so it is important that you verify that any clients and applications no longer need NetBIOS support before you disable it. For example, computers that cannot direct host will be unable to browse, locate, or create file and print share connections to a Windows Server 2003 computer with NetBIOS disabled.

NetBIOS Names

The NetBIOS namespace is flat, meaning that all names within the name space must be unique. NetBIOS names are 16 characters in length. Resources are identified by NetBIOS names, which are registered dynamically when computers boot, services or applications start, or users log on. Names can be registered as unique (one owner) or as group (multiple owner) names. A NetBIOS Name Query is used to locate a resource by resolving the name to an IP address.

Microsoft networking components, such as Workstation and Server services, allow the first 15 characters of a NetBIOS name to be specified by the user or administrator, but reserve the sixteenth character of the NetBIOS name to indicate a resource type (0x00-0xFF). Many popular third-party software packages also use this character to identify and register their specific services. Table 3 lists some example NetBIOS names used by Microsoft components.

Table 3. Examples of NetBIOS names used by Windows components

Unique name	Service
computer_name[00h]	Workstation service
computer_name[03h]	Messenger service
computer_name[1Fh]	NetDDE service
computer_name[20h]	Server service
computer_name[BEh]	Network Monitor Agent
computer_name[BFh]	Network Monitor Application
user_name[03]	Messenger service
domain_name[1Dh]	Master Browser
domain_name[1Bh]	Domain Master Browser
domain_name[00h]	Domain name
domain_name[1Ch]	Domain controllers
domain_name[1Eh]	Browser service elections
\\--__MSBROWSE__[01h]	Master browser

To see which names a computer has registered over NetBT, type the following from a command prompt:

```
nbtstat -n
```

Windows Server 2003 allows you to re-register names with the WINS server after a computer has already been started. To do this, type the following from a command prompt:

```
nbtstat –RR
```

NetBIOS Name Registration and Resolution

Windows TCP/IP systems use several methods to locate NetBIOS resources:

- NetBIOS name cache
- NetBIOS name server
- Broadcast NetBIOS Name Query Request messages
- Static Lmhosts file
- Local host name (optional, depends on *EnableDns* registry parameter)
- Static Hosts file (optional, depends on *EnableDns* registry parameter)
- DNS servers (optional, depends on *EnableDns* registry parameter)

NetBIOS name resolution order depends upon the node type and system configuration. The following node types are supported:

- *B-node* uses broadcasts for name registration and resolution.
- *P-node* uses a NetBIOS name server (such as WINS) for name registration and resolution.
- *M-node* uses broadcasts for name registration. For name resolution, it tries broadcasts first, but switches to p-node if it receives no answer.
- *H-node* uses a NetBIOS name server for both registration and resolution. However, if no name server can be located, it switches to b-node. It continues to poll for a name server and switches back to p-node when one becomes available.
- *Microsoft-enhanced* uses the local Lmhosts file or WINS proxies plus Windows Sockets *gethostbyname*() calls (using standard DNS and/or local Hosts files) in addition to standard node types.

Microsoft ships a NetBIOS name server known as the Windows Internet Name Service (WINS). Most WINS clients are set up as H-nodes; that is, they first attempt to register and resolve names using WINS, and if that fails, they try local subnet broadcasts. Using a WINS server to locate resources is generally preferable to broadcasting for two reasons:

- Routers do not usually forward broadcasts.
- All computers on a subnet process broadcasts.

NetBIOS Name Registration and Resolution for Multihomed Computers

As previously described, NetBT binds to only one IP address per physical network interface. From the NetBT viewpoint, a computer is multihomed only if it has more than one NIC installed. When a name registration packet is sent from a multihomed machine, it is flagged as a multihomed name registration so that it does not conflict with the same name being registered by another interface in the same computer.

If a multihomed machine receives a broadcast name query, all NetBT-interface bindings receiving the query respond with their addresses, and by default the client chooses the first response and connects to the address supplied by it. This behavior can be controlled by the *RandomAdapter* registry parameter described in Appendix B.

When a directed name query is sent to a WINS server, the WINS server responds with a list of all IP addresses that were registered with WINS by the multihomed computer.

Choosing the best IP address to connect to on a multihomed computer is a client function. Currently, the following algorithm is employed, in the order listed:

1. If one of the IP addresses in the name query response list is on the same logical subnet as the calling binding of NetBT on the local computer, that address is selected. If more than one of the addresses meets the criteria, one is picked at random from those that match.
2. If one of the IP addresses in the list is on the same (classless) network as the calling binding of NetBT on the local computer, that address is selected. If more than one of the addresses meets the criteria, one is picked at random from those that match.
3. If one of the IP addresses in the list is on the same logical subnet as any binding of NetBT on the local computer, that address is selected. If more than one of the addresses meets the criteria, one is picked at random from those.
4. If none of the IP addresses in the list is on the same subnet as any binding of NetBT on the local computer, an address is selected at random from the list.

This algorithm provides a reasonably good way of balancing connections to a server across multiple NICs, and still favoring direct (same subnet) connections when they are available. When a list of IP addresses is returned, they are sorted into the best order, and NetBT attempts to ping each of the addresses in the list until one responds. NetBT then attempts a connection to that address. If no addresses respond, a connection attempt is made to the first address in the list anyway. This is tried in case there is a firewall or other device filtering ICMP traffic. Windows Server 2003 supports per interface NetBT name caching, and the **nbtstat -c** command displays the name cache on a per-interface basis.

NetBT Internet/DNS Enhancements and the SMB Device

It has always been possible to connect from one Windows-based computer to another using NetBT over the Internet. To do so, some means of name resolution had to be provided. Two common methods were to use the Lmhosts file or a WINS server. Several enhancements were introduced in Windows NT 4.0 and carried forward in Windows Server 2003 to eliminate these special configuration needs.

It is possible to connect to a NetBIOS over TCP/IP resource in the following ways:

- Use the command **net use ***ip address******share_name*. This eliminates the need for NetBIOS name resolution configuration.
- Use the command **net use ***FQDN******share_name*. This allows the use of a DNS to connect to a computer using its fully qualified domain name (FQDN).

Examples of using new functionality to map a drive to ftp.microsoft.com are shown here. The IP address listed here is subject to change.

- **net use f: \\ftp.microsoft.com\data**
- **net use \\131.107.232.1\data**
- **net view \\131.107.232.1**
- **dir \\ftp.microsoft.com\bussys\winnt**

In addition, various applications, such as the Event Viewer **Select Computer** option on the **Log** menu, allow you to enter an FQDN or IP address directly. In Windows Server 2003, it is also possible to use

direct hosting to establish redirector or server connections between computers running Windows Server 2003 without the use of the NetBIOS namespace or mapping layer at all. By default, Windows attempts to make connections using both methods so that it can support connections to lower-level computers. However, in Windows XP and Windows Server 2003–only environments, you can disable NetBIOS completely for each network connection in Network Connections.

The interface in Windows Server 2003 that makes NetBIOS-less operation possible is termed the *Server Message Block (SMB) device.* It appears to the redirector and server as another interface, much as an individual network adapter/protocol stack combination does. At the TCP/IP stack however, the SMB device is bound to ADDR_ANY, and it uses the DNS namespace natively, like a Windows Sockets application. Calls placed on the SMB device will result in a standard DNS lookup to resolve the (DNS) name to an IP address, followed by a single outbound connection request (even on a multihomed computer) using the best source IP address and interface as determined by the route table. Additionally, there is no NetBIOS session setup on top of the TCP connection, as there is with traditional NetBIOS over TCP/IP. By default, the redirector places calls on both the NetBIOS device(s) and the SMB device, and the file server receives calls on both. The file server SMB device listens on TCP port 445 instead of the traditional NetBIOS over TCP port 139.

NetBIOS over TCP Sessions

NetBIOS sessions are established between two names. For example, when a Windows XP Professional-based workstation makes a file-sharing connection to a server using NetBIOS over TCP/IP, the following sequence of events takes place:

1. The NetBIOS name for the server is resolved to an IP address.
2. The next-hop address for the IP address of the server is resolved to a MAC address.
3. A TCP connection is established from the workstation to the server, using port 139.
4. The workstation sends a NetBIOS Session Request to the server name over the TCP connection. If the server is listening on that name, it responds affirmatively, and a session is established.

When the NetBIOS session has been established, the workstation and server negotiate which level of the SMB protocol to use. Microsoft networking uses only one NetBIOS session between two names at any time. Any additional file or print sharing connections are multiplexed over the same NetBIOS session using identifiers within the SMB header.

NetBIOS keep-alives are used on each connection to verify that both the server and workstation are still able to maintain their session. Therefore, if a workstation is shut down ungracefully, the server eventually cleans up the connection and associated resources, and vice versa. NetBIOS keep-alives are controlled by the *SessionKeepAlive* registry parameter and default to once per hour.

If Lmhosts files are used and an entry is misspelled, it is possible to attempt to connect to a server using the correct IP address but an incorrect name. In this case, a TCP connection is still established to the server. However, the server rejects the NetBIOS session request (using the wrong name), because there is no listen posted on that name. An Error 51, “Remote computer not listening,” is returned.

NetBIOS Datagram Services

Datagrams are sent from one NetBIOS name to another over UDP port 138. The datagram service provides the ability to send a message to a unique name or to a group name. Group names may resolve to a list of IP addresses or a broadcast. For example, the command **net send /d:***mydomain* **test** sends a datagram containing the text "test" to the group name *mydomain*[03]. The *mydomain*[03] name resolves to an IP subnet broadcast, so the datagram is sent with the following characteristics:

- Destination MAC address: broadcast (0xFF-FF-FF-FF-FF-FF).
- Source MAC address: The MAC address of the sending interface.
- Destination IP address: The local subnet broadcast address.
- Source IP address: The IP address of the local computer.
- Destination name: *mydomain*[03] (the messenger service on the remote computers).
- Source name: *username*[03] (the messenger service on the local computer).

All hosts on the subnet pick up the datagram and process it, at least to the UDP protocol. On hosts that are running a NetBIOS datagram service, UDP hands the datagram to NetBT on port 138. NetBT checks the destination name to see if any application has posted a datagram receive on it and if so, passes the datagram up. If no receive is posted, the datagram is discarded.

If support for NetBIOS is disabled in Windows Server 2003 (as described earlier in this section), NetBIOS datagram services are not available.

Critical Client Services and Stack Components

The focus of this paper is on core TCP/IP stack components, not on the many available services that use it. However, the stack itself relies upon a few services for configuration information and name and address resolution. A few of these critical client services are discussed here.

Automatic Client Configuration and Media Sense

One of the most important client services is the Dynamic Host Configuration Protocol (DHCP) client. The DHCP client has an expanded role in Windows Server 2003. Its primary new feature is the ability to automatically configure an IP address and subnet mask when the client is started on a small private network without a DHCP server available to assign addresses (such as a home network). Another new feature is support for *Media Sense*, which can improve the roaming experience for portable device users.

If the TCP/IP is configured to dynamically obtain TCP/IP protocol configuration information from a DHCP server (instead of being manually configured with an IP address and other parameters), the DHCP client service is engaged each time the computer is restarted. The DHCP client service now uses a two-step process to configure the client with an IP address and other configuration information.

When the client is installed, it attempts to locate a DHCP server and obtain a configuration from it. Many TCP/IP networks use DHCP servers that are administratively configured to allocate TCP/IP configuration information to clients on the network. If this attempt to locate a DHCP server fails, the Windows Server 2003 DHCP client uses one of the following:

1. If Automatic Private IP Addressing (APIPA) is selected, TCP/IP automatically selects an IP address from the IANA-reserved class B network 169.254.0.0 with the subnet mask 255.255.0.0. The subnet and subnet mask used can be controlled using the IPAutoconfigurationSubnet and IPAutoconfigurationMask registry keys. These keys are described in Appendix A. The DHCP client tests (using a gratuitous ARP) to make sure that the IP address that it has chosen is not already in use. If it is in use, it selects another IP address (it does this for up to 10 addresses). Once the DHCP client has selected an address that is verifiably not in use, it configures the interface with this address. APIPA allows single subnet home office or small office networks to use TCP/IP without static configuration or the administration of a DHCP server. Note that APIPA does not configure a default gateway. Therefore, only local subnet traffic is possible. IP autoconfiguration can be disabled using the IPAutoconfigurationEnabled registry key.
2. If an alternate configuration is selected, TCP/IP is configured with the alternate configuration settings. Alternate configuration is useful when a computer is used on more than one network, at least one of the networks does not have a DHCP server, and an APIPA configuration is not wanted. For example, if you have a laptop computer that you use both at the office and at home, it is useful to configure TCP/IP for an alternate configuration. At the office, the laptop uses a DHCP-allocated TCP/IP configuration. At home, where there is no DHCP server present, the laptop automatically uses the alternate configuration, which provides easy access to home network devices and the Internet and allows seamless operation on both networks, without the manual reconfiguration of TCP/IP settings.

You can select whether to use APIPA or an alternate configuration from the **Alternate Configuration** tab from the properties of the TCP/IP protocol in the Network Connections folder.

Once configured for either APIPA or an alternate configuration, TCP/IP continues to check for a DHCP server in the background every 5 minutes. If a DHCP server is found, the APIPA or alternate configuration information is abandoned, and the configuration offered by the DHCP server is used instead.

If the DHCP client has previously obtained a lease from a DHCP server, the following modified sequence of events occurs:

1. If the client's lease is still valid (not expired) at boot time, the client tries to renew its lease with the DHCP server. If the client fails to locate a DHCP server during the renewal attempt, it tries to ping the default gateway that is listed in the lease. If pinging the default gateway succeeds, the DHCP client assumes that it is still located on the same network where it obtained its current lease and continues to use the lease. By default, the client attempts to renew its lease in the background when half of its assigned lease time has expired.
2. If the attempt to ping the default gateway fails, the client assumes that it has been moved to a network that has no DHCP services currently available (such as a home network), and autoconfigures itself as described above. Once autoconfigured, it continues to try to locate a DHCP server every 5 minutes, in the background.

Media Sense support was added in NDIS 5.0. It provides a mechanism for the NIC to notify the protocol stack of media connect and media disconnect events. Windows Server 2003 TCP/IP utilizes these notifications to assist in automatic configuration. For instance, in Windows NT 4.0, when a portable computer was located and DHCP was configured on an Ethernet subnet, and then moved to another subnet without rebooting, the protocol stack received no indication of the move. This meant that the configuration parameters became stale, and not relevant to the new network. Additionally, if the computer was shut off, carried home and rebooted, the protocol stack was not aware that the NIC was no longer connected to a network, and again stale configuration parameters remained. This could be problematic, as subnet routes, default gateways, and so on, could conflict with dial-up parameters.

Media Sense support allows the protocol stack to react to events and invalidate stale parameters. For instance, if a computer running Windows Server 2003 is unplugged from the network (assuming the NIC supports Media Sense), after a damping period implemented in the stack (currently 3 seconds), TCP/IP will invalidate the parameters associated with the network which has been disconnected. The IP address(es) will no longer allow sends, and any routes associated with the interface are invalidated. You can make the network connection status when connected visible on the taskbar by selecting a connection, right-clicking it, clicking **Properties**, and then selecting the **Show icon in taskbar when connected** check box. The network connection icon will also appear automatically with a red "X" when the adapter is having a connectivity problem.

If an application is bound to a socket that is using an invalidated address, it should handle the event and recover in a graceful way, such as attempting to use another IP address on the system or notifying the user of the disconnect.

DNS Dynamic Update Client

Windows Server 2003 includes support for DNS dynamic updates as described in RFC 2136. Every time there is an address event (new address or renewal), the DHCP client sends option 81 and its fully qualified name to the DHCP server, and requests the DHCP server to register a DNS pointer (PTR) resource record (RR) on its behalf. The DHCP client handles the dynamic update of the address (A) RR

on its own. This is done because only the client knows which IP addresses on the host map to that name. The DHCP server may not be able to properly do the A RR registration because it has incomplete knowledge. However, the DHCP server can be configured to instruct the client to allow the server to register both records with the DNS. Registry parameters associated with the DNS dynamic update client are documented in Appendix C.

The Windows Server 2003 DHCP server handles option 81 requests as specified in the Internet draft titled "The DHCP Client FQDN Option" (draft-ietf-dhc-fqdn-option-0*x*.txt). If a Windows Server 2003 DHCP client obtains an IP address configuration from a down-level DHCP server that does not handle option 81, it registers a PTR RR on its own. The Windows Server 2003 DNS server is capable of handling dynamic updates.

Statically configured (non-DHCP) clients register both the A and PTR RRs with the DNS server themselves.

DNS Resolver Cache Service

Windows Server 2003 includes a caching DNS resolver service, which is enabled by default. For troubleshooting purposes, this service can be viewed, stopped, and started like any other Windows service. The caching resolver reduces DNS network traffic and speeds name resolution by providing a local cache for DNS queries. Name query responses are cached for the TTL specified in the response (not to exceed the value specified in the *MaxCacheEntryTtlLimit* parameter), and future queries are answered from the cache, when possible. One interesting feature of the DNS Resolver Cache Service is that it supports negative caching. For example, if a query is made to a DNS server for a given host name and the response is negative, succeeding queries for the same name are answered (negatively) from the cache for *NegativeCacheTime* seconds (the default is 300). Another example of negative caching is that if all DNS servers are queried and none are available, for *NetFailureCacheTime* seconds (the default is 30) all succeeding name queries fail instantly, instead of timing out. This feature can save time for services that query the DNS during the boot process, especially when the client is booted from the network.

The DNS Resolver Cache Service has a number of other adjustable registry parameters, which are documented in Appendix C.

Appendix A: TCP/IP Configuration Parameters

The TCP/IP protocol suite implementation for Windows Server 2003 obtains all of its configuration data from the registry. This information is written to the registry by the Setup program. Some of this information is also supplied by the Dynamic Host Configuration Protocol (DHCP) client service, if it is enabled. This appendix defines all of the registry parameters used to configure the protocol driver, Tcpip.sys, which implements the standard TCP/IP network protocols.

The implementation of the protocol suite should perform properly and efficiently in most environments using only the configuration information gathered by Setup and DHCP. Optimal default values for all other configurable aspects of the protocols for most cases have been encoded into the drivers. Some customer installations may require changes to certain default values. To handle these cases, optional registry parameters can be created to modify the default behavior of some parts of the protocol drivers.

Note: The Windows TCP/IP implementation is largely self-tuning. Adjusting registry parameters may adversely affect system performance.

All of the TCP/IP parameters are registry values located under the registry key

```
HKEY_LOCAL_MACHINE
     \SYSTEM
          \CurrentControlSet
               \Services:
                    \Tcpip
                         \Parameters
```

Adapter-specific values are listed under subkeys for each adapter identified by the adapter's globally unique identifier (GUID). To determine the GUID value for an adapter corresponding to a LAN connection in the Network Connections folder, do the following:

1. Open the Network Connections folder and note the name of the LAN connection, such as "Local Area Connection."
2. Click **Start**, click **Run**, type **regedit.exe**, and then click **OK**.
3. Use the tree view (the left pane) of the Registry Editor tool to open the following key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Network\{4D36E972-E325-11CE-BFC1-08002BE10318}
4. Under this key are one or more keys for the globally unique identifiers (GUIDs) corresponding to the installed LAN connections. Each of these GUID keys has a Connection subkey. Open each of the *GUID*\Connection keys and look for the Name setting in the contents pane whose value matches the name of your LAN connection from step 1.
5. When you have found the *GUID*\Connection key that contains the Name setting that matches the name of your LAN connection, write down or otherwise note the GUID value.

Depending on whether the system or adapter is DHCP-configured or static override values are specified, parameters may have both DHCP and statically configured values. If any of these parameters are changed using the registry editor, a restart of the system is generally required for the change to take effect. A restart is usually not required if values are changed using the Network Connections folder.

Parameters Configurable Using the Registry Editor

The following parameters receive default values during the installation of the TCP/IP components. To modify any of these values, use the Registry Editor (**Regedit.exe**). A few of the parameters are visible in the registry by default, but most must be created to modify the default behavior of the TCP/IP protocol driver. Parameters configurable from the user interface are listed separately.

AllowUserRawAccess

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (False, True)

Default: 0 (False)

Description: This parameter controls access to raw sockets. If true, non - administrative users have access to raw sockets. By default, only administrators have access to raw sockets. For more information on raw sockets, see the Windows Sockets Specifications, available from ftp://ftp.microsoft.com/bussys/winsock/winsock2/.

ArpAlwaysSourceRoute

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1, or not present (false, true, or not present)

Default: not present

Description: By default, the stack transmits ARP queries without source routing first and retries with source routing enabled if no reply is received. Setting this parameter to 0 causes all IP broadcasts to be sent without source routing. Setting this parameter to 1 forces TCP/IP to transmit all ARP queries with source routing enabled on Token Ring networks.

ArpCacheLife

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number of seconds

Valid Range: 0–0xFFFFFFFF

Default: In absence of an *ArpCacheLife* parameter, the defaults for ARP cache time-outs are a two-minute time-out on unused entries and a ten-minute time-out on used entries.

Description: See *ArpCacheMinReferencedLife*

ArpCacheMinReferencedLife

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number of seconds

Valid Range: 0–0xFFFFFFFF

Default: 600 seconds (10 minutes)

Description: *ArpCacheMinReferencedLife* controls the minimum time until a referenced ARP cache entry expires. This parameter can be used in combination with the *ArpCacheLife* parameter, as follows:

- If *ArpCacheLife* is greater than or equal to *ArpCacheMinReferencedLife*, referenced and unreferenced ARP cache entries expire in *ArpCacheLife* seconds.
- If *ArpCacheLife* is less than *ArpCacheMinReferencedLife*, unreferenced entries expire in *ArpCacheLife* seconds, and referenced entries expire in *ArpCacheMinReferencedLife* seconds.

Entries in the ARP cache are referenced each time that an outbound packet is sent to the IP address in the entry.

ArpRetryCount

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number

Valid Range: 0–3

Default: 3

Description: This parameter controls the number of times that the computer sends a gratuitous ARP for its own IP address(es) while initializing. Gratuitous ARPs are sent to ensure that the IP address is not already in use on the locally attached subnet. The value controls the actual number of ARPs sent, not the number of retries.

ArpTRSingleRoute

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Setting this parameter to 1 causes ARP broadcasts that are source-routed (Token Ring) to be sent as single-route broadcasts, instead of all-routes broadcasts.

ArpUseEtherSNAP

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Setting this parameter to 1 forces TCP/IP to transmit Ethernet packets using IEEE 802.3 SNAP encoding. By default, the stack transmits packets in Ethernet II format, also known as Ethernet DIX format. It always receives both formats.

DatabasePath

Key: Tcpip\Parameters

Value Type: REG_EXPAND_SZ—Character string

Valid Range: A valid file path

Default: *%SystemRoot%*\system32\drivers\etc

Description: This parameter specifies the path to the standard Internet database files (Hosts, Lmhosts, Network, Protocols, Services). It is used by the Windows Sockets interface.

DefaultTTL

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number of seconds/hops

Valid Range: 0–0xff (0–255 decimal)

Default: 128

Description: Specifies the default time-to-live (TTL) value set in the header of outgoing IP packets. The TTL determines the maximum amount of time that an IP packet may live in the network without reaching its destination. It is effectively a limit on the number of links on which an IP packet is allowed to travel before being discarded.

DisableDHCPMediaSense

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter can be used to control DHCP Media Sense behavior. If set to 1, the DHCP client ignores Media Sense events from the interface. By default, Media Sense events trigger the DHCP client to take an action, such as attempting to obtain a lease (when a connect event occurs), or invalidating the interface and routes (when a disconnect event occurs).

DisableIPSourceRouting

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1, 2

```
0 - forward all packets
1 - do not forward Source Routed packets
2 - drop all incoming Source Routed packets
```

Default: 1 (for Windows Server 2003 with no service packs installed), 2 (for Windows Server 2003 with Service Pack 1)

Description: IP source routing is a mechanism allowing the sender to determine the IP route that a datagram should take through the network. The Ping and Tracert tools have options to specify source routing.

DisableMediaSenseEventLog

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter can be used to disable logging of DHCP Media Sense events. By default, Media Sense events (connection/disconnection from the network) are logged in the event log for troubleshooting purposes.

DisableTaskOffload

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter instructs the TCP/IP stack to disable offloading of tasks to the NIC for troubleshooting and test purposes.

DisableUserTOSSetting

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter can be used to allow programs to manipulate the Type Of Service (TOS) bits in the header of outgoing IP packets. In Windows Server 2003, this defaults to **True**. In general, individual applications should not be allowed to manipulate TOS bits.

DontAddDefaultGateway

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0

Description: When you make a remote access connection that uses TCP/IP, a new default route is added to the route table with a metric lower than all other existing default routes. You can disable the automatic adding of this new default route by setting this registry parameter to 1. You can also disable this behavior from the properties of the TCP/IP protocol for a remote access connection in Network Connections. After doing so, you may need to configure static routes for destinations that are reachable across the remote access connection.

EnableAddrMaskReply

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter controls whether the computer responds to an ICMP address mask request.

EnableBcastArpReply

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter controls whether the computer responds to an ARP request when the source Ethernet address in the ARP is not unicast. Network Load Balancing Service (NLBS) will not work properly if this value is set to 0.

EnableDeadGWDetect

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: When this parameter is set to 1, TCP is allowed to perform dead gateway detection. With this feature enabled, TCP informs IP to change to a backup default gateway if a number of connections are experiencing difficulty. Backup default gateways are configured as advanced TCP/IP settings from the Network Connections folder. See the “Dead Gateway Detection” section in this paper for details.

EnableICMPRedirects

Key: Tcpip\Parameters

Value Type: REG_DWORD--BOOLEAN

Valid Range: 0, 1 (False, True)

Default: 1 (True)

Recommendation: 0 (False)

Description: This parameter controls whether Windows Server 2003 TCP/IP will update its route table in response to ICMP Redirect messages that are sent to it by network devices such as a routers.

EnableFastRouteLookup

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Fast route look-up is enabled if this flag is set. This can make route lookups faster at the expense of non-paged pool memory. This flag is used only if the computer runs Windows Server 2003 and falls into the medium or large class (in other words, contains at least 64 MB of memory). This parameter is created by the Routing and Remote Access service.

EnableMulticastForwarding

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: The routing service uses this parameter to control whether or not IP multicasts are forwarded. This parameter is created by the Routing and Remote Access service.

EnablePMTUBHDetect

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Setting this parameter to 1 (true) causes TCP to try to detect PMTU black hole routers while doing Path MTU Discovery. A PMTU black hole router does not return ICMP Destination Unreachable messages when it needs to fragment an IP datagram with the Don't Fragment bit set. TCP depends on receiving these messages to perform Path MTU Discovery. With this feature enabled, TCP tries to send segments without the Don't Fragment bit set if several retransmissions of a segment go unacknowledged. If the segment is acknowledged as a result, the MSS is decreased and the Don't Fragment bit is set in future packets on the connection. Enabling PMTU black hole detection increases the maximum number of retransmissions that are performed for a given segment.

EnablePMTUDiscovery

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: When this parameter is set to 1 (true) TCP attempts to discover the Maximum Transmission Unit (MTU), or largest packet size, over the path to a remote host. By discovering the Path MTU (PMTU) and limiting TCP segments to this size, TCP can eliminate fragmentation at routers along the path that connect networks with different MTUs. Fragmentation adversely affects TCP throughput and network congestion. Setting this parameter to 0 (not recommended) causes an MTU of 576 bytes to be used for all connections that are not to destinations on a locally attached subnet.

FFPControlFlags

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: If this parameter is set to 1, Fast Forwarding Path (FFP) is enabled. If it is set to 0, TCP/IP instructs all FFP-capable adapters not to do any fast forwarding on this computer. FFP-capable network adapters can receive routing information from the stack and forward subsequent packets in hardware without passing them up to the stack. FFP parameters are located in the TCP/IP registry key, but are actually placed there by the Routing and Remote Access service.

FFPFastForwardingCacheSize

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number of bytes

Valid Range: 0–0xFFFFFFFF

Default: 100,000 bytes

Description: This is the maximum amount of memory that a driver that supports fast forwarding path (FFP) can allocate for its fast-forwarding cache if it uses system memory for its cache. If the device has its own memory for fast-forwarding cache, this value is ignored.

GlobalMaxTcpWindowSize

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number of bytes

Valid Range: 0–0x3FFFFFFF (1073741823 decimal; however, values greater than 64 KB can only be achieved when connecting to other systems that support RFC 1323 window scaling, which is discussed in the TCP section of this paper.)

Default: This parameter does not exist by default.

Description: The *TcpWindowSize* parameter can be used to set the receive window on a per-interface basis. This parameter can be used to set a global limit for the TCP window size on a system-wide basis.

IPAutoconfigurationAddress

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—String

Valid Range: A valid IP address

Default: None

Description: The DHCP client stores the IP address chosen using APIPA autoconfiguration here. This value should not be altered.

IPAutoconfigurationEnabled

Key: Tcpip\Parameters, Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter enables or disables IP autoconfiguration using APIPA. See the “Automatic Client Configuration and Media Sense” section of this paper for details. This parameter can be set

globally or per interface. If a per-interface value is present, it overrides the global value for that interface.

IPAutoconfigurationMask

Key: Tcpip\Parameters, Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—String

Valid Range: A valid IP subnet mask

Default: 255.255.0.0

Description: This parameter controls the subnet mask assigned to the client using APIPA autoconfiguration. See the "Automatic Client Configuration and Media Sense" section of this paper for details. This parameter can be set globally or per interface. If a per-interface value is present, it overrides the global value for that interface.

IPAutoconfigurationSeed

Key: Tcpip\Parameters, Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Number

Valid Range: 0–0xFFFF

Default: 0

Description: This parameter is used internally by the DHCP client and should not be modified.

IPAutoconfigurationSubnet

Key: Tcpip\Parameters, Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—String

Valid Range: A valid IP subnet

Default: 169.254.0.0

Description: This parameter controls the initial network ID used by APIPA autoconfiguration to pick an IP address for the client. See the "Automatic Client Configuration and Media Sense" section of this paper for details. This parameter can be set globally or per interface. If a per-interface value is present, it overrides the global value for that interface.

IGMPLevel

Key: Tcpip\Parameters

Value Type: REG_DWORD—Number

Valid Range: 0,1,2

Default: 2

Description: This parameter determines to what extent the system supports IP multicasting and participates in the Internet Group Management Protocol. At level 0, the system provides no multicast support. At level 1, the system can send IP multicast packets but cannot receive them. At level 2, the system can send IP multicast packets and fully participate in IGMP to receive multicast packets.

IPEnableRouter

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Setting this parameter to 1 (true) causes the system to forward IP packets that have a destination address that is not assigned to it.

IPEnableRouterBackup

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Setup writes the previous value of *IPEnableRouter* to this key. It should not be adjusted manually.

KeepAliveInterval

Key: Tcpip\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 1–0xFFFFFFFF

Default: 1000 (one second)

Description: This parameter determines the interval between TCP keep-alive retransmissions until a response is received. Once a response is received, the delay until the next keep-alive transmission is again controlled by the value of *KeepAliveTime*. The connection is aborted after the number of retransmissions specified by *TcpMaxDataRetransmissions* have gone unanswered.

KeepAliveTime

Key: Tcpip\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 1–0xFFFFFFFF

Default: 7,200,000 (two hours)

Description: The parameter controls how often TCP attempts to verify that an idle connection is still intact by sending a keep-alive packet. If the remote system is still reachable and functioning, it acknowledges the keep-alive transmission. Keep-alive packets are not sent by default. This feature may be enabled on a connection by an application.

MaxForwardBufferMemory

Key: Tcpip\Parameters

Value Type: REG_DWORD—number of bytes

Valid Range: *network MTU*–0xFFFFFFFF

Default: 2097152 decimal (2 MB)

Description: This parameter limits the total amount of memory that IP can allocate to store packet data in the router packet queue. This value must be greater than or equal to the value of the *ForwardBufferMemory* parameter. See the description of *ForwardBufferMemory* for more details.

MaxForwardPending

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number of packets

Valid Range: 1–0xFFFFFFFF

Default: 0x1388 (5000 decimal)

Description: This parameter limits the number of packets that the IP forwarding engine can submit for transmission to a specific network interface at any time. Additional packets are queued in IP until outstanding transmissions on the interface complete. Most network adapters transmit packets very quickly, so the default value is sufficient. A single remote access interface, however, may multiplex many slow serial lines. Configuring a larger value for this type of interface may improve its performance. The appropriate value depends on the number of outgoing lines and their load characteristics.

MaxFreeTcbs

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFFFFFF

Default: The following default values are used (note that *small* is defined as a computer with less than19 MB of RAM, *medium* is 19–63 MB of RAM, and *large* is 64 MB or more of RAM. Although this code still exists, nearly all computers are *large* now).

For Windows Server 2003:

- Small system—500
- Medium system—1000
- Large system—2000

For Windows XP:

- Small system—250
- Medium system—500
- Large system—1000

Description: This parameter controls the number of cached (pre-allocated) Transport Control Blocks (TCBs) that are available. A TCB is a data structure that is maintained for each TCP connection.

MaxHashTableSize

Key: Tcpip\Parameters

Value Type: REG_DWORD—number (must be a power of 2)

Valid Range: 0x40–0x10000 (64-65536 decimal)

Default: 512

Description: This value should be set to a power of 2 (for example, 512, 1024, 2048, and so on.) If this value is not a power of 2, the system configures the hash table to the next power of 2 value (for example, a setting of 513 is rounded up to 1024.) This value controls how fast the system can find a TCB and should be increased if *MaxFreeTcbs* is increased from the default.

MaxICMPHostRoutes

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0x7FFFFFFE

Default: 0x3E8 (1000 in decimal) or 0x2710 (10000 in decimal) with the update to Windows Server 2003 SP1 available from Microsoft Knowledge Base article 898060.

Description: This value restricts the number of host routes that can be added to the local IP route table by receiving ICMP Redirect messages. You should not change this value unless the computer needs to be able to add a large number of host routes by receiving ICMP Redirect messages.

MaxNormLookupMemory

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: Any **DWORD** (0xFFFFFFFF means no limit on memory.)

Default: The following default values are used (Small is defined as a computer with less than 19 MB of RAM, Medium is 19–63 MB of RAM, and Large is 64 MB or more of RAM. Although this code still exists, nearly all computers are Large now).

For Windows Server 2003:

- Small system—150,000 bytes, which accommodates 1000 routes
- Medium system—1,500,000 bytes, which accommodates 10,000 routes
- Large system—5,000,000 bytes, which accommodates 40,000 routes

For Windows XP:

- 150,000 bytes, which accommodates 1000 routes

Description: This parameter controls the maximum amount of memory that the system allows for the route table data and the routes themselves. It is designed to prevent memory exhaustion on the computer caused by adding large numbers of routes.

MaxNumForwardPackets

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 1–0xFFFFFFFF

Default: 0xFFFFFFFF

Description: This parameter limits the total number of IP packet headers that can be allocated for the router packet queue. This value must be greater than or equal to the value of the *NumForwardPackets* parameter. See the description of *NumForwardPackets* for more details.

MaxUserPort

Key: Tcpip\Parameters

Value Type: REG_DWORD—maximum port number

Valid Range: 5000–65534 (decimal)

Default: 0x1388 (5000 decimal)

Description: This parameter controls the maximum port number used when an application requests any available user port from the system. Normally, short-lived ports are allocated in the range from 1024 through 5000. Setting this parameter to a value outside of the valid range causes the nearest valid value to be used (5000 or 65534).

MTU

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 88–*the MTU of the underlying network*

Default: 0xFFFFFFFF

Description: This parameter overrides the default Maximum Transmission Unit (MTU) for a network interface. The MTU is the maximum IP packet size, in bytes, that can be transmitted over the underlying network. For values larger than the default for the underlying network, the network default MTU is used. For values smaller than 88, the MTU of 88 is used.

Note: Windows Server 2003 TCP/IP uses PMTU detection by default and queries the NIC driver to find out what local MTU is supported. Altering the MTU parameter is generally not necessary and may result in reduced performance. See the "Path Maximum Transmission Unit (PMTU) Discovery" section of this paper for more details.

NumForwardPackets

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 1—*some reasonable value smaller than 0xFFFFFFFF*

Default: 0x32 (50 decimal)

Description: This parameter determines the number of IP packet headers that are allocated for the router packet queue. When all headers are in use, the system attempts to allocate more, up to the value configured for *MaxNumForwardPackets*. This value should be at least as large as the *ForwardBufferMemory* value divided by the maximum IP data size of the networks that are connected to the router. It should be no larger than the *ForwardBufferMemory* value divided by 256 because at least 256 bytes of forward buffer memory is used for each packet. The optimal number of forward packets for a given *ForwardBufferMemory* size depends on the type of traffic that is carried on the network and is somewhere between these two values. This parameter is ignored and no headers are allocated if routing is not enabled.

NumTcbTablePartitions

Key: Tcpip\Parameters

Value Type: REG_DWORD—number of TCB table partitions

Valid Range: 1–0xFFFF

Default: 4

Description: This parameter controls the number of TCB table partitions. The TCB table can be portioned to improve scalability on multi-processor systems by reducing contention on the TCB table. This value should not be modified without a careful performance study. A suggested maximum value is (number of CPUs)×2.

PerformRouterDiscovery

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD

Valid Range: 0, 1, 2

```
0 (disabled)
1 (enabled)
2 (enable only if DHCP sends the Perform Router Discovery option)
```

Default: 2, DHCP-controlled but off by default.

Description: This parameter controls whether Windows Server 2003 TCP/IP attempts to perform router discovery per RFC 1256 on a per-interface basis. See also *SolicitationAddressBcast*.

PerformRouterDiscoveryBackup

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: none

Description: This parameter is used internally to keep a back-up copy of the *PerformRouterDiscovery* value. It should not be modified.

PPTPTcpMaxDataRetransmissions

Key: Tcpip\Parameters

Value Type: REG_DWORD—number of times to retransmit a PPTP packet

Valid Range: 0–0xFF

Default: 5

Description: This parameter controls the number of times that a PPTP packet is retransmitted if it is not acknowledged. This parameter was added to allow retransmission of PPTP traffic to be configured separately from regular TCP traffic.

ReservedPorts

Key: Tcpip\Parameters

Value Type: REG_MULTI_SZ

Valid Range: *xxxx-yyyy* The string uses the format *xxxx-yyyy*. (port range)

Default: NA

Description: Allows ports to be reserved so that they are not used as part of the 1024 or greater range. This is useful for applications that want a specific port range.

SackOpts

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter controls whether or not Selective Acknowledgment (SACK) support, as specified in RFC 2018, is enabled. SACK is described in more detail in the "Transmission Control Protocol (TCP)" section of this paper.

SolicitationAddressBcast

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter can be used to configure Windows to send router discovery messages as broadcasts instead of multicasts, as described in RFC 1256. By default, if router discovery is enabled, router discovery solicitations are sent to the all-routers multicast group (224.0.0.2). See also *PerformRouterDiscovery*.

StrictARPUpdate

Key: Tcpip\Parameters

Value Type: REG_DWORD

Valid Range: 0-1

Default: 0

Description: Specifies whether TCP/IP in Windows Server 2003 SP1 will store in the ARP cache the MAC address of the last ARP reply received (StrictARPUpdate=0) or the MAC address of the first ARP reply received (StrictARPUpdate=1). With StrictARPUpdate set to 1, TCP/IP will not update the MAC address of an existing ARP cache entry if it receives additional unsolicited ARP replies.

SynAttackProtect

Key: Tcpip\Parameters

Value Type: REG_DWORD

Valid Range: 0, 1

0 (no SYN attack protection)
1 (reduced retransmission retries and delayed RCE [route cache entry] creation if the *TcpMaxHalfOpen* and *TcpMaxHalfOpenRetried* settings are satisfied and a delayed indication to Winsock is made.)

Note: When the system finds itself under attack the following options on any socket can no longer be enabled: scalable windows (RFC 1323) and per adapter configured TCP parameters (Initial RTT, window size). This is because when protection is functioning the route cache entry is not queried before the SYN-ACK is sent and the Winsock options are not available at this stage of the connection.

Default: 1 - enabled for Windows Server 2003 Service Pack 1, 0 -disabled for Windows Server 2003 with no service packs installed

Recommendation: 1

Description: SYN attack protection involves reducing the amount of retransmissions for the SYN-ACKS, which will reduce the time for which resources have to remain allocated. The allocation of route cache entry resources is delayed until a connection is made and the connection indication to AFD is delayed until the three-way handshake is completed. Note that the actions taken by the protection mechanism only occur if *TcpMaxHalfOpen* and *TcpMaxHalfOpenRetried* settings are exceeded.

Tcp1323Opts

Key: Tcpip\Parameters

Value Type: REG_DWORD—number (flags)

Valid Range: 0, 1, 2, 3

```
0 (disable RFC 1323 options)
1 (window scaling enabled only)
2 (timestamps enabled only)
3 (both options enabled)
```

Default: No value. The default behavior is as follows: do not use the Timestamp and Window Scale options when initiating TCP connections but use them if the TCP peer that is initiating communication includes them in the SYN segment.

Description: This parameter controls the use of RFC 1323 Timestamp and Window Scale TCP options. Explicit settings for timestamps and window scaling are manipulated with flag bits. Bit 0 controls window scaling, and bit 1 controls timestamps.

TcpAckFrequency

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 0–255

Default: 2

Description: Specifies the number of ACKs that will be outstanding before the delayed ACK timer is ignored. Microsoft does not recommend changing this value from the default without careful study of the environment.

TcpDelAckTicks

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 2–6

Default: 2

Description: Specifies the number of 100-millisecond intervals to use for the delayed-ACK timer on a per-interface basis. By default, the delayed-ACK timer is 200 milliseconds. If you set this value to 0 or 1, the delayed-ACK time is 200 milliseconds. Microsoft does not recommend changing this value from the default without careful study of the environment.

TcpInitialRTT

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFF

Default: 3

Description: This parameter controls the initial time-out in seconds used for a TCP connection request and initial data retransmission on a per-interface basis. Use caution when tuning with this parameter because exponential backoff is used. Setting this value to larger than 3 results in much longer time-outs to nonexistent addresses.

TcpMaxConnectResponseRetransmissions

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–255

Default: 2

Description: This parameter controls the number of times that a SYN-ACK is retransmitted in response to a connection request if the SYN is not acknowledged. If this value is greater than or equal to 2, the stack employs SYN attack protection internally. If this value is less than 2, the stack does not read the registry values at all for SYN attack protection. See also *SynAttackProtect*, *TCPMaxHalfOpen*, and *TCPMaxHalfOpenRetried*.

TcpMaxConnectRetransmissions

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–255 (decimal)

Default: 2

Description: This parameter determines the number of times that TCP retransmits a connect request (a SYN segment) before aborting the attempt. The retransmission time-out is doubled with each successive retransmission in a given connect attempt. The initial time-out is controlled by the *TcpInitialRtt* registry value.

TcpMaxDataRetransmissions

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFFFFFF

Default: 5

Description: This parameter controls the number of times that TCP retransmits an individual data segment (not connection request segments) before aborting the connection. The retransmission time-out is doubled with each successive retransmission on a connection. It is reset when responses resume. The Retransmission Timeout (RTO) value is dynamically adjusted, using the historical measured round-trip time (Smoothed Round Trip Time) on each connection. The starting RTO on a new connection is controlled by the *TcpInitialRtt* registry value.

TcpMaxDupAcks

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 1–3

Default: 2

Description: This parameter determines the number of duplicate ACKs that must be received for the same sequence number of sent data before fast retransmit is triggered to resend the segment that has been dropped in transit. This mechanism is described in more detail in the "Transmission Control Protocol (TCP)" section of this paper.

TcpMaxSendFree

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFF

Default: 5000

Description: This parameter controls the size limit of the TCP header table. On machines with large amounts of RAM increasing this setting can improve responsiveness during a SYN attack.

TcpNumConnections

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFFFE

Default: 0xFFFFFE

Description: This parameter limits the maximum number of connections that TCP can have open simultaneously.

TcpTimedWaitDelay

Key: Tcpip\Parameters

Value Type: REG_DWORD—time in seconds

Valid Range: 30-300 (decimal)

Default: 0xF0 (240 decimal)

Description: This parameter determines the length of time that a connection stays in the TIME_WAIT state when being closed. While a connection is in the TIME_WAIT state, the socket pair cannot be

reused. This is also known as the 2MSL state because the value should be twice the maximum segment lifetime on the network. See RFC 793 for further details.

TcpUseRFC1122UrgentPointer

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter determines whether TCP uses the RFC 1122 or RFC 793 specification for urgent data (used by BSD-derived systems). There are two ways to interpret the value of the Urgent Pointer field in the TCP header: RFC 793 defines the value as indicating the first byte of normal data, RFC 1122 defines the value as indicating the last byte of urgent data. These two interpretations are not interoperable. Windows Server 2003 TCP/IP defaults to the RFC 793 interpretation (BSD mode).

TcpWindowSize

Key: Tcpip\Parameters, Tcpip\Parameters\Interface*interfaceGUID*

Value Type: REG_DWORD—number of bytes

Valid Range: 0–0x3FFFFFFF (1073741823 decimal). In practice the TCP/IP stack will round the number set to the nearest multiple of maximum segment size (MSS). Values greater than 64 KB can be achieved only when connecting to other systems that support RFC 1323 Window Scaling, which is discussed in the "Transmission Control Protocol (TCP)" section of this paper.

Default: The smaller of the following values:

- 0xFFFF
- *GlobalMaxTcpWindowSize* (another registry parameter)
- The larger of four times the MSS
- 16384 rounded up to an even multiple of the MSS

The stack also tunes itself based on the media speed:

- Below 1 Mbps: 8 KB
- 1 Mbps – 100 Mbps: 17 KB
- Greater than 100 Mbps: 64 KB

The default can start at 17520 for Ethernet, but may shrink slightly when the connection is established to another computer that supports extended TCP header options, such as Selective Acknowledgements (SACK) and TCP Timestamps, because these options increase the size of the TCP header beyond the usual 20 bytes, leaving slightly less room for data.

Description: This parameter determines the maximum TCP receive window size offered. The receive window specifies the number of bytes that a sender can transmit without receiving an acknowledgment. In general, larger receive windows improve performance over high-delay, high-bandwidth networks. For greatest efficiency, the receive window should be an even multiple of the TCP Maximum Segment Size (MSS). This parameter is both a per-interface parameter and a global parameter, depending upon

where the registry key is located. If there is a value for a specific interface, that value overrides the system-wide value. See also *GobalMaxTcpWindowSize*.

TrFunctionalMcastAddress

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter determines whether IP multicasts are sent using the Token Ring functional address of 0xC0-00-00-04-00-00 (=1) or the MAC-level broadcast address of 0xFF-FF-FF-FF-FF-FF (=0). For more information, see RFC 1469.

TypeOfInterface

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD

Valid Range: 0, 1, 2, 3

Default: 0 (allow multicast and unicast)

Description: This parameter determines whether the interface gets routes plumbed for unicast, multicast, or both traffic types, and whether those traffic types can be forwarded. If it is set to 0, both unicast and multicast traffic are allowed. If it is set to 1, unicast traffic is disabled. If it is set to 2, multicast traffic is disabled. If it set to 3, both unicast and multicast traffic are disabled. Since this parameter affects forwarding and routes, it may still be possible for a local application to send multicasts out over an interface, if there are no other interfaces in the computer that are enabled for multicast, and a default route exists.

UseZeroBroadcast

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: If this parameter is set to 1 (true), IP will use the all 0s address for the limited broadcast address (0.0.0.0) instead of the all 1s address (255.255.255.255). Most systems use the all 1s broadcasts, but some systems derived from BSD implementations use the all 0s broadcasts. Systems that use different broadcasts do not interoperate well on the same network.

Parameters Configurable from the User Interface

The following parameters are created and modified automatically by configuring TCP/IP properties from Network Connections. There should be no need to configure them directly in the registry.

DefaultGateway

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of dotted decimal IP addresses

Valid Range: Any set of valid IP addresses

Default: None

Description: This parameter specifies the list of gateways to be used to route packets that are not destined for a subnet that the computer is directly connected to, and for which a more specific route does not exist. This parameter, if it has a valid value, overrides the *DhcpDefaultGateway* parameter. There is only one active default gateway for the computer at any time, so adding multiple addresses is only done for redundancy. See the "Dead Gateway Detection" section in this paper for details.

Domain

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—character string

Valid Range: Any valid DNS domain name

Default: None

Description: This parameter specifies the DNS domain name of the interface. In Windows Server 2003, this and *NameServer* are per-interface parameters, rather than system-wide parameters. This parameter overrides the *DhcpDomain* parameter (filled in by the DHCP client), if it exists.

EnableDhcp

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: If this parameter is set to 1 (true), the DHCP client service attempts to use DHCP to configure the first IP interface on this adapter.

EnableSecurityFilters

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: If this parameter is set to 1 (true), IP security filters are enabled. See *TcpAllowedPorts*, *UdpAllowedPorts*, and *RawIPAllowedPorts*. To configure these values, on the **Start** menu, point to **Settings**, then click **Network Connections**, right-click **Local Area Connection**, and then click **Properties**. Select **Internet Protocol (TCP/IP)**, and click **Properties**, then click **Advanced**. Click the **Options** tab, select **TCP/IP filtering**, and click **Properties**.

Hostname

Key: Tcpip\Parameters

Value Type: REG_SZ—character string

Valid Range: Any valid DNS hostname

Default: The computer name of the system

Description: This parameter specifies the DNS host name of the system, which is returned by the **hostname** command.

IPAddress

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of dotted-decimal IP addresses

Valid Range: Any set of valid IP addresses

Default: None

Description: This parameter specifies the IP addresses of the IP interfaces to be bound to the adapter. If the first address in the list is 0.0.0.0, the primary interface on the adapter is configured from DHCP. A system with more than one IP interface for an adapter is *logically multihomed*. There must be a valid subnet mask value in the *SubnetMask* parameter for each IP address that is specified in this parameter. To add parameters with **Regedit.exe**, select this key and type the list of IP addresses, pressing **Enter** after each one. Then modify the *SubnetMask* value, and type a corresponding list of subnet masks.

NameServer

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—a space delimited list of dotted decimal IP addresses

Valid Range: Any set of valid IP address

Default: None (blank)

Description: This parameter specifies the DNS name servers that Windows Sockets queries to resolve names. In Windows Server 2003, this and the *DomainName* are per-interface settings.

RawIpAllowedProtocols

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of IP protocol numbers

Valid Range: Any set of valid IP protocol numbers

Default: None

Description: This parameter specifies the list of IP protocol numbers for which incoming datagrams are accepted on an IP interface when security filtering is enabled (*EnableSecurityFilters* = 1). The parameter controls the acceptance of IP datagrams by the raw IP transport, which is used to provide raw sockets. It does not control IP datagrams that are passed to other transports (for example, TCP). An empty list indicates that no values are acceptable. A single value of 0 indicates that all values are acceptable. The behavior of a list containing the value 0 mixed with other, nonzero values is undefined. If this parameter is missing from an interface, all values are acceptable. This parameter applies to all IP interfaces that are configured on a specific adapter.

SearchList

Key: Tcpip\Parameters

Value Type: REG_SZ—space delimited list of DNS domain name suffixes

Valid Range: 1-50

Default: None

Description: This parameter specifies a list of domain name suffixes to append to a name to be resolved through DNS if resolution of the unadorned name fails. By default, only the value of the *Domain* parameter is appended. This parameter is used by the Windows Sockets interface. See also the *AllowUnqualifiedQuery* parameter.

SubnetMask

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of dotted decimal subnet masks

Valid Range: Any set of valid subnet masks.

Default: None

Description: This parameter specifies the subnet masks to be used with the IP interfaces bound to the adapter. If the first mask in the list is 0.0.0.0, the primary interface on the adapter is configured using DHCP. There must be a valid subnet mask value in this parameter for each IP address specified in the *IPAddress* parameter.

TcpAllowedPorts

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of TCP port numbers

Valid Range: Any set of valid TCP port numbers

Default: None

Description: This parameter specifies the list of TCP port numbers for which incoming SYNs are accepted on an IP interface when security filtering is enabled (*EnableSecurityFilters* = 1). An empty list indicates that no values are acceptable. A single value of 0 indicates that all values are acceptable. The behavior of a list containing the value 0 mixed with other, nonzero values is undefined. If this parameter is missing from an interface, all values are acceptable. This parameter applies to all IP interfaces configured on a specified adapter.

UdpAllowedPorts

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of UDP port numbers

Valid Range: Any set of valid UDP port numbers

Default: None

Description: This parameter specifies the list of UDP port numbers for which incoming datagrams are accepted on an IP interface when security filtering is enabled (*EnableSecurityFilters* = 1). An empty list indicates that no values are acceptable. A single value of 0 indicates that all values are acceptable. The behavior of a list containing the value 0 mixed with other, nonzero values is undefined. If this parameter is missing from an interface, all values are acceptable. This parameter applies to all IP interfaces configured on a specified adapter.

Parameters Configurable Using the Route Command

The **route** command can store persistent IP routes as values under the Tcpip\Parameters\PersistentRoutes registry key. Each route is stored in the value name string as a comma-delimited list of the form:

```
destination,subnet mask,gateway,metric
```

For example, the command:

```
route add 10.99.100.0 MASK 255.255.255.0 10.99.99.1 METRIC 1 /p
```

produces the registry value:

```
10.99.100.0,255.255.255.0,10.99.99.1,1
```

The value type is a REG_SZ. There is no value data (empty string). Addition and deletion of these values can be accomplished using the route command. There should be no need to configure them directly.

Non-Configurable Parameters

The following parameters are created and used internally by the TCP/IP components. They should never be modified using the Registry Editor. They are listed here for reference only.

DhcpDefaultGateway

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—list of dotted decimal IP addresses

Valid Range: Any set of valid IP addresses

Default: None

Description: This parameter specifies the list of default gateways to be used to route packets that are not destined for a subnet to which the computer is directly connected and for which a more specific route does not exist. This parameter is written by the DHCP client service, if enabled. This parameter is overridden by a valid *DefaultGateway* parameter value. Although this parameter is set on a per-interface basis, there is always only one default gateway active for the computer. Additional entries are treated as alternatives if the first one is down.

DhcpIPAddress

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—dotted decimal IP address

Valid Range: Any valid IP address

Default: None

Description: This parameter specifies the DHCP-configured IP address for the interface. If the *IPAddress* parameter contains a first value other than 0.0.0.0, that value overrides this parameter.

DhcpDomain

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—Character string

Valid Range: Any valid DNS domain name

Default: None (provided by DHCP server)

Description: This parameter specifies the DNS domain name of the interface. In Windows Server 2003, this and *NameServer* are now per-interface parameters, rather than system-wide parameters. If the *Domain* key exists, it overrides the *DhcpDomain* value.

DhcpNameServer

Key: Tcpip\Parameters

Value Type: REG_SZ—A space delimited list of dotted decimal IP addresses

Valid Range: Any set of valid IP address

Default: None

Description: This parameter specifies the DNS name servers to be queried by Windows Sockets to resolve names. It is written by the DHCP client service, if enabled. If the *NameServer* parameter has a valid value, it overrides this parameter.

DhcpServer

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—dotted decimal IP address

Valid Range: Any valid IP address

Default: None

Description: This parameter specifies the IP address of the DHCP server that granted the lease on the IP address in the *DhcpIPAddress* parameter.

DhcpSubnetMask

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—dotted decimal IP subnet mask

Valid Range: Any subnet mask that is valid for the configured IP address

Default: None

Description: This parameter specifies the DHCP-configured subnet mask for the address specified in the *DhcpIPAddress* parameter.

DhcpSubnetMaskOpt

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_SZ—dotted decimal IP subnet mask

Valid Range: Any subnet mask that is valid for the configured IP address

Default: None

Description: This parameter is filled in by the DHCP client service and is used to build the *DhcpSubnetMask* parameter, which the stack actually uses. Validity checks are performed before the value is inserted into the *DhcpSubnetMask* parameter.

Lease

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—time in seconds

Valid Range: 1–0xFFFFFFFF

Default: None

Description: The DHCP client service uses this parameter to store the time, in seconds, for which the lease on the IP address for this adapter is valid.

LeaseObtainedTime

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—absolute time, in seconds, since midnight of 1/1/70

Valid Range: 1–0xFFFFFFFF

Default: None

Description: The DHCP client service uses this parameter to store the time at which the lease on the IP address for this adapter was obtained.

LeaseTerminatesTime

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—absolute time, in seconds, since midnight of 1/1/70

Valid Range: 1–0xFFFFFFFF

Default: None

Description: The DHCP client service uses this parameter to store the time at which the lease on the IP address for this adapter expires.

LLInterface

Key: Tcpip\Parameters\Adapters*interfaceGUID*

Value Type: REG_SZ—Windows Server 2003 device name

Valid Range: A legal Windows Server 2003 device name

Default: Empty string (blank)

Description: This parameter is used to direct IP to bind to a different link-layer protocol than the built-in ARP module. The value of the parameter is the name of the Windows Server 2003 device to which IP should bind. This parameter is used in conjunction with the RAS component, for example. It is only present when ARP modules other than LAN bind to IP.

NTEContextList

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—number

Valid Range: 0–0xFFFF

Default: none

Description: This parameter identifies the context of the IP address associated with an interface. Each IP address associated with an interface has its own context number. The values are used internally to identify an IP address and should not be altered.

T1

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—absolute time, in seconds, since midnight of 1/1/70

Valid Range: 1–0xFFFFFFFF

Default: None

Description: The DHCP client service uses this parameter to store the time at which the service first tries to renew the lease on the IP address for the adapter by contacting the server that granted the lease.

T2

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—absolute time, in seconds, since midnight of 1/1/70

Valid Range: 1–0xFFFFFFFF

Default: None

Description: The DHCP client service uses this parameter to store the time at which the service tries to renew the lease on the IP address for the adapter by broadcasting a renewal request. Time *T2* should only be reached if the service is unable to renew the lease with the original server for some reason.

Appendix B:
NetBIOS over TCP/IP Configuration Parameters

All of the NetBIOS over TCP/IP (NetBT) parameters are registry values located under one of two different subkeys of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:

- NetBT\Parameters
- NetBT\Adapters\Interfaces*interface*, in which *interface* refers to the subkey for a network interface to which NetBT is bound

Values under the latter key(s) are specific to each interface. If the system is configured using DHCP, a change in parameters takes effect if you issue the command **ipconfig /renew** from a command prompt. Otherwise, you must reboot the system for a change in any of these parameters to take effect.

Parameters Configurable Using the Registry Editor

The following parameters are installed with default values by Control Panel-Network Connections during the installation of the TCP/IP components. They may be modified using the Registry Editor (**Regedit.exe**). A few of the parameters are visible in the registry by default, but most must be created in order to modify the default behavior of the NetBT driver.

BacklogIncrement

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 3–0x14 (1–20 decimal)

Default: 3

Description: This parameter was added in response to Internet SYN-ATTACK issues. When a connection attempt is made to the NetBIOS TCP port (139), if the number of free connection blocks is below 2, a *BackLogIncrement* number of new connection blocks are created by the system. Each connection block consumes 78 bytes of memory. A limit on the total number of connection blocks allowed can be set using the *MaxConnBackLog* parameter. One connection block is required for each NetBT connection.

BcastNameQueryCount

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 1–0xFFFF

Default: 3

Description: This value determines the number of times NetBT broadcasts a query for a specific name without receiving a response.

BcastQueryTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 100–0xFFFFFFFF

Default: 0x2ee (750 decimal)

Description: This value determines the time interval between successive broadcast name queries for the same name.

BroadcastAddress

Key: Netbt\Parameters

Value Type: REG_DWORD—4-byte, little-endian encoded IP address

Valid Range: 0–0xFFFFFFFF

Default: The 1s-broadcast address for each network

Description: This parameter can be used to force NetBT to use a specific address for all broadcast name-related packets. By default, NetBT uses the 1s-broadcast address appropriate for each net (that is, for a network of 10.101.0.0 with a subnet mask of 255.255.0.0, the subnet broadcast address would be 10.101.255.255). This parameter would be set, for example, if the network uses the 0s-broadcast address (set using the *UseZeroBroadcast* TCP/IP parameter). The appropriate subnet broadcast address would then be 10.101.0.0 in the example above. This parameter would then be set to 0x0b650000. This parameter is global and is used on all subnets to which NetBT is bound.

CachePerAdapterEnabled

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This value determines whether NetBIOS remote name caching is done on a per-adapter basis. Nbtstat -c has been enhanced to show the per-adapter name cache.

CacheTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD—time, in milliseconds

Valid Range: 0xEA60–0xFFFFFFFF

Default: 0x927c0 (600000 milliseconds = 10 minutes)

Description: This value determines the time interval that names are cached in the remote name table. The **nbtstat –c** command can be used to view the remaining time for each name in the cache.

ConnectOnRequestedInterfaceOnly

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This value can be used to allow NetBT connections on the requested adapter only. When the redirector on a multihomed computer calls another *computername*, it places calls on all NetBT transports (protocol/adapter combinations) to which it is bound. Each transport independently attempts to reach the target name. Setting this parameter limits each transport to connecting to other computers that are reachable via its own adapter, preventing crossover traffic. For more details, see the "NetBIOS Name Registration and Resolution for Multihomed Computers" section of this paper.

It no longer works no wonder it doesn't make sense.

EnableDns

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: If this value is set to 1 (true), NetBT queries the DNS server for names that cannot be resolved by WINS, broadcast, or the Lmhosts file.

EnableProxyRegCheck

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: If this parameter is set to 1 (true), the proxy name server sends a negative response to a broadcast name registration if the name is already registered with WINS or is in the proxy's local name cache with a different IP address. This feature prevents a system from changing its IP address as long as WINS has a mapping for the name. For this reason, it is disabled by default.

InitialRefreshT.O.

Key: Netbt\Parameters

Value Type: REG_DWORD—time, in milliseconds

Valid Range: 960000–0xFFFFFFF

Default: 960000 (16 minutes)

Description: This parameter specifies the initial refresh time-out used by NetBT during name registration. NetBT tries to contact the WINS servers at one-eighth of this time interval when it is first registering names. When it receives a successful registration response, that response contains the new refresh interval to use.

LmhostsTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 1000–0xFFFFFFFF

Default: 6000 (6 seconds)

Description: This parameter specifies the time-out value for Lmhosts and DNS name queries submitted by NetBT. The timer has a granularity of the time-out value, so the actual time-out could be as much as twice the value.

MaxConnBackLog

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 2–0x9c40 (1-40,000 decimal)

Default: 1000

Description: This value determines the maximum number of connection blocks that NetBT allocates. See the *BackLogIncrement* parameter for more details.

MaxPreloadEntries

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 0x3E8–0x7D0 (1000–2000 decimal)

Default: 1000 decimal

Description: This value determines the maximum number of entries that are preloaded from the Lmhosts file. Entries to preload into the cache are flagged in the Lmhosts file with the #PRE tag.

MaxDgramBuffering

Key: Netbt\Parameters

Value Type: REG_DWORD—number of bytes

Valid Range: 0x20000–0xFFFFFFFF

Default: 0x20000 (128K)

Description: This parameter specifies the maximum amount of memory that NetBT dynamically allocates for all outstanding datagram sends. Once this limit is reached, further sends fail due to insufficient resources.

MinimumRefreshSleepTime

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 21600000-4294967295

Default: 21600000 ms (6 hours)

Description: This parameter is used to reset the TTL on the WakeupTimer if ½ of the TTL is less than 6 hours when the machine is put into sleep or hiberate mode.

MinimumFreeLowerConnections

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 20-500

Default: 50

Description: This parameter is used allocate the number of free handles that the system has upon boot to accept incoming connections. These handles are allocated in addition to the number of active connections that are being serviced. Once the machine is in a steady state the number of free handles increases to ½ the value of the used handles. The number of free handles is never less than 50 unless specified in the registry.

NameServerPort

Key: Netbt\Parameters

Value Type: REG_DWORD—UDP port number

Valid Range: 0–0xFFFF

Default: 0x89

Description: This parameter determines the destination port number to which NetBT sends name service-related packets, such as name queries and name registrations, to WINS. The Microsoft WINS Server listens on port 0x89 (138 decimal). NetBIOS name servers from other vendors may listen on different ports.

NameSrvQueryCount

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 0–0xFFFF

Default: 3

Description: This value determines the number of times that NetBT sends a query to a WINS server for a specified name without receiving a response.

NameSrvQueryTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 100–0xFFFFFFFF

Default: 1500 (1.5 seconds)

Description: This value determines the time interval between successive name queries to WINS for a specified name.

NodeType

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 1, 2, 4, 8 (b-node, p-node, m-node, h-node)

Default: 1 or 8 based on the WINS server configuration

Description: This parameter determines what methods NetBT uses to register and resolve names. A b-node system uses broadcasts. A p-node system uses only point-to-point name queries to a name server (WINS). An m-node system broadcasts first, then queries the name server. An h-node system queries the name server first, then broadcasts. Resolution through Lmhosts and DNS, if enabled, follows these methods. If this key is present, it overrides the DhcpNodeType key. If neither key is present, the system defaults to b-node if there are no WINS servers configured for the client. The system defaults to h-node if there is at least one WINS server configured.

NoNameReleaseOnDemand

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter determines whether the computer releases its NetBIOS name when it receives a name-release request from the network. It was added to allow the administrator to protect the machine against malicious name-release attacks.

RandomAdapter

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter applies to a multihomed host only. If it is set to 1 (true), NetBT randomly chooses the IP address to put in a name-query response from all of its bound interfaces. Usually, the response contains the address of the interface to which the query arrived. This feature would be used for load balancing by a server with two interfaces on the same network.

RefreshOpCode

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 8, 9

Default: 8

Description: This parameter forces NetBT to use a specific opcode field in name-refresh packets. The specification for the NetBT protocol is somewhat ambiguous in this area. Although the default of 8 that is used by Microsoft implementations appears to be the intended value, some other implementations, such as those by Ungermann-Bass, use the value 9. Two implementations must use the same opcode field to interoperate.

ScopeId

Key: Netbt\Parameters

Value Type: REG_SZ—character string

Valid Range: Any valid DNS domain name consisting of two dot-separated parts or an asterisk (*).

Default: None

Description: This parameter specifies the NetBIOS name scope for the node. This value must not begin with a period. If this parameter contains a valid value, it overrides the DHCP parameter of the same name. A blank value (empty string) is ignored. Setting this parameter to the value "*" indicates a null scope and overrides the DHCP parameter.

SessionKeepAlive

Key: Netbt\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 60,000–0xFFFFFFFF

Default: 3,600,000 (1 hour)

Description: This value determines the time interval between keep-alive transmissions on a session. Setting the value to 0xFFFFFFF disables keep-alives.

SingleResponse

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter applies to a multihomed host only. If this parameter is set to 1 (true), NetBT supplies only the IP address from one of its bound interfaces in name-query responses. By default, the addresses of all bound interfaces are included.

Size/Small/Medium/Large

Key: Netbt\Parameters

Value Type: REG_DWORD

Valid Range: 1, 2, 3 (small, medium, large)

Default: 1 (small)

Description: This value determines the size of the name tables that are used to store local and remote names. In general, a setting of 1 (small) is adequate. If the system is acting as a proxy name server, the value is automatically set to 3 (large) to increase the size of the name cache hash table. Hash table buckets are sized as follows:

- Small: 16
- Medium: 128
- Large: 256

SMBDeviceEnabled

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: Windows Server 2003 supports a new network transport known as the SMB Device, which is enabled by default. This parameter can be used to disable the SMB device for troubleshooting purposes. See the "NetBT Internet/DNS Enhancements and the SMB Device" section of this paper for more details.

TryAllNameServers

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter controls whether the client continues to query additional name servers from the list of configured servers when a NetBIOS session setup request to one of the IP addresses fails. If this parameter is enabled, attempts are made to query all the WINS servers in the list and connect to all the IP addresses supplied before failing the request to the user.

TryAllIPAddrs

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: When a WINS server returns a list of IP addresses in response to a name query, they are sorted into a preference order based on whether any of them are on the same subnet as an interface belonging to the client. This parameter controls whether the client pings the IP addresses in the list and attempts to connect to the first one that responds, or whether it tries to connect to the first IP address in the (sorted) list and fails if that connection attempt fails. By default, the client pings each address in the list and attempts to connect to the first one that answers the ping.

UseDnsOnlyForNameResolutions

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter is used to disable all NetBIOS name queries. NetBIOS name registrations and refreshes are still used, and NetBIOS sessions are still allowed. To completely disable NetBIOS on an interface, see the *NetbiosOptions* parameter.

WinsDownTimeout

Key: Netbt\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 1000–0xFFFFFFFF

Default: 15,000 (15 seconds)

Description: This parameter determines the amount of time that NetBT waits before trying to use WINS again after it fails to contact any WINS server. This feature primarily allows computers that are temporarily disconnected from the network, such as laptops, to proceed through boot processing without waiting to time out each WINS name registration or query individually.

Parameters Configurable from the Connections UI

The following parameters can be set using the Control Panel-Network Connections. There should be no need to configure them directly.

EnableLmhosts

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: If this value is set to 1 (true), NetBT searches the Lmhosts file, if it exists, for names that cannot be resolved by WINS or broadcast. By default, there is no Lmhosts file database directory (specified by Tcpip\Parameters\DatabasePath), so no action is taken. This value is written by the Advanced TCP/IP Configuration dialog box.

EnableProxy

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: If this value is set to 1 (true), the system acts as a proxy name server for the networks to which NetBT is bound. A proxy name server answers broadcast queries for names that it has resolved through WINS. A proxy name server allows a network of b-node implementations to connect to servers on other subnets that are registered with WINS.

NameServerList

Key: Netbt\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—space separated, dotted decimal IP address (that is, 10.101.1.200)

Valid Range: any list of valid WINS server IP addresses.

Default: blank (no address)

Description: This parameter specifies the IP addresses of the list of WINS servers configured for the computer. If this parameter contains a valid value, it overrides the DHCP parameter of the same name. This parameter replaces the Windows NT 4.0 parameters *NameServer* and *NameServerBackup*, which are no longer used.

NetbiosOptions

Key: Netbt\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 1, 2

Default: 1

Description: This parameter controls whether NetBIOS is enabled on a per-interface basis. On the **Start** menu, point to **Settings**, and click **Network Connections**. Right-click **Local Area Connection**, and click **Properties**. Select **Internet Protocol (TCP\IP)**, and click **Properties**, then click **Advanced**. Click the **WINS** tab. The NetBIOS options are **Enable NetBIOS over TCP\IP**, **Disable NetBIOS over TCP\IP**, or **Use NetBIOS setting from the DHCP server**, the default. When enabled, the value is 1. When disabled, the value is set to 2. If this key does not exist, the *DHCPNetbiosOptions* key is read. If this key does exist, *DHCPNetbiosOptions* is ignored.

Non-Configurable Parameters

The following parameters are created and used internally by the NetBT components. They should never be modified using the Registry Edit or it can cause the component to become unstable. They are listed here for reference only.

DHCPNameServerList

Key: Netbt\Parameters\Interfaces*interfaceGUID*

Value Type: REG_MULTI_SZ—space separated, dotted decimal IP address (that is, 10.101.1.200)

Valid Range: any list of valid WINS server IP addresses.

Default: blank (no address)

Description: This parameter specifies the IP addresses of the list of WINS servers, as provided by the DHCP service. This parameter replaces the Windows NT 4.0 parameters *DHCPNameServer* and *DHCPNameServerBackup*, which are no longer used. See also *NameServerList*, which overrides this parameter if it is present.

DHCPNetbiosOptions

Key: Netbt\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD—number

Valid Range: 1, 2

Default: 1

Description: This parameter is written by the DHCP client service. See the *NetbiosOptions* parameter for a description.

DhcpNodeType

Key: Netbt\Parameters

Value Type: REG_DWORD—number

Valid Range: 1–8

Default: 1

Description: This parameter specifies the NetBT node type. It is written by the DHCP client service, if enabled. A valid NodeType value overrides this parameter. See the entry for *NodeType* for a complete description.

DhcpScopeId

Key: Netbt\Parameters

Value Type: REG_SZ—character string

Valid Range: a dot-separated name string such as microsoft.com

Default: none

Description: This parameter specifies the NetBIOS name scope for the node. It is written by the DHCP client service, if enabled. This value must not begin with a period. See the entry for *ScopeId* for more information.

NbProvider

Key: Netbt\Parameters

Value Type: REG_SZ—character string

Valid Range: _tcp

Default: _tcp

Description: This parameter is used internally by the RPC component. The default value should not be changed.

TransportBindName

Key: Netbt\Parameters

Value Type: REG_SZ—character string

Valid Range: N/A

Default: \Device\

Description: This parameter is used internally during product development. The default value should not be changed.

Appendix C:
Windows Sockets and DNS Registry Parameters

AFD Registry Parameters

Afd.sys is the kernel-mode driver that is used to support Windows Sockets applications. When there are three default values, the default is calculated based on the amount of memory detected in the system:

- The first value is the default for smaller computers (less than 19 MB).
- The second value is the default for medium computers (<32 MB on Windows XP Professional, <64 MB on Windows Server 2003).
- The third value is the default for large computers (>32 MB on Windows XP Professional, >64 MB on Windows Server 2003).

For example, if the default is given as 0/2/10, a system containing 12.5 to 20 MB of RAM would default to 2.

The following values can be set under:

```
HKEY_LOCAL_MACHINE
     \SYSTEM
          \CurrentControlSet
               \Services
                    \Afd
                         \Parameters
```

DefaultReceiveWindow

Value Type: REG_DWORD

Default: 4096/8192/8192

Description: The number of receive bytes that AFD buffers on a connection before imposing flow control. For some applications, a larger value here gives slightly better performance at the expense of increased resource utilization. Applications can modify this value on a per-socket basis with the SO_RCVBUF socket option.

DefaultSendWindow

Value Type: REG_DWORD

Default: 4096/8192/8192

Description: This is similar to *DefaultReceiveWindow*, but for the send side of connections.

DisableAddressSharing

Value Type: REG_DWORD

Default: 0

Range: 0, 1

Description: This parameter is used to prevent address sharing (SO_REUSEADDR) between processes so that if a process opens a socket, no other process can steal data from it. A similar effect

can be achieved if an application uses the new socket option SO_EXCLUSIVEADDRUSE. This setting allows administrators to secure older applications that are not aware of this option.

FastCopyReceiveThreshold

Value Type: REG_DWORD

Default: 1024

Description: When an application posts a receive with a buffer that is smaller than the current packet being buffered by Winsock, AFD can either make an additional copy of the packet and then copy data to the application buffers directly (which is a two-stage copy because application buffers cannot be accessed directly under the lock), or it can lock and map application buffers and copy data once. This value represents a compromise between extra code execution for data copying, and extra code execution in the I/O subsystem and memory manager. The default value was found, by testing, to be the best overall value for performance. Changing this value is not generally recommended.

FastSendDatagramThreshold

Value Type: REG_DWORD

Default: 1024

Description: Datagrams smaller than the value of this parameter go through the fast I/O path or are buffered on send. Larger ones are held until the datagram is actually sent. The default value was found by testing to be the best overall value for performance. Fast I/O means copying data and bypassing the I/O subsystem, instead of mapping memory and going through the I/O subsystem. This is advantageous for small amounts of data. Changing this value is not generally recommended.

IgnorePushBitOnReceives

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: Normally, Windows Server 2003 completes a Windows Sockets **Receive** when one of the following occurs:

- Data arrives with the push bit set.
- The user **recv** buffer is full.
- 0.5 seconds have elapsed since any data arrived.

Setting this parameter to a 1 causes Afd.sys to treat all incoming packets as though the push bit was set. This should only be done when necessary to work around client TCP/IP implementations that are not properly pushing data.

LargeBufferSize

Value Type: REG_DWORD

Default: PAGE_SIZE (4096 bytes on i386, 8192 bytes on Alpha)

Description: The size, in bytes, of large buffers used by AFD. Smaller values use less memory and larger values can improve performance.

LargeBufferListDepth

Value Type: REG_DWORD

Default: 0/2/10

Description: Depth of large buffer look-aside list.

MaxFastTransmit

Value Type: REG_DWORD

Valid Range: 0–0xffffffff

Default: 64 KB

Description: This parameter controls the maximum amount of data that is transferred in a **TransmitFile** request on the fast path. Fast I/O is essentially copying data and bypassing the I/O subsystem, instead of mapping memory and going through the I/O subsystem. This is advantageous for small amounts of data. Changing this value is not generally recommended.

MaxFastCopyTransmit

Value Type: REG_DWORD

Valid Range: 0–0xFFFFFFFF

Default: 128

Description: This parameter controls the maximum size of data that uses copy instead of cached memory on the fast-path. Fast I/O is essentially copying data and bypassing the I/O subsystem, instead of mapping memory and going through the I/O subsystem. This is advantageous for small amounts of data. Changing this value is not generally recommended.

MediumBufferSize

Value Type: REG_DWORD

Default: 1504

Description: The size, in bytes, of medium buffers used by AFD.

MediumBufferListDepth

Value Type: REG_DWORD

Default: 4/8/24

Description: Depth of medium buffer look-aside list.

OverheadChargeGranularity

Value Type: REG_DWORD

Default: 1 page

Valid Range: A power of 2

Description: This parameter determines in what increments overhead is actually charged. The default is one page, and the intention is to properly charge and contain attacker type applications that try to run the system out of memory.

PriorityBoost

Value Type: REG_DWORD

Default: 2

Valid Range: 0–16

Description: The priority boost that AFD gives to a thread when it completes I/O for that thread. If a multithreaded application experiences starvation of some threads, the problem may be remedied by reducing this value.

SmallBufferListDepth

Value Type: REG_DWORD

Default: 8/16/32

Description: Depth of the small buffer look-aside list.

SmallBufferSize

Value Type: REG_DWORD

Default: 128

Description: The size in bytes of small buffers used by AFD.

StandardAddressLength

Value Type: REG_DWORD

Default: 22

Description: The length of TDI addresses that are typically used for the computer. When using an alternate transport protocol, such as TP4, which uses very long addresses, increasing this value results in a slight performance improvement.

TransmitIoLength

Value Type: REG_DWORD

Default: PAGE_SIZE/PAGE_SIZE*2/65536

Description: The default size for I/O (reads and sends) performed by TransmitFile(). For Windows XP Professional, the default I/O size is exactly one page.

TransmitWorker

Value Type: REG_DWORD

Default: 0x10

Valid Range: 0x10, 0x20

Description: This parameter controls how Afd.sys uses system threads. Setting it to 0x10 causes AFD to use system threads to perform IO that results from a long (more than 2 *SendPacketLength* worth of data) TransmitFile request. Setting it to 0x20 causes AFD to use kernel-mode APC for IO and to execute everything in the context of the same thread. This is new in Windows Server 2003 and can improve performance by reducing the number of context switches in long TransmitFile requests.

Dynamic DNS Registration Parameters

These parameters control behavior of the dynamic DNS registration client. If a parameter is not present, the default value listed is used.

DNSQueryTimeouts

Key: Tcpip\Parameters

Value Type: REG_MULTI_SZ—list of timeouts terminated by a zero

Valid Range: valid list of numbers

Default: 1 2 2 4 8 0 (format note: after entering each number hit return and terminate the list with zero)

Description: This parameter can be used to change the DNS query timeouts that the DNS client uses. In a controlled non-Internet or low-delay environment this could be used to decrease the time to failure of the query.

DefaultRegistrationTTL

Key: Tcpip\Parameters

Value Type: REG_DWORD—seconds

Default: 0x4B0 (1200 decimal, or 20 minutes)

Valid Range: 0–0xFFFFFFFF

Description: This parameter can be used to control the TTL value sent with dynamic DNS registrations.

EnableAdapterDomainNameRegistration

Key: Tcpip\Parameters\Interfaces*interface*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter can be used to enable DNS dynamic update registration of a specific adapter's domain name information. This setting is useful when registrations of the adapter address(es) under the adapter's domain name are needed. When this key is set to true and *DisableDynamicUpdate* is false, the given adapter's address(es) is registered under the specific adapter's domain name and under the system's primary domain name.

DisableDynamicUpdate

Key: Tcpip\Parameters, Tcpip\Parameters\Interfaces*interface*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false; dynamic DNS-enabled)

Description: This parameter can be used to completely disable DNS dynamic update registration. This parameter is both a per-interface parameter and a global parameter, depending upon where the registry key is located. If the value at the Tcpip\Parameters level is set to 1, dynamic update is disabled for the

entire system. If the value at the Tcpip\Parameters level is set to 0, dynamic updates can be disabled on a per-adapter basis.

DisableReplaceAddressesInConflicts

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter is used to turn off the address registration conflict rule that the last writer wins. By default, a computer does not replace any current records on the DNS server that do not appear to have been owned by it at one time.

DisableReverseAddressRegistrations

Key: Tcpip\Parameters\Interfaces*interface*

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false; registration of PTR records enabled)

Description: This parameter can be used to turn off DNS dynamic update reverse address (PTR) record registration. If the DHCP server that configures this computer is running Windows Server 2003, then it is capable of registering the PTR record with the DNS dynamic update protocol. However, if the DHCP server is not capable of performing DNS dynamic update PTR registrations and you do not want to register PTR records with the DNS dynamic update protocol, set this parameter to 1.

UpdateSecurityLevel

Key: Tcpip\Parameters

Value Type: REG_DWORD—flags

Default: 0

Valid Range: 0,0x00000010, 0x00000020, 0x00000100

Description: This parameter can be used to control the security that is used for DNS dynamic updates. It defaults to 0, to try nonsecure update, and if refused, to send Windows Server 2003 secure dynamic updates. Valid values are listed below:

- **0x00000000—**default, nonsecure updates
- **0x00000010—**security OFF
- **0x00000100—**secure ONLY ON

DNS Caching Resolver Service Registry Parameters

Windows Server 2003 includes a DNS caching resolver service. This service performs the function of caching DNR answers so that the DNS server does not need to be repeatedly queried for the same information. The service can be stopped using the Service Control Manager MMC snap-in. Registry parameters for this service are located under the **\System\CurrentControlSet\Services\Dnscache\Parameters** key.

AdapterTimeoutCacheTime

Value Type: REG_DWORD—seconds

Valid Range: 0–0xFFFFFFFF

Default: 300 (5 minutes)

Description: The amount of time that a particular adapter on a multihomed machine is disabled when a DNS query attempt fails (times out) for all of the given adapter's DNS servers. For instance, if you have two adapters and the DNS servers on one of the networks are unreachable, mark the adapter as unusable for this time period. (A Plug and Play event or cache time-out forces the resolver to retry this interface and mark it as disabled, if needed.)

CacheHashTableSize

Value Type: REG_DWORD—number

Default: 0xD3 (211 decimal)

Valid Range: Any prime number greater than 0

Description: This parameter can be used to control the maximum number of rows in the hash table used by the DNS caching resolver service. It should not be necessary to adjust this parameter.

CacheHashTableBucketSize

Value Type: REG_DWORD—number

Default: 0xa (10 decimal)

Range: 0–0x32 (50 decimal)

Description: This parameter can be used to control the maximum number of columns in the hash table used by the DNS caching resolver service. It should not be necessary to adjust this parameter.

DefaultRegistrationRefreshInterval

Value Type: REG_DWORD—time in seconds

Default: 0x15180 (86400 decimal, or 24 hours)

Range: 0–0xFFFFFFFF

Description: This parameter can be used to control the dynamic DNS registration refresh interval.

MaxCacheEntryTtlLimit

Value Type: REG_DWORD—time in seconds

Default: 0x15180 (86400 decimal)

Valid Range: 0–0xFFFFFFFF (suggested value less than one day, to prevent very stale records)

Description: This parameter can be used to control the maximum cache entry time-to-live (TTL) value. It overrides any value that may have been set on a specific record that is larger.

MaxSOACacheEntryTtlLimit

Value Type: REG_DWORD—time, in seconds

Valid Range: 0–0xFFFFFFFF

Default: 120 (2 minutes)

Description: The maximum number of seconds that the resolver cache caches any SOA records. This value overrides any TTL value greater than itself for a specific SOA record that is returned from a DNS query. SOA records are essential for dynamic updates; therefore, they are not cached for long, to ensure that the most up-to-date record data is available for the DNS start of authority.

NegativeCacheTime

Value Type: REG_DWORD—time, in seconds

Default: 0x12c (300 decimal, or 5 minutes)

Valid Range: 0–0xFFFFFFFF (the suggested value is less one day, to prevent very stale records)

Description: This parameter can be used to control the cache time for negative records.

NegativeSOACacheTime

Value Type: REG_DWORD—time, in seconds

Default: 0x78 (120 decimal, or 2 minutes)

Valid Range: 0–0xFFFFFFFF (the suggested value is less than five minutes)

Description: This parameter can be used to control the cache time for negative Start of Authority (SOA) records. DNS registrations that fail are retried at five and ten minutes, so if this value is set to five minutes or more, retries are answered negatively from cache, instead of from the server, which could be available.

NetFailureErrorPopupLimit

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Description: This parameter enables the UI popup to indicate that the DNS resolver was unable to query (reach) the configured DNS servers for a repeated number of query attempts.

NetFailureCacheTime

Value Type: REG_DWORD—time, in seconds

Default: 0x1e (30 decimal)

Valid Range: 0–0xFFFFFFFF (suggested value is less than five minutes)

Description: This parameter is used to control the general network failure cache time. It prevents the resolver from querying for a period of time when it has been detected that a time-out error is occurring for queries against all known DNS servers. This avoids slowness (caused by time-outs) when the network does not respond.

Name Resolution Parameters

The following list of parameters is used by the Domain Name Resolver service.

AllowUnqualifiedQuery

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0

Description: This parameter controls whether or not the Domain Name Resolver queries the Domain Name Server(s) with the host name, followed by a dot (.) only (an unqualified query). For example, if your computer is in mydomain.com and you ping *target*, by default the DNS is queried for *target.mydomain.com* only. When this parameter is set to 1, *target* is also queried.

DisjointNameSpace

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1

Description: This parameter instructs the DNR to treat each interface as a disjoint name space. On a multihomed computer, a query to the DNS server(s) that is/are configured for one interface may result in a name error. This parameter is used to instruct the resolver to try the query against the possible DNS servers that are configured for other interfaces before returning results.

PrioritizeRecordData

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1

Description: This parameter controls whether or not the Domain Name Resolver sorts the addresses that are returned in response to a query for a multihomed host. By default, the DNR sorts addresses that are on the same subnet as one of the interfaces in the querying computer to the top of the list. This is done to give preference to a common-subnet (non-routed) IP address, when possible.

QueryIpMatching

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0

Description: This parameter controls whether or not the IP address of the DNS server queried is matched to the IP address of the server that sent the DNS response. This can be used as a primitive security feature to ensure that the resolver is not being fooled by a random query response from some computer other than the intended DNS server.

UseDomainNameDevolution

Key: Tcpip\Parameters

Value Type: REG_DWORD—binary

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Description: This parameter can be used to disable domain name *devolution* for unqualified DNS queries. Devolution describes the process of attempting to locate a host in the DNS by first appending the domain suffix of the client to the host name, and then querying for the full string. If that query fails, one label is removed at a time, and the query is resubmitted. For example, if a user or application on the computer *mycomputer.support.microsoft.com* attempts to reach a host named *target*, the DNR by default tries *target.support.microsoft.com*, and *target.microsoft.com,* and possibly *target*, depending on the value of the *AllowUnqualifiedQuery* parameter.

Appendix D: Tuning TCP/IP Response to Attack

TCP/IP Security Settings

In addition to the settings that are listed above, the following keys can be altered to assist the system to deal more effectively with an attack. It is important to note that these recommendations by no means makes the system impervious to attack and only focuses on tuning the TCP/IP stack's response to an attack. The setting of these keys does not address any of the many other components on the system, which could be used to attack the system. As with any change to the registry, the administrator needs to fully understand how these changes affect the default function of the system and whether they are appropriate in their environment.

SynAttackProtect

Key: Tcpip\Parameters

Value Type: REG_DWORD

Valid Range: 0, 1

0 (no SYN attack protection)
1 (reduced retransmission retries and delayed RCE [route cache entry] creation if the *TcpMaxHalfOpen* and *TcpMaxHalfOpenRetried* settings are satisfied and a delayed indication to Winsock is made.)

Note: When the system finds itself under attack the following options on any socket can no longer be enabled: scalable windows (RFC 1323) and per adapter configured TCP parameters (Initial RTT, window size). This is because when protection is functioning the route cache entry is not queried before the SYN-ACK is sent and the Winsock options are not available at this stage of the connection.

Default: 1 - enabled for Windows Server 2003 Service Pack 1, 0 -disabled for Windows Server 2003 with no service packs installed

Recommendation: 1

Description: SYN attack protection involves reducing the amount of retransmissions for the SYN-ACKS, which will reduce the time for which resources have to remain allocated. The allocation of route cache entry resources is delayed until a connection is made and the connection indication to AFD is delayed until the three-way handshake is completed. Note that the actions taken by the protection mechanism only occur if *TcpMaxHalfOpen* and *TcpMaxHalfOpenRetried* settings are exceeded.

TcpMaxHalfOpen

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 100–0xFFFF

Default: 100 (Windows Server 2003, Standard Edition), 500 (Windows Server 2003, Enterprise Edition; Windows Server 2003, Datacenter Edition)

Description: This parameter controls the number of connections in the SYN-RCVD state allowed before SYN-ATTACK protection begins to operate. If SynAttackProtect is set to 1, ensure that this value is lower than the AFD listen backlog on the port that you want to protect (see backlog parameters in Appendix C for more information). See the *SynAttackProtect* parameter for more details.

TcpMaxHalfOpenRetried

Key: Tcpip\Parameters

Value Type: REG_DWORD—number

Valid Range: 80–0xFFFF

Default: 80 (Windows Server 2003, Standard Edition), 400 (Windows Server 2003, Enterprise Edition; Windows Server 2003, Datacenter Edition)

Description: This parameter controls the number of connections in the SYN-RCVD state for which there has been at least one retransmission of the SYN sent, before SYN-ATTACK attack protection begins to operate. See the *SynAttackProtect* parameter for more details.

EnablePMTUDiscovery

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Recommendation: 1

Description: When this parameter is set to 1 (true) TCP attempts to discover the Maximum Transmission Unit (MTU or largest packet size) over the path to a remote host. By discovering the Path MTU and limiting TCP segments to this size, TCP can eliminate fragmentation at routers along the path that connect networks with different MTUs. Fragmentation adversely affects TCP throughput and network congestion. Setting this parameter to 0 causes an MTU of 576 bytes to be used for all connections that are not to hosts on the local subnet.

NoNameReleaseOnDemand

Key: Netbt\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 0 (false)

Recommendation: 1

Description: This parameter determines whether the computer releases its NetBIOS name when it receives a name-release request from the network. It was added to allow the administrator to protect the machine against malicious name-release attacks.

EnableDeadGWDetect

Key: Tcpip\Parameters

Value Type: REG_DWORD—Boolean

Valid Range: 0, 1 (false, true)

Default: 1 (true)

Recommendation: 0

Description: When this parameter is set to 1, TCP is allowed to perform dead-gateway detection. With this feature enabled, TCP may ask IP to change to a backup gateway if a number of connections are experiencing difficulty. Backup gateways may be defined in the advanced properties of the TCP/IP protocol. See the "Dead Gateway Detection" section in this paper for details.

KeepAliveTime

Key: Tcpip\Parameters

Value Type: REG_DWORD—time in milliseconds

Valid Range: 1–0xFFFFFFFF

Default: 7,200,000 (two hours)

Recommendation: 300,000

Description: The parameter controls how often TCP attempts to verify that an idle connection is still intact by sending a keep-alive packet. If the remote system is still reachable and functioning, it acknowledges the keep-alive transmission. Keep-alive packets are not sent by default. This feature may be enabled on a connection by an application.

PerformRouterDiscovery

Key: Tcpip\Parameters\Interfaces*interfaceGUID*

Value Type: REG_DWORD

Valid Range: 0, 1, 2

```
0 (disabled)
1 (enabled)
2 (enable only if DHCP sends the router discover option)
```

Default: 2, DHCP-controlled but off by default.

Recommendation: 0

Description: This parameter controls whether Windows Server 2003 attempts to perform router discovery per RFC 1256 on a per-interface basis. See also *SolicitationAddressBcast*.

Appendix E: Format of the Daytime Service Response String

RFC 867 does not specify the format of the response sent by the daytime service. For Windows 2000 and Windows Server 2003, the daytime service sends an ASCII text string containing the current system time and date formatted in a single line as follows:

TimeString DateString

with a single ASCII space character separating the time (*TimeString*) and date (*DateString*) components. Each of these components is formatted according to the default format for the selected locale of the computer. For example, the default format for “English (United States)” is:

TimeString = h:mm:ss tt

DateString = m/dd/yyyy

in which tt is either "AM" or "PM".

Here is example output for the English (United States) locale:

```
10:23:16 AM 4/11/2003
```

The default formats for the date and time for any locale may be viewed or configured from Control Panel-Regional Options (for Windows 2000) or Control Panel-Regional and Language Options (for Windows Server 2003).

There is no ability to change the format of the daytime service ASCII text string.

Summary

The TCP/IP stack in Windows Server 2003 has extensive support for standard features, performance enhancements, and services. Windows Server 2003 TCP/IP consists of the following core protocols: Address Resolution Protocol (ARP), Internet Protocol (IP), Internet Protocol security (IPsec), Internet Group Management Protocol (IGMP), Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and NetBIOS over TCP/IP (NetBT). Windows TCP/IP applications typically use the Windows Sockets or NetBIOS APIs. Windows Server 2003 TCP/IP includes support for automatic client configuration, media sense, and DNS dynamic update and client-side caching.

For More Information

For the latest information about Windows Server 2003, see the Windows Server 2003 Web site at http://www.microsoft.com/windowsserver2003.

For more information about IPv6, see the Microsoft Windows IPv6 Web site (http://www.microsoft.com/ipv6)