Microsoft® Windows NT™ 3.5/3.51: TCP/IP Implementation Details

Microsoft

A White Paper from Corporate Network Systems and the Business Systems Division

Microsoft®

Microsoft® Windows NT™ 3.5/3.51: TCP/IP Implementation Details

TCP/IP Protocol Stack and Services, Version 1.0

A White Paper from Corporate Network Systems
and the Business Systems Division

Dave MacDonald
September 1995

Legal Notice

0995
Part No. 098-62170

Table of Contents

- UDP and Name Resolution
- Mailslots over UDP
- NetBIOS over TCP/IP
- The Transport Driver Interface (TDI)
  - TDI features

Network Application Interfaces

- Overview
- Windows Sockets
  - Applications
  - Name Resolution
  - Support for IP Multicasting
  - The Backlog Parameter
- NetBIOS Over TCP/IP
  - NetBIOS Names
  - NetBIOS Name Registration and Resolution
  - NetBIOS Over TCP Sessions
  - NetBIOS Datagram Services

Microsoft TCP/IP Client and Server Applications

- Overview
- Dynamic Host Configuration Protocol (DHCP)
  - Obtaining Configuration Parameters Using DHCP
  - Lease Expiration and Renewal
- Windows Internet Name Service (WINS)
  - WINS Name Registration and Resolution
  - WINS in a DHCP Environment
- Domain Name System (DNS)
  - Integration of the DNS and WINS
- The Browser
  - Master Browser Elections
  - Maintaining Browse Lists
  - Requesting Browse Lists
  - The Domain Master Browser
  - Browser Enhancements
- Windows NT Workstation and Windows NT Server Services
  - Logging On
  - Connecting to Network Resources
  - Optimizations
- Microsoft Remote Access PPP/SLIP Support
  - RAS Servers
  - RAS Clients
  - Using RAS To Route Between Networks
  - Bandwidth Considerations
- Simple Network Management Protocol (SNMP) Agent

TCP/IP Troubleshooting Tools and Strategies

- Overview
- IPConfig
- Ping
- ARP
- Tracert
- Route
- Netstat
- NBTStat

Introduction

Microsoft® has adopted TCP/IP as the strategic enterprise network transport for its platforms. Several years ago, 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 Windows® NT™ 3.5 operating system, Microsoft introduced a completely re-written TCP/IP stack. This new stack was designed to incorporate many of the advances in performance and ease of administration made over the past decade. The stack is a high-performance, portable, 32-bit implementation of the industry standard TCP/IP protocol.

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

- Standards compliant
- Interoperable
- Portable
- Scaleable
- High performance
- Versatile
- Self-tuning
- Easy to administer
- Adaptable

The base code described here is shared by all Microsoft 32-bit TCP/IP protocol stacks (TCP/IP-32, Windows NT™, and Windows® 95); however, there are small differences in implementation, configuration methods, and available services.

This paper is intended to provide implementation details and is a supplement to the Microsoft Windows NT 3.5 and 3.51 TCP/IP manuals. The primary target audience consists of network engineers and support professionals who are already familiar with TCP/IP. The Microsoft TCP/IP protocol suite is examined in this paper from the bottom up.

Network traces are used throughout to help illustrate concepts. These traces were gathered and formatted using Microsoft Network Monitor, a software-based protocol tracing and analyses tool included in the Microsoft Systems Management Server product.

Capabilities/Functionality

Overview

The TCP/IP suite for Windows NT 3.5*x* was designed to make it much easier to integrate Microsoft systems into large-scale corporate and government networks. The product offers many new features and services to make administration easier and to improve interoperability. This TCP/IP suite makes Windows NT an "Internet Ready" platform.

Support for standard features

- Ability to bind to multiple network cards with different media types
- Logical multi-homing
- Internal IP routing capability
- IGMP (IP Multicasting) support
- Duplicate IP address detection
- Multiple default gateways
- Dead gateway detection
- Automatic PMTU (Path Maximum Transfer Unit) discovery

Performance enhancements

- Greatly reduced broadcast traffic
- Shorter code paths/reduced CPU utilization
- Self-tuning features

Services available

- DHCP (Dynamic Host Configuration Protocol) client and server
- WINS (Windows Internet Name Service), a NetBIOS name server
- Dial-up (PPP/SLIP) support
- TCP/IP network printing (lpr/lpd)
- SNMP agent
- NetBIOS interface
- Windows Sockets interface
- RPC (Remote Procedure Call)
- NetDDE (Network Dynamic Data Exchange)
- WAN (Wide Area Network) browsing support
- High performance FTP server

- Basic TCP/IP connectivity utilities, including: finger, ftp, rcp, rexec, rsh, telnet, and tftp
- Server software for simple network protocols, including: Character Generator, Daytime, Discard, Echo, and Quote of the Day
- TCP/IP management and diagnostic tools, including: arp, hostname, ipconfig, lpq, nbtstat, netstat, ping, route, and tracert

Internet RFCs (Requests for Comments) Supported by Microsoft Windows NT 3.5*x* TCP/IP

RFCs are a constantly evolving series of reports, proposals for protocols, and protocol standards used by the Internet community. RFCs can be obtained via FTP from NIS.NSF.NET, NISC.JVNC.NET, VENERA.ISI.EDU, WUARCHIVE.WUSTL.EDU, SRC.DOC.IC.AC.UK, FTP.CONCERT.NET, DS.INTERNIC.NET, or NIC.DDN.MIL. Details on obtaining RFCs via FTP or EMAIL may be obtained by sending an EMAIL message to "rfc-info@ISI.EDU" with the message body "help: ways_to_get_rfcs". For example:

> To: rfc-info@ISI.EDU
> Subject: getting rfcs
> help: ways_to_get_rfcs

The relevant RFCs supported by this version of Microsoft TCP/IP (and for Microsoft Remote Access Service) are listed below:

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
1009	Requirements for Internet Gateways
1034, 1035	Domain Name System (DNS)
1042	IP over Token Ring
1055	Transmission of IP over Serial Lines (IP-SLIP)
1112	Internet Gateway Multicast Protocol (IGMP)
1122, 1123	Host Requirements (communications and applications)
1134	Point to Point Protocol (PPP)
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

RFC	Title
1191	Path MTU Discovery
1201	IP over ARCNET
1231	IEEE 802.5 Token Ring MIB (MIB-II)
1332	PPP Internet Protocol Control Protocol (IPCP)
1334	PPP Authentication Protocols
1518	An Architecture for IP Address Allocation with CIDR
1519	Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy
1533	DHCP Options and BOOTP Vendor Extensions[1]
1534	Interoperation Between DHCP and BOOTP
1541	Dynamic Host Configuration Protocol (DHCP)
1542	Clarifications and Extensions for the Bootstrap Protocol
1547	Requirements for Point to Point Protocol (PPP)
1548	Point to Point Protocol (PPP)
1549	PPP in High-level Data Link Control (HDLC) Framing
1552	PPP Internetwork Packet Exchange Control Protocol (IPXCP)
Draft RFCs	PPP over ISDN; PPP over X.25; Compression Control Protocol

[1] The Microsoft DHCP server does not support BOOTP. BOOTP requests are silently ignored. However, a DHCP server and a BOOTP server can co-exist.

Architectural Model

Overview

The Microsoft TCP/IP protocol suite is comprised of *core protocol elements*, *services*, and the *interfaces* between them. The Transport Driver Interface (TDI) and the Network Driver Interface (NDIS) are public and their specifications are available from Microsoft[2]. In addition, there are a number of higher level interfaces available to user-mode applications. The two most commonly used are Windows Sockets and NetBIOS.

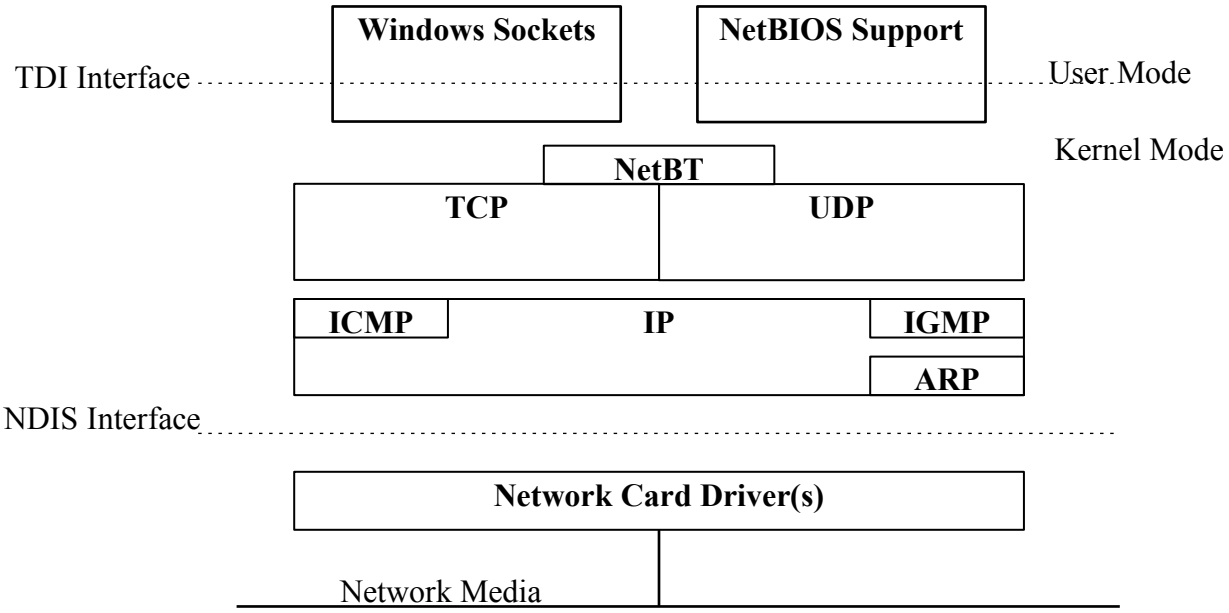

Figure 1: The Windows NT TCP/IP Network Model

[2] Specifications and programming information are included in the Windows NT Device Driver Kit (DDK). Some information is also available from the Microsoft Internet site (http://www.microsoft.com and ftp://ftp.microsoft.com).

The NDIS Interface and Below

Overview

Microsoft networking protocols communicate with network card drivers using the *Network Driver Interface Specification* (NDIS). 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 (3.0)

The NDIS interface supports basic services that allow a protocol module to send raw packets over a network device, and allow it to be notified of incoming packets received by a network device. NDIS-compliant drivers are available for a wide variety of network interface cards (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 from the Windows NT network control panel.

Since the NDIS interface handles raw packets, the protocol stack is normally responsible for building each frame, including MAC (Media Access Control) layer headers. This means that the protocol stack must explicitly support each media type. Windows NT 3.5*x* TCP/IP provides support for:

- Ethernet (and 802.3 SNAP)
- FDDI
- Token Ring (802.5)
- ARCNET
- WAN (switched virtual circuit wide area media, such as ISDN, X.25, and dial-up or dedicated asynchronous lines)

In addition, there are now some ATM adapters available for Windows NT. The drivers for these adapters use “LAN emulation” to appear to the protocol stack as a supported media type, such as Ethernet.

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 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 1’s broadcast address (FF-FF-FF-FF-FF-FF)

- Multicast addresses that a protocol driver on this host has registered interest in using an NDIS primitive

Because this first filtering decision is made by the hardware, all frames not meeting the filter criteria are discarded by the NIC without any CPU processing. All frames (including broadcasts) that do pass the hardware filter get passed up to the NIC driver through a hardware interrupt[3]. The NIC driver is software 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 specification provides more detail on this process.

Frames are indicated up to all bound transport drivers, in the order that they are bound. By default, the binding order is the alphabetical order of their key names in the registry.

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 change with each "hop" through a network-layer device (router).

MTU (Maximum Transfer Unit)

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, which optimizes packet sizes for each media automatically.

If a NIC driver such as an ATM driver uses LAN emulation mode, it may report that it has an MTU higher than what is expected for that media type. For instance, it may emulate Ethernet but report an MTU of 9180 bytes. Windows NT will accept and use the MTU size reported by the adapter even when it exceeds the normal MTU for a given media type.

[3] Most NICs have the ability to be placed into "promiscuous mode." When placed in this mode, the NIC does not perform any address filtering on frames that appear on the media. Instead, it indicates every frame upwards that passes the cyclic redundancy check (CRC). This feature is used by some protocol analysis software, such as Microsoft Network Monitor.

Core Protocol Stack Components and the TDI Interface

Overview

The core protocol stack components are those shown between the NDIS and TDI interfaces in Figure 1. They are implemented in the Windows NT TCPIP.SYS driver. The Microsoft stack is accessible using the TDI interface and the NDIS interface, but does not support "raw" sockets access to the IP layer.

Address Resolution Protocol (ARP)

IP address-to-MAC address resolution for outgoing packets is performed by ARP. As each outgoing IP datagram is encapsulated into 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 destination IP address on every outbound IP datagram to the *ARP cache* for the NIC that frame will be sent over. If there is a matching entry, then the MAC address is retrieved from the cache. If not, ARP broadcasts an *ARP Request Packet* onto 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, 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

Entries in the ARP cache may be viewed, added, or deleted using the *ARP* utility[4]. Examples are shown below. Entries added manually are static, and do not get aged out of the cache like dynamic entries do.

To view the ARP cache:

```
C:\>arp -a

Interface: 199.199.40.123
 Internet Address   Physical Address   Type
 199.199.40.1       00-00-0c-1a-eb-c5   dynamic
 199.199.40.124     00-dd-01-07-57-15   dynamic

Interface: 10.57.8.190
 Internet Address   Physical Address   Type
 10.57.9.138        00-20-af-1d-2b-91   dynamic
```

The computer in this example is multi-homed (has more than one NIC), so there is a separate ARP cache for each interface. *ARP -s* can be used to add a static entry to the arp cache used by the second interface, for the host whose IP address is 10.57.10.32 and whose NIC address is 00608C0E6C6A, shown on the next page:

[4] Windows NT 3.51 includes fixes to the ARP utility related to adding entries. See the Microsoft KnowledgeBase for details.

```
C:\>arp -s 10.57.10.32 00-60-8c-0e-6c-6a 10.57.8.190

C:\>arp -a

Interface: 199.199.40.123
 Internet Address   Physical Address   Type
 199.199.40.1       00-00-0c-1a-eb-c5   dynamic
 199.199.40.124     00-dd-01-07-57-15   dynamic

Interface: 10.57.8.190
 Internet Address   Physical Address   Type
 10.57.9.138        00-20-af-1d-2b-91   dynamic
 10.57.10.32        00-60-8c-0e-6c-6a   static
```

ARP Cache Aging

Windows NT 3.5*x* adjusts the size of the ARP cache automatically to meet the needs of the system. Entries are aged out of the ARP cache if they are not used by any outgoing datagrams for two minutes. Entries that are being referenced get aged out of the ARP cache after 10 minutes. Entries added manually are not aged out of the cache. Entries can be deleted from the cache using *arp -d* as shown below:

```
C:\>arp -d 10.57.10.32

C:\>arp -a

Interface: 199.199.40.123
 Internet Address   Physical Address   Type
 199.199.40.1       00-00-0c-1a-eb-c5   dynamic
 199.199.40.124     00-dd-01-07-57-15   dynamic

Interface: 10.57.8.190
 157.57.9.1  Internet Address   Physical Address   Type
10.57.9.138        00-20-af-1d-2b-91  dynamic
```

ARP will only queue one outbound IP datagram to a given 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.

Internet Protocol (IP)

IP is the "mailroom" of the TCP/IP stack, where packet sorting and delivery takes 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 the MAC addresses, the IP addresses in a datagram remain the same throughout a packet's journey across an internetwork. IP layer functions are described below.

Routing

Routing is the primary function of IP. Datagrams are handed to the IP protocol from UDP and TCP above, and from the NIC(s) below. Each datagram is labeled with a source and destination IP address. The IP protocol 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 via one of the locally attached NICs.
- It can be discarded.

The route table maintains four different types of routes. They are listed below in the order that they are searched for a match:

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

The route table may be viewed from the command prompt as shown below:

```
C:\>route print

Network Address  Netmask          Gateway Address  Interface       Metric
0.0.0.0          0.0.0.0          199.199.40.1     199.199.40.123  1
127.0.0.0        255.0.0.0        127.0.0.1        127.0.0.1       1
199.199.40.0     255.255.255.0    199.199.40.123   199.199.40.123  1
199.199.40.123   255.255.255.255  127.0.0.1        127.0.0.        1
199.199.40.255   255.255.255.255  199.199.40.123   199.199.40.123  1
224.0.0.         224.0.0.0        199.199.40.123   199.199.40.123  1
255.255.255.255  255.255.255.255  199.199.40.123   199.199.40.123  1
```

The route table above is for a computer with the class C IP address 199.199.40.123. It contains 7 entries, described below:

- The first entry, to address 0.0.0.0, is the default route.
- The second entry is for the loopback address, 127.0.0.0.
- The third entry is a network route, for the network 199.199.40. The local interface is specified as the path to this network.
- The fourth entry is a host route for the local host. Note that it specifies the loopback address, which makes sense because a datagram bound for the local host should be handled internally.
- The fifth entry is for the subnet broadcast address (again specifying the local interface).
- The sixth entry is for IP multicasting, which is discussed later in this document.
- The final entry is for the limited broadcast address.

On this host, if a packet is sent to 199.199.40.122, the table is first scanned for a host route (not found), then for a subnet route (not found), then for a network route (that is found). The packet is sent via the local interface 199.199.40.123. If a packet is sent to 199.200.1.1, the same search is used, and no host, subnet, or network route is found. In this case, the packet is directed to the default gateway, by inserting the MAC address of the default gateway into the destination MAC address field.

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 system may advise it (using ICMP, as explained later) of a better route to a specific network, subnet, or host. Routes also may be added manually by using the **route** command. Under Windows NT 3.5, routes added manually were

temporary and would be gone from the table after a reboot. However, under Windows NT 3.51, the **-p (persistent)** switch can be used with the route command to specify permanent routes. Permanent routes are stored in the registry under:

> HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\PersistentRoutes

Most routers use a protocol such as *RIP* (Routing Information Protocol) or OSPF (Open Shortest Path First) to exchange routing tables with each other. However, Windows NT 3.5*x* does not include RIP. This means that if Windows NT computers are used as routers, they do not exchange routing tables, so manual configuration of static routes may be necessary. Information on how to set up static routes is available in the Windows NT TCP/IP manuals, and from the Microsoft KnowledgeBase. Another alternative is to obtain the publicly available Multiple Provider Router beta from Microsoft[5].

By default, Windows NT systems do not behave as routers. Internal routing may be enabled from the TCP/IP Advanced Configuration screen in the network control panel.

When running multiple logical subnets on the same physical network, the following command can be used to tell IP to treat all subnets as local and to use ARP directly for the destination:

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

Thus, packets destined for "non-local" subnets will be 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 might be useful where several class "C" networks are being used on one physical network with no router to the outside world.

Duplicate IP Address Detection

Duplicate address detection is an important feature. When the stack is first initialized, a "gratuitous" ARP Request is broadcast for the IP address of the local host. If another system replies, the IP address is already in use. When this happens, the Windows NT computer will still boot; however, IP on the offending interface is disabled, a system log entry is generated, and an error popup is displayed. If the system that is "defending" the address is also a Windows NT computer, a system log entry is generated and an error popup is displayed there; however, its interface will continue to operate. After transmitting the ARP reply, the "defending" system ARPs for its own address again so that other hosts on the network will maintain the correct mapping for the address in their ARP caches.

A computer using a duplicate IP address may be started while it is not attached to the network, in which case no conflict would be detected at that point. However, if it is then plugged into the network, the first time that it ARPs for another IP address, any Windows NT computer with a conflicting address will detect the conflict. The computer detecting the conflict will display an error popup and log a detailed event in the system log. A sample event log entry is shown below:

> ** The system detected an address conflict for IP address 199.199.40.123 with the system having network hardware address 00:DD:01:0F:7A:B5. Network operations on this system may be disrupted as a result. **

[5] RIP is part of the public beta of MPR (Multiple Provider Router) available from ftp.microsoft.com.

Multi-homing

When a computer is configured with more than one IP address it is referred to as a multi-homed system. Multi-homing is supported in three different manners:

- Multiple IP addresses per NIC.
  - Five addresses per card may be configured using Control Panel; however, more may be added in the registry. For details see the *IPAddress* registry parameter in Appendix A.
  - NetBT (NetBIOS over TCP/IP per RFC1001/1002) only supports one IP address per interface card. When a NetBIOS name registration is sent out, only one IP address will be registered per interface. This registration will occur over the IP address that is listed first in the network control panel.
- Multiple NICs per physical network.
  - No restrictions other than hardware.
- Multiple networks and media types.
  - No restrictions other than hardware and media support. See the Network Interface Card/Driver section of this document for supported media types.

When an IP datagram is sent from a multi-homed host, it will be handed down to the interface card with the best apparent route to the destination. Accordingly, the datagram may bear the source IP address of one interface in the multi-homed host, yet be placed on the media by a different NIC. The source MAC address on the frame will be that of the NIC that actually transmitted the frame onto the media, and the source IP address will be the one that the sending application sourced it from, not necessarily one of those associated with the sending NIC in the configuration screens in the network control panel.

When a computer is multi-homed with NICs attached to disjoint networks (networks that are separate from and unaware of each other, such as one connected via RAS), routing problems may arise. It is often necessary to set up static routes to remote networks in this scenario.

Classless Interdomain Routing (CIDR)

Also known as *supernetting,* CIDR may be used to consolidate several class C network addresses into one logical network. CIDR is described in RFC1518/1519. To use supernetting, the IP network addresses that are to be combined must share the same high-order bits, and the subnet mask is “shortened” to take bits away from the network portion of the address and add them to the host portion.

This is best explained with an example. The class C network addresses 199.199.5.0, 199.199.6.0, and 199.199.7.0 can be combined by using a subnet mask of 255.255.252.0 for each:

NET 199.199.5 (1100 0111 . 1100 0111 . 0000 0101.0000 0000)
NET 199.199.6 (1100 0111 . 1100 0111 . 0000 0110.0000 0000)
NET 199.199.7 (1100 0111 . 1100 0111 . 0000 0111.0000 0000)
MASK 255.255.252.0 (1111 1111 . 1111 1111 . 1111 1100.0000 0000)

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

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 in order to participate in a wide-area conference, for instance.

Windows NT 3.5*x* is level-2 (send and receive) compliant with RFC1112. IGMP is the protocol used to manage IP multicasting, as described later in this document.

Internet Control Message Protocol (ICMP)

ICMP is a maintenance protocol specified in RFC792 and is normally considered to be part of the IP layer. ICMP messages are encapsulated within IP datagrams, so they may be routed throughout an internetwork. ICMP is used by Windows NT to:

- Build and maintain route tables.
- Assist in Path Maximum Transfer Unit (PMTU) discovery.
- Diagnose problems (ping, tracert).
- Adjust flow control to prevent link or router saturation.

Maintaining Route Tables

When a Windows NT computer is initialized, the route table normally contains only a few entries. One of those specifies a *default gateway*. Datagrams that have a destination IP address with no match in the route table are sent to the default gateway. However, since routers share information about network topology with each other, the default gateway may know of a better route to a given address. When this is the case, upon receiving a datagram that could be taking the better path, the router forwards the datagram normally, then advises the sender of the better route using an *ICMP redirect* message. These messages can specify redirection for one host, a subnet, or for an entire network. When a Windows NT computer receives an ICMP redirect, a 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, the route table is adjusted accordingly. 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, then the ICMP redirect is ignored.

Path Maximum Transfer Unit (PMTU) Discovery

PMTU discovery is used by TCP, as described later in this document. The mechanism relies on ICMP *destination unreachable* messages.

Use Of ICMP For Diagnosing Problems

Ping 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. Ping is explored in more detail in the troubleshooting section of this document.

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 TTL (Time To Live) field in the IP header by one starting at one, and analyzing the ICMP errors that get returned. Each succeeding echo request should get one hop further into the network before the TTL field reaches 0 and an *ICMP Time Exceeded* error is returned by the router attempting to forward it. Tracert simply prints out an ordered list of the routers in the path that returned these error messages. If the **-d (don't do a DNS lookup on each IP address)** switch is used, then the IP address of the near-side interface of the routers is reported. The example below illustrates using tracert to find the route from a computer dialed in over PPP to an Internet 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]
```

Flow Control Via ICMP

If a host is sending datagrams to another at a rate that is saturating the routers or links between them, it may receive an *ICMP Source Quench* message asking it to slow down. The TCP/IP stack in Windows NT honors a source quench message as long as it contains the header fragment of one of its own datagrams from an active TCP connection. If a Windows NT computer is being used as a router, and it is unable to forward datagrams at the rate they are arriving, it drops any datagrams that cannot be buffered but does not send ICMP source quench messages to the senders.

Internet Group Management Protocol (IGMP)

Windows NT 3.5 and 3.51 provide level 2 (full) support for IP multicasting as specified in RFC1112. The introduction to RFC1112 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-efforts" reliability as regular unicast IP datagrams, i.e., the datagram is not guaranteed to arrive intact at 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 by the system. 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 via the local interface card, not forwarded to the default gateway. The following route (seen with the “route print” command) illustrates this:

```
Network Address  Netmask    Gateway Address  Interface     Metric
224.0.0.0        224.0.0.0  10.57.9.138      10.57.9.138   1
```

Host group addresses are easily identified, as they are from the class D range, 224.0.0.0 to 239.255.255.255. These IP addresses all have “1110” as their high-order 4 bits.

To send a packet to a host group using the local interface, the IP address must be resolved to a MAC address. From RFC1112:

> 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 instance, 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).

Since more than one host group address might map to the same Ethernet multicast address, the NIC may indicate up some multicasts for a host group for which no local applications have registered interest. These extra multicasts are discarded.

Finally, the protocol stack must provide a means of joining and leaving host groups.

Multicast Extensions to Windows Sockets[6]

Internet Protocol multicasting is currently supported only on AF_INET sockets of type SOCK_DGRAM. By default, IP multicast datagrams are sent with a time-to-live (TTL) of 1. The setsockopt() call can be used by an application 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.

Transmission Control Protocol (TCP)

TCP provides a connection-based, reliable byte-stream service to applications. Microsoft networking relies upon the TCP transport 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 headers and data of each segment to reduce the chance of network corruption going undetected.

TCP Receive Window Size Calculation

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 NT 3.5*x* TCP/IP stack was designed to self-tune itself in most environments. Instead of using a hard-coded default receive window size, TCP adjusts to even increments of the MSS (maximum segment size) negotiated during connection setup. Matching the receive window to even increments of the MSS increases the percentage of full-sized TCP segments utilized during bulk data transmission. The receive window size defaults in the following manner:

- TCPWindowSize = 8Kbytes rounded up to the nearest MSS increment for the connection.
- If that isn't at least 4 times the MSS, then it's adjusted to 4 * MSS, with a maximum size of 64K[7]

[6] Details on multicast extensions for Windows Sockets are available from ftp.microsoft.com.
[7] 64K is the maximum window size due to the 16-bit size of the field in the TCP header. RFC1323 describes a TCP window scale option that can be used to obtain larger receive windows; however Windows NT TCP/IP does not yet implement that option.

For Ethernet, the window will normally be set to 8760 bytes (8192 rounded up to six 1460-byte segments), and for 16/4 Token Ring or FDDI it will be around 16Kbytes. These values are default and it's not generally advisable to alter them; however, there are two methods for setting the receive window size to specific values:

1. The *TcpWindowSize* registry parameter (a global setting for the system).
2. The setsockopt() Windows Sockets call (on a per-socket basis).

Delayed Acknowledgments

Per RFC1122, TCP uses delayed acknowledgments (acks) to reduce the number of packets sent on the media. The Microsoft stack takes a common approach to implementing delayed acks. As data is received by TCP on a given connection, it only sends an acknowledgment back if one of the following conditions is met:

- If no ack was sent yet for the previous segment received.
- If a segment was received, and no other segment arrives within 200ms.

In summary, normally an ack is sent for every other TCP segment received on a connection, unless the delayed ack timer (200ms) expires. There is no configuration parameter to disable delayed acks.

PMTU (Path Maximum Transfer Unit) Discovery

PMTU discovery is described in RFC1191. 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. The MSS for a system is usually the MTU at the link layer minus 40 bytes for the IP and TCP headers.

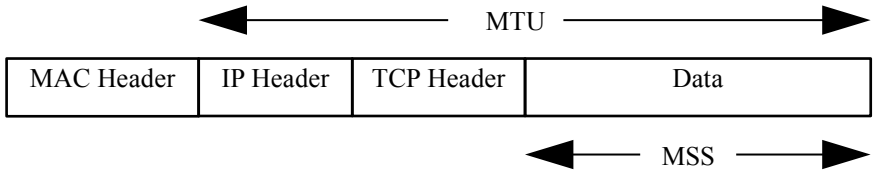

Figure 2: MTU versus MSS

When TCP segments are destined to a non-local network, the "don't fragment" bit is set in the IP header. Any router or media along the path may have an MTU that differs from that of the two hosts. If a media is encountered with an MTU that is too small for the IP datagram being routed, the router will attempt to fragment the datagram accordingly. Upon attempting to do so, it will find that the "don't fragment" bit in the IP header is set. At this point, the router should inform the sending host with an ICMP destination unreachable message that the datagram can't be forwarded further without fragmentation. Most routers will also specify the MTU that is allowed for the next hop by putting the value for it in the low-order 16 bits of the ICMP header field that is labeled "unused" in the ICMP specification. See RFC1191, 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 will be no larger than the maximum size that can

traverse the path without fragmentation. The minimum MTU permitted by RFCs is 68 bytes, and this limit is enforced by Windows NT TCP .

Some non-compliant 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 the TCP/IP stack in Windows NT 3.5*x* to work around these problematic routers. These registry entries are described in more detail in Appendix A:

- *EnablePMTUBHDetect* – Adjusts the PMTU discovery algorithm to attempt to detect these "black hole" routers. Black Hole detection is disabled by default.
- *EnablePMTUDiscovery* – Completely enables or disables the PMTU discovery mechanism. When PMTU discovery is disabled, an MTU of 576 bytes is used for *all non-local* destination addresses. PMTU discovery is enabled by default.

The PMTU between two systems can be discovered manually using ping 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. Note that 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 would normally be 20 bytes. In the case below, (Ethernet) the link layer MTU is the maximum-sized ping buffer plus 28, or 1500 bytes:

```
C:\temp>ping -f -n 1 -l 1472 10.57.8.1
Pinging 10.57.8.1 with 1472 bytes of data:
Reply from 10.57.8.1: bytes=1472 time<10ms TTL=30

C:\temp>ping -f -n 1 -l 1473 10.57.8.1
Pinging 10.57.8.1 with 1473 bytes of data:
Packet needs to be fragmented but DF set.
```

In the example shown above, the router returned an ICMP error message, that ping interpreted for us. If the router had been a "black hole" router, the 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 error message is shown below:

```
+ FRAME: Base frame properties
+ FDDI: Length = 77
+ LLC: UI DSAP=0xAA SSAP=0xAA C
+ SNAP: ETYPE = 0x0800
+ IP: ID = 0x0; Proto = ICMP; Len: 56
 ICMP: Destination Unreachable, Destination: 199.199.40.125
   ICMP: Packet Type = Destination Unreachable
   ICMP: Unreachable Code = Fragmentation Needed, DF Flag Set
   ICMP: CheckSum = 0x8ABF
   ICMP: Data: Number of data bytes remaining = 28 (0x001C)

00000: 50 00 60 8C 14 C7 0E 00 00 0C 1A EB C0 AA AA 03
00010: 00 00 00 08 00 45 00 00 38 00 00 00 00 FF 01 D3
00020: 36 C7 C7 2C 01 C7 C7 2C FE 03 04 8A BF 00 00 05
00030: C7 45 00 05 F8 55 24 40 00 1F 01 1B D7 C7 C7 2C
00040: FE C7 C7 28 7D 08 00 00 75 01 00 63 00
```

Network Monitor did not parse the MTU suggestion in this frame, but it is shown underlined in the hex portion of the trace. This error was generated by using *ping -f -l*

2000 on an FDDI-based host to send a large datagram through a router to an Ethernet host. When the router tried to place the large frame onto the Ethernet segment, it found that fragmentation was not allowed, so it returned the error message indicating the largest datagram that could be forwarded is 0x5c7, or 1479 bytes.

Dead Gateway Detection

Dead gateway detection is used to allow TCP to detect failure of the default gateway, and to make an adjustment to the IP routing table to use another default gateway. The Microsoft TCP/IP stack uses the TRIGGERED RESELECTION method described in RFC816. When TCP has tried one-half of the *TcpMaxDataRetransmissions* times to send a packet through the default gateway, it will advise IP to switch to the next default gateway in the list and try that one[8]. Additional default gateways can be configured in the TCP/IP Advanced Configuration screen in the network control panel.

Re-transmission Behavior

TCP starts a re-transmission 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, then the segment is retransmitted, up to the *TcpMaxDataRetransmissions* times. The default value for this parameter is 5.

The re-transmission timer is initialized to 3 seconds when a TCP connection is established; however it is adjusted "on the fly" to match the characteristics of the connection using Smoothed Round Trip Time (SRTT) calculations as described in RFC793. The timer for a given segment is doubled after each re-transmission of that segment. Using this algorithm, TCP tunes itself to the "normal" delay of a connection. TCP connections over high-delay links will take much longer to time out than those over low-delay links[9].

The following trace clip shows the re-transmission algorithm for two hosts connected over Ethernet on the same subnet. An FTP file transfer was in progress, when the receiving host was disconnected from the network. Since the SRTT for this connection was very small, the first re-transmission was sent after about one-half second. The timer was then doubled for each of the re-transmissions that followed. After the fifth re-transmission, the timer is once again doubled, and if no acknowledgment is received before it expires, then the transfer is 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
```

[8] IP utilities such as ping will not trigger the dead gateway detection process. However, they will use the current default gateway, so if TCP detects a dead gateway and selects a new one, IP utilities will then function using the new gateway.
[9] Adding [1] to the registry parameter *TcpMaxDataRetransmissions* approximately doubles the total re-transmission time-out period for all connections.

TCP Keepalive Messages

A TCP keepalive packet is simply an "ack" with the sequence number set to one less than the current sequence number for the connection. A system receiving one of these acks should respond with an ack for the current sequence number. Keepalives can be used to verify that the computer at the remote end of a connection is still available. TCP keepalives can be sent once every KeepAliveTime (defaults to 7,200,000 milliseconds or two hours), if no other data or higher level keepalives have been carried over the TCP connection. If there is no response to a keepalive, it is repeated once every KeepAliveInterval seconds. KeepAliveInterval defaults to 1 second. NetBT connections, such as those used by many Microsoft networking components, send NetBIOS keepalives more frequently, so normally no TCP keepalives will be sent on a NetBIOS connection. TCP keepalives are disabled by default, but Windows Sockets applications may enable them using setsockopt().

Slow Start Algorithm and Congestion Avoidance

When a connection is established, TCP treads lightly at first in order to assess the bandwidth of the connection and to avoid overflowing the receiving host or any other devices/links in the path. The send window is set to one TCP segment, and if that is acknowledged, then it is doubled to two segments[10]. If those are acknowledged, then it is doubled 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, at any time during transmission, congestion could still occur on a connection. If this happens (evidenced by the need to re-transmit) , a congestion avoidance algorithm is used to reduce the send window size temporarily, and to grow it back towards the receive window size more slowly. Slow start and congestion avoidance are discussed further in RFC1122.

Silly Window Syndrome (SWS)

Silly Window Syndrome is described in RFC1122 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 NT TCP/IP Windows NT implements SWS avoidance per RFC1122 by not sending more data until there is a sufficient window size advertised by the receiving end to send a full segment. It also implements SWS on the receive end of a connection by not opening the receive window in increments of less than a TCP segment.

Nagle Algorithm

Windows NT TCP/IP implements the Nagle algorithm described in RFC896. The purpose of this algorithm is to reduce the number of "tiny" segments sent, especially on high-delay (remote) links. The Nagle algorithm allows only one small segment to be

[10] As of Windows NT 3.51 Service Pack 1, a small change was made to the slow start implementation. Instead of sending one TCP segment when starting out, Windows NT TCP now sends two. This avoids the need to wait for the delayed ack timer to expire on the destination machine, which improves performance for some applications.

outstanding at a time without acknowledgment. If more small segments are generated while awaiting the ack for the first one, then these segments are coalesced into one larger segment. Any full-sized segment is always transmitted immediately, assuming 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 trace captured by Microsoft Network Monitor. The trace was captured by using PPP to dial up an Internet provider at 9600 BPS. A Telnet (character mode) session was established, then the "y" key was held down on the Windows NT Workstation. 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, 3 to 4 "y" characters were saved up each time and sent together in one segment. The Nagle algorithm resulted in a huge savings in the number of packets sent–it was reduced by a factor of about three.

```
Time  Source IP     Dest IP       Prot  Description
0.644 204.182.66.83 199.181.164.4 TELNET To Server From Port = 1901
0.144 199.181.164.4 204.182.66.83 TELNET To Client With Port = 1901
0.000 204.182.66.83 199.181.164.4 TELNET To Server From Port = 1901
0.145 199.181.164.4 204.182.66.83 TELNET To Client With Port = 1901
0.000 204.182.66.83 199.181.164.4 TELNET To Server From Port = 1901
0.144 199.181.164.4 204.182.66.83 TELNET To Client With 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 hex at the bottom.

```
*****************************************************************
Time  Source IP      Dest IP        Prot    Description
0.644 204.182.66.83  199.181.164.4  TELNET  To Server From 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 connection(s) by setting the TCP_NODELAY socket option. However, this practice should be avoided unless absolutely necessary as 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.

Throughput Considerations

TCP was designed to provide optimum performance over varying link conditions. Actual throughput for a link is dependent on a number of variables, but the most important factors are:

- Link speed (bits/second that can be transmitted)
- Propagation delay
- Window size (amount of unacknowledged data that may be outstanding on a TCP connection)
- Link reliability
- Router Congestion

TCP throughput calculation is discussed in detail in Chapters 20-24 of *TCP/IP Illustrated,* by W. Richard Stevens. Some key considerations are listed below:

- The capacity of a pipe is (bandwidth * 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. 65535 is the largest window size that can be specified due to its 16-bit field in the TCP header. RFC1323 describes a Window Scale option; however it has not been implemented yet by Windows NT TCP.
- Throughput will never exceed (window size / 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.
- Propagation delay is dependent upon the speed of light and latencies in transmission equipment and so on.
- Transmission delay depends on the speed of the media.
- For a given 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 NT TCP/IP will adapt to most network conditions and dynamically provide the best throughput and reliability possible on a per-connection basis. Attempts at manual tuning are *often counter-productive* unless a careful study of data flow is performed by a qualified network engineer.

User Datagram Protocol (UDP)

UDP provides a connectionless, unreliable transport service. It is often used for one-to-many communications, using broadcast or multicast IP datagrams. As delivery of UDP datagrams is not guaranteed, applications using UDP must supply their own mechanisms for reliability if needed. Microsoft networking uses UDP for logon, browsing, and name resolution.

UDP and Name Resolution

UDP is used for NetBIOS name resolution via unicast to a NetBIOS name server or subnet broadcasts, and for DNS (Domain Name System) hostname/IP address resolution. NetBIOS name resolution is accomplished over UDP port 137. DNS queries use UDP port 53. Since UDP itself does not guarantee delivery of datagrams, both of these services use their own re-transmission 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 nameserver of some type, or the use of static database files.

Mailslots over UDP

Mailslot messaging is used by many NetBIOS applications. A 2nd class mailslot is a simple mechanism for sending a message from one NetBIOS name to another over UDP. Mailslot messages may be broadcast on a subnet, or may be directed to the remote system. In order to direct a mailslot message to another system, there must be some method of NetBIOS name resolution available. Microsoft provides WINS (Windows Internet Name Server) for this purpose.

NetBIOS over TCP/IP

The Windows NT 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 RFC1001 and RFC1002. The NETBT.SYS driver is a kernel-mode component that supports the TDI interface. Services such as Windows NT Workstation and Windows NT Server services use the TDI interface directly, while traditional NetBIOS applications have their calls mapped to TDI calls via 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 document.

The Transport Driver Interface (TDI)

The Transport Driver Interface was developed by Microsoft to provide greater flexibility and functionality than is provided by existing interfaces such as NetBIOS and Windows Sockets. The TDI interface is exposed by all Windows NT transport providers. The TDI interface specification describes the set of primitive functions by which transport drivers and TDI clients communicate, and the call mechanisms used for accessing them. Currently, the TDI Interface is *kernel-mode* only.

The Windows NT 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 254 session limit.

TDI features

TDI may be the most difficult to use of all the Windows NT 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 NT transports support TDI. (*DLC does not.*)
- An open naming/addressing scheme.
- Message and stream mode data transfer.
- Asynchronous operation.
- Support for unsolicited indication of events.
- It is extensible–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 ACLs (Access Control Lists, used for security) on TDI address objects.

More information on the TDI interface is available from the Windows NT Device Driver Kit (DDK).

Network Application Interfaces

Overview

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 currently popular. A quick overview of the Windows Sockets Interface and the NetBIOS Interface is presented here.

Windows Sockets

Windows Sockets specifies a programming interface 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 2.0 is in a provisional state at the time of this writing[11]. Currently, support for raw sockets is not available, however it is planned for a future release.

Applications

There are many Windows Sockets applications available. A number of the utilities that ship with Windows NT are Windows Sockets based, including the FTP and DHCP clients and servers, telnet client, etc.

Name Resolution

Windows Sockets applications generally use the gethostbyname() call to resolve a hostname to an IP address. The gethostbyname() call uses the following (default) name lookup sequence:

- Check the hosts file for a matching name entry.
- If a Domain Name Server is configured, query it .
- If no match is found, try the NetBIOS name resolution sequence described in Figure 3, up until the point at which DNS resolution is attempted.

Support for IP Multicasting

The Windows Sockets API has been extended to provide support for IP multicasting. The extensions and the sample application *party.exe* illustrating usage are available from ftp.microsoft.com. Multicasting is also described in the Windows Sockets 2.0 specification and in the IGMP section of this document.

Internet Protocol multicasting is currently supported only on AF_INET sockets of type SOCK_DGRAM.

[11] Both specifications are available from the Microsoft Internet site on www.microsoft.com and ftp.microsoft.com.

The Backlog Parameter

Windows Sockets server applications generally create a socket and then use *listen()* to listen on it 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. The Windows Sockets 1.1 specification indicates that the maximum allowable value for *backlog* is 5; however, Windows NT will accept a backlog of up to 100. The FTP server in version 3.51 was modified to allow configuration of the backlog parameter by the administrator. FTP servers that are heavily used may benefit from increasing the backlog to a larger number than the default of 5. See Appendix D for details on this configuration setting.

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 NetBEUI local-area networking protocol with a NetBIOS application programming interface. NetBEUI is a small, fast protocol with no networking layer; thus, it is not routable and is often not suitable for WAN implementations. NetBEUI relies on broadcasts 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.

NetBIOS Names

The NetBIOS namespace is flat, meaning that all names within a network must be unique. NetBIOS names are 16 characters in length. Resources are identified by NetBIOS names, that are registered dynamically when computers boot, services 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 Windows NT Workstation and Windows NT Server services, allow the first 15 characters of a NetBIOS name to be specified by the user or administrator, but reserve the 16th character of the NetBIOS name to indicate a resource type (00-FF hex). Following are some example NetBIOS names used by Microsoft components:

Unique Names	
<computername>[00h]	W
<computername>[03h]	
<computername>[06h]	R
<computername>[1Fh]	
<computername>[20h]	
<computername>[21h]	F
<computername>[BEh]	Network
<computername>[BFh]	Network
<username>[03]	
<domain_name>[1Dh]	Master Browser
<domain_name>[1Bh]	Domain Master Browser

Group Names

<domain_name>[00h]	Domain Name
<domain_name>[1Ch]	Domain Controllers
<domain_name>[1Eh]	Browser Service Elections

To see which names a computer has registered over NetBT, type *nbtstat -n*.

NetBIOS Name Registration and Resolution

Windows NT TCP/IP 3.5*x* systems may use several methods for locating NetBIOS resources:

- NetBIOS name cache
- NetBIOS name server
- IP subnet broadcasts
- Static *LMHOSTS* files
- Static *HOSTS* files
- DNS servers

Earlier implementations used only cache, broadcasts, and LMHOSTS files; however, in version 3.5, a NetBIOS name server (WINS) was added, and modifications were made to allow NetBIOS applications to query the DNS namespace by appending configurable domain suffixes to a NetBIOS name.

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 for name registration and resolution.
- M-node – uses broadcasts for name registration. For name resolution, tries broadcasts first, but switches to p-node if no answer is received.
- H-node – uses NetBIOS name server for both registration and resolution; however, if no name server can be located, it switches to b-node. Continues to poll for nameserver and switches back to p-node when one becomes available.
- Microsoft-enhanced – Local LMHOSTS files or WINS proxies plus Windows Sockets gethostbyname() calls (using standard DNS and/or local HOSTS files) in addition to standard node types.

The plethora of configurable options sometimes makes it difficult to determine what name resolution methods to choose, and what name resolution order each configuration will use. Prakash Narasimhamurthy of Microsoft Consulting Services provided the flow charts shown in figures 3 and 4 to illustrate name resolution for the various node types.

- Microsoft ships a NetBIOS name server known as WINS (Windows Internet Name Service[12]). Most WINS clients are set up as h-nodes, i.e., they first attempt to register and resolve names using WINS, and if that fails they try local subnet broadcasts. Using a name server to locate resources is generally preferable to broadcasting, for two reasons:
  - Broadcasts are not usually forwarded over routers.
  - Broadcasts are received by all computers on a subnet, requiring processing time at each one.

[12] WINS is discussed in detail in a separate Windows NT Whitepaper (DHCPWINS.DOC, Part No. 098-56544) available on the Internet from ftp.microsoft.com and from the Microsoft Sales Information Center.

The Windows NT workstation service, server service, browser, messenger, and netlogon services are all (direct) NetBT clients. They use the Transport Driver Interface (described later in this whitepaper) to communicate with NetBT. Windows NT also includes a NetBIOS emulator. The emulator takes standard NetBIOS requests from NetBIOS applications and translates them to equivalent TDI primitives.

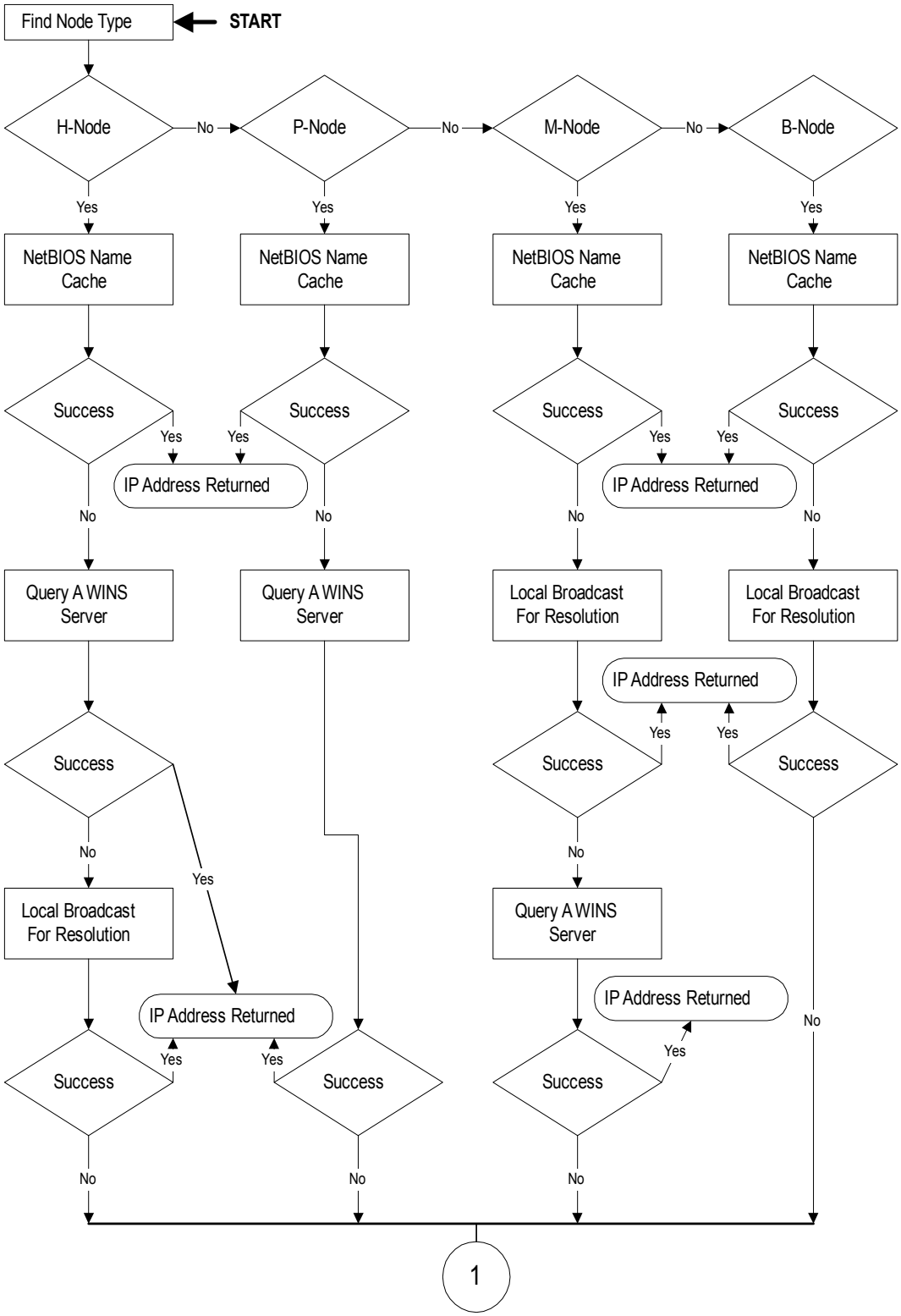

Figure 3: NetBIOS Name Resolution Flowchart (part 1 of 2)

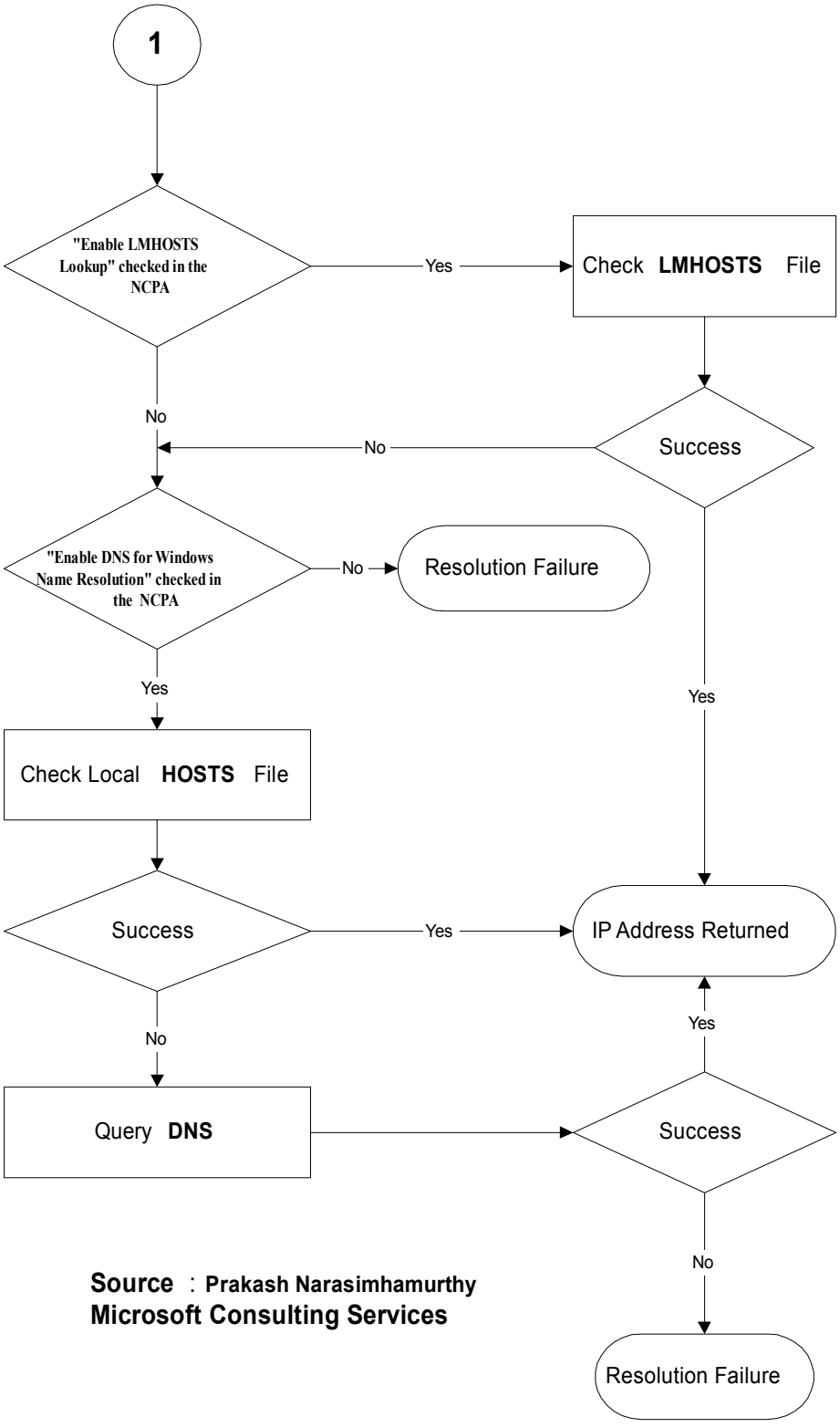

Figure 4: NetBIOS Name Resolution (part 2 of 2)

NetBIOS Over TCP Sessions

NetBIOS sessions are established between two names. For instance, when a Windows NT Workstation makes a file sharing connection to a server, the following sequence of events takes place:

- The NetBIOS name for the server is resolved to an IP address.
- A TCP connection is established from the workstation to the server, using port 139.
- The workstation sends a *NetBIOS Session Request* to the server name over the TCP connection. Assuming the server is listening on that name, it will respond affirmatively and a session is established.

Once the NetBIOS session has been established, the workstation and server negotiate a higher level protocol to use over it. Microsoft networking uses only one NetBIOS session between two names at any point in time. Any additional file or print sharing connections made after the first one are multiplexed over that same NetBIOS session.

NetBIOS keepalives are used on each connection to verify that the server and workstation are still both up and able to maintain their session. This way, if a workstation is shut down ungracefully, the server will eventually clean up the connection and associated resources, and vice versa. NetBIOS keepalives 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 will still be established to the server. However, the NetBIOS session request (using the wrong name) will be rejected by the server, as there is no listen posted on that name. Error 51 “remote computer not listening” will be 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 instance, the command *net send /d:mydomain test* would send a datagram containing the text “test” to the group name <mydomain>[03]. The <mydomain>[03] name would resolve to an IP subnet broadcast, so the datagram would be sent with the following characteristics:

- Destination MAC address: broadcast (FFFFFFFFFFFF).
- Source MAC address: The NIC address of the local computer.
- 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: <localmachine>[03] (the messenger service on the local computer).

All hosts on the subnet would pick up the datagram and process it at least to the UDP protocol. On hosts running a NetBIOS datagram service, UDP would hand the datagram to NetBT on port 138. NetBT would check the destination name to see if any application had posted a datagram receive on it, and if so would pass the datagram up. If no receive was posted, the datagram would be discarded.

Microsoft TCP/IP Client and Server Applications

Overview

This whitepaper is intended to provide an overview of the Windows NT 3.5*x* implementation of the TCP/IP stack, not the many clients and services that are shipped with the product or available from third parties. However, there are a few client and server components that are critical to the configuration and operation of the protocol suite, so an overview of them is presented here.

Dynamic Host Configuration Protocol (DHCP)

The DHCP client and server are Windows Sockets applications that are used to provide automatic configuration of various TCP/IP protocol components[13]. The server is configured with "scopes" that are ranges of IP addresses to hand out, along with additional configuration parameters that go along with those addresses. For instance, a scope might be set up for a range of IP addresses, and it might also include a default gateway, DNS server, NetBIOS Name Server (WINS), etc.

Obtaining Configuration Parameters Using DHCP

When a DHCP-enabled client boots for the very first time, it broadcasts a *DHCP Discover* request onto the local subnet. Any DHCP server that receives the request may respond with a *DHCP Offer* that contains proposed configuration parameters. The client can evaluate the offer, and respond with a *DHCP Request* to accept it. The server finalizes the transaction with a *DHCP Acknowledgment*. A sample of this sequence is explained below.

First, the DHCP Discover is sent as the stack initializes:

```
*****************************************************************
Time   Source IP   Dest IP          Prot Description
0.000  0.0.0.0     255.255.255.255  DHCP Discover  (xid=68256CA8)

+ FRAME: Base frame properties
 ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
  + ETHERNET: Destination address : FFFFFFFFFFFF
  + ETHERNET: Source address : 00DD01075715
    ETHERNET: Frame Length : 342 (0x0156)
    ETHERNET: Ethernet Type : 0x0800 (IP: DOD Internet Protocol)
    ETHERNET: Ethernet Data: Number of data bytes remaining = 328
              (0x0148)
 IP: ID = 0x0; Proto = UDP; Len: 328
    IP: Version = 4 (0x4)
    IP: Header Length = 20 (0x14)
  + IP: Service Type = 0 (0x0)
    IP: Total Length = 328 (0x148)
    IP: Identification = 0 (0x0)
  + IP: Flags Summary = 0 (0x0)
    IP: Fragment Offset = 0 (0x0) bytes
    IP: Time to Live = 32 (0x20)
    IP: Protocol = UDP - User Datagram
    IP: CheckSum = 0x99A6
```

[13] DHCP is described in more detail in RFC1541, and RFC1542, and in a separate whitepaper from Microsoft (Part No. 098-56544). The Microsoft client and server implementations were tested with preliminary clients and servers from many other vendors at DHCP "Bakeoff" events and should interoperate successfully with any other compliant implementations.

```
   IP: Source Address = 0.0.0.0
   IP: Destination Address = 255.255.255.255
   IP: Data: Number of data bytes remaining = 308 (0x0134)
 UDP: IP Multicast: Src Port: BOOTP Client, (68); Dst Port: BOOTP
      Server (67); Length = 308 (0x134)
   UDP: Source Port = BOOTP Client
   UDP: Destination Port = BOOTP Server
   UDP: Total length = 308 (0x134) bytes
   UDP: CheckSum = 0x4A0E
   UDP: Data: Number of data bytes remaining = 300 (0x012C)
 DHCP: Discover      (xid=68256CA8)
   DHCP: Op Code      (op)    = 1 (0x1)
   DHCP: Hardware Type   (htype) = 1 (0x1) 10Mb Ethernet
   DHCP: Hardware Address Length (hlen) = 6 (0x6)
   DHCP: Hops       (hops)  = 0 (0x0)
   DHCP: Transaction ID  (xid)  = 1747283112 (0x68256CA8)
   DHCP: Seconds      (secs)  = 0 (0x0)
   DHCP: Flags       (flags) = 0 (0x0)
     DHCP: 0............... = No Broadcast
   DHCP: Client IP Address (ciaddr) = 0.0.0.0
   DHCP: Your  IP Address (yiaddr) = 0.0.0.0
   DHCP: Server IP Address (siaddr) = 0.0.0.0
   DHCP: Relay IP Address (giaddr) = 0.0.0.0
   DHCP: Client Ethernet Address (chaddr) = 00DD01075715
   DHCP: Server Host Name (sname) = <Blank>
   DHCP: Boot File Name  (file)  = <Blank>
   DHCP: Magic Cookie = [OK]
   DHCP: Option Field   (options)
     DHCP: DHCP Message Type   = DHCP Discover
     DHCP: Client-identifier   = (Type: 1) 00 dd 01 07 57 15
     DHCP: Host Name       = DAVEMAC4
     DHCP: End of this option field
```

There are several interesting points to note in the DHCP discover packet. First, it is sent as a broadcast at both the link layer and the IP layer. Second, the DHCP broadcast flag is set to 0, indicating that the client is capable of receiving a response that is directed to its MAC address (indicated by chaddr). This means that the DHCP server is not required to broadcast the response[14]. Finally, note that there is a transaction ID (XID) used to track each configuration sequence. Any response to this discover packet should reference the same XID.

A DHCP offer follows:

```
*****************************************************************************
Time   Source IP       Dest IP        Prot  Description
0.165  199.199.41.254  199.199.40.13  DHCP  Offer     (xid=68256CA8)

+ FRAME: Base frame properties
 ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
  + ETHERNET: Destination address : 00DD01075715
  + ETHERNET: Source address : 00000C1AEBC5
   ETHERNET: Frame Length : 590 (0x024E)
   ETHERNET: Ethernet Type : 0x0800 (IP: DOD Internet Protocol)
   ETHERNET: Ethernet Data: Number of data bytes remaining = 576
             (0x0240)
 IP: ID = 0x906; Proto = UDP; Len: 576
   IP: Version = 4 (0x4)
   IP: Header Length = 20 (0x14)
  + IP: Service Type = 0 (0x0)
   IP: Total Length = 576 (0x240)
   IP: Identification = 2310 (0x906)
  + IP: Flags Summary = 0 (0x0)
   IP: Fragment Offset = 0 (0x0) bytes
   IP: Time to Live = 31 (0x1F)
   IP: Protocol = UDP - User Datagram
   IP: CheckSum = 0xAF0D
```

[14] Setting this flag to 0 is new in Windows NT 3.51; version 3.5 systems required a broadcast response.

```
   IP: Source Address = 199.199.41.254
   IP: Destination Address = 199.199.40.13
   IP: Data: Number of data bytes remaining = 556 (0x022C)
 UDP: Src Port: BOOTP Server, (67); Dst Port: BOOTP Client (68); Length
      = 556 (0x22C)
 DHCP: Offer       (xid=68256CA8)
   DHCP: Op Code      (op)    = 2 (0x2)
   DHCP: Hardware Type   (htype) = 1 (0x1) 10Mb Ethernet
   DHCP: Hardware Address Length (hlen) = 6 (0x6)
   DHCP: Hops       (hops)  = 0 (0x0)
   DHCP: Transaction ID  (xid)  = 1747283112 (0x68256CA8)
   DHCP: Seconds      (secs)  = 0 (0x0)
   DHCP: Flags       (flags) = 0 (0x0)
     DHCP: 0............... = No Broadcast
   DHCP: Client IP Address (ciaddr) = 0.0.0.0
   DHCP: Your  IP Address (yiaddr) = 199.199.40.13
   DHCP: Server IP Address (siaddr) = 0.0.0.0
   DHCP: Relay IP Address (giaddr) = 199.199.40.1
   DHCP: Client Ethernet Address (chaddr) = 00DD01075715
   DHCP: Server Host Name (sname) = <Blank>
   DHCP: Boot File Name  (file)  = <Blank>
   DHCP: Magic Cookie = [OK]
   DHCP: Option Field   (options)
     DHCP: DHCP Message Type   = DHCP Offer
     DHCP: Subnet Mask      = 255.255.255.0
     DHCP: Renewal Time Value (T1) = 1 Days, 12:00:00
     DHCP: Rebinding Time Value (T2) = 2 Days, 15:00:00
     DHCP: IP Address Lease Time = 3 Days, 0:00:00
     DHCP: Server Identifier   = 199.199.41.254
     DHCP: End of this option field
```

The DHCP offer is also interesting. The XID is the same as that in the discover packet. It is a directed offer, not sent as a broadcast, and it is directed to the MAC address of the client, and to the *proposed* IP address for the client. The source address is from a different subnet (199.199.41) than the subnet that the client is attached to, indicating that both the discover and the offer must have traversed a router. This can be verified by checking the DHCP "giaddr" field, that is set to 199.199.40.1. As you might suspect, a router is configured to forward DHCP broadcasts from this subnet to the one where the DHCP server is located. DHCP forwarding is discussed in RFC1542, and routers used for this purpose must explicitly support the RFC and be configured accordingly[15].

Next, the client accepts the offer:

```
*****************************************************************
Time  Source IP  Dest IP          Prot  Description
0.172 0.0.0.0    255.255.255.255  DHCP  Request   (xid=08186BD1)

+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ IP: ID = 0x100; Proto = UDP; Len: 328
+ UDP: IP Multicast: Src Port: BOOTP Client, (68); Dst Port: BOOTP
       Server (67); Length = 308 (0x134)
 DHCP: Request      (xid=08186BD1)
   DHCP: Op Code      (op)    = 1 (0x1)
   DHCP: Hardware Type   (htype) = 1 (0x1) 10Mb Ethernet
   DHCP: Hardware Address Length (hlen) = 6 (0x6)
   DHCP: Hops       (hops)  = 0 (0x0)
   DHCP: Transaction ID  (xid)  = 135818193 (0x8186BD1)
   DHCP: Seconds      (secs)  = 0 (0x0)
   DHCP: Flags       (flags) = 0 (0x0)
     DHCP: 0............... = No Broadcast
   DHCP: Client IP Address (ciaddr) = 0.0.0.0
   DHCP: Your  IP Address (yiaddr) = 0.0.0.0
   DHCP: Server IP Address (siaddr) = 0.0.0.0
   DHCP: Relay IP Address (giaddr) = 0.0.0.0
   DHCP: Client Ethernet Address (chaddr) = 00DD01075715
```

[15] Although *BOOTP* and DHCP are similar, the Microsoft DHCP server does not support BOOTP. It will silently ignore BOOTP requests.

```
   DHCP: Server Host Name (sname) = <Blank>
   DHCP: Boot File Name  (file)  = <Blank>
   DHCP: Magic Cookie = [OK]
   DHCP: Option Field   (options)
     DHCP: DHCP Message Type   = DHCP Request
     DHCP: Client-identifier   = (Type: 1) 00 dd 01 07 57 15
     DHCP: Requested Address   = 199.199.40.13
     DHCP: Server Identifier   = 199.199.41.254
     DHCP: Host Name       = DAVEMAC4
     DHCP: Parameter Request List = (Length: 7) 01 0f 03 2c 2e 2f 06
     DHCP: End of this option field
```

The request was again broadcast, and the proposed IP address from the server is referenced. The request is broadcast for a reason–the client could have received more than one offer and, by broadcasting its request, it allows the other DHCP servers to see that it isn't going to use their offers.

Finally, the client acknowledges that it will accept the lease:

```
*****************************************************************
Time  Source IP    Dest IP    Prot Description
0.061 199.199.41.254 199.199.40.13 DHCP ACK    (xid=08186BD1)

+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ IP: ID = 0xA06; Proto = UDP; Len: 576
+ UDP: Src Port: BOOTP Server, (67); Dst Port: BOOTP Client (68);
       Length = 556 (0x22C)
 DHCP: ACK         (xid=08186BD1)
   DHCP: Op Code       (op)    = 2 (0x2)
   DHCP: Hardware Type   (htype) = 1 (0x1) 10Mb Ethernet
   DHCP: Hardware Address Length (hlen) = 6 (0x6)
   DHCP: Hops        (hops)  = 0 (0x0)
   DHCP: Transaction ID  (xid)  = 135818193 (0x8186BD1)
   DHCP: Seconds       (secs)  = 0 (0x0)
   DHCP: Flags       (flags) = 0 (0x0)
     DHCP: 0............... = No Broadcast
   DHCP: Client IP Address (ciaddr) = 0.0.0.0
   DHCP: Your  IP Address (yiaddr) = 199.199.40.13
   DHCP: Server IP Address (siaddr) = 0.0.0.0
   DHCP: Relay IP Address (giaddr) = 199.199.40.1
   DHCP: Client Ethernet Address (chaddr) = 00DD01075715
   DHCP: Server Host Name (sname) = <Blank>
   DHCP: Boot File Name  (file)  = <Blank>
   DHCP: Magic Cookie = [OK]
   DHCP: Option Field   (options)
     DHCP: DHCP Message Type   = DHCP ACK
     DHCP: Renewal Time Value (T1) = 1 Days, 12:00:00
     DHCP: Rebinding Time Value (T2) = 2 Days, 15:00:00
     DHCP: IP Address Lease Time = 3 Days, 0:00:00
     DHCP: Server Identifier   = 199.199.41.254
     DHCP: Subnet Mask       = 255.255.255.0
     DHCP: Domain Name       = (Length: 22) 63 73 77 61 74 63 70 2e 6d
           69 63 72 6f 73 6f 66 ...
     DHCP: Router          = 199.199.40.1
     DHCP: NetBIOS Name Service  = 199.199.41.254
     DHCP: NetBIOS Node Type   = (Length: 1) 08
     DHCP: End of this option field
```

The acknowledgment is the final packet of the transaction, and it contains all of the configuration parameters that the client will use.

Lease Expiration and Renewal

DHCP-supplied configurations are "leased" from the server. Periodically, the client will contact the server to renew the lease. The protocol and implementation are very robust and configurable and short-term server or network outages do not generally affect lease renewal. For instance, DHCP clients start to try to renew their lease when 50% of the lease time has expired. Repeated attempts are made to contact the DHCP server and renew the lease, until 87.5% of the lease time has expired. At this point, the client attempts to get a new lease from any available DHCP server.

When a DHCP client is rebooted, it attempts to verify that the lease it holds is valid for the current subnet. If it is moved to another subnet and rebooted, the following sequence takes place:

```
Source MAC  Dest MAC     Source IP   Dest IP          Pro     Description
davemacp    *BROADCAST   0.0.0.0     255.255.255.255  DHCP    Request   (xid=6E3A2E74)
router      *BROADCAST   10.57.8.1   255.255.255.255  DHCP    NACK      (xid=6E3A2E74)
davemacp    *BROADCAST   0.0.0.0     255.255.255.255  DHCP    Discover  (xid=51CA7FED)
router      davemacp     10.57.8.1   10.57.13.152     DHCP    Offer     (xid=51CA7FED)
davemacp    *BROADCAST   0.0.0.0     255.255.255.255  DHCP    Request   (xid=2081237D)
router      davemacp     10.57.8.1   10.57.13.152     DHCP    ACK       (xid=2081237D)
```

In this example the portable computer "davemacp" was moved to a new subnet and re-started. It broadcasted a DHCP request for renewal of its old parameters, but the DHCP server responsible for the new subnet recognized that these were invalid for the subnet and NAK'd them. The DHCP client software automatically went through a normal discovery process to get reconfigured with parameters that are valid for the new location.

Windows Internet Name Service (WINS)

WINS is a NetBIOS name service as described in RFC1001/RFC1002[16]. When a Windows NT system is configured as an h-node (default for WINS clients), it attempts to use a WINS server for name registration and resolution first and, if that fails, it resorts to subnet broadcasts.

WINS Name Registration and Resolution

Using WINS for name services dramatically reduces the number of IP broadcasts used by Microsoft network clients. The trace snippet below illustrates name registration and resolution traffic caused by booting a Windows NT 3.51 workstation.

```
Source IP      Dest IP        Prot Description
199.199.40.124 199.199.41.254 NBT  NS: MultiHomed Name Registration req. for DAVEMAC4<00>
199.199.41.254 199.199.40.124 NBT  NS: Registration resp. for DAVEMAC4<00>, Success
199.199.40.124 199.199.41.254 NBT  NS: Registration req. for DAVEMACD<00>
199.199.41.254 199.199.40.124 NBT  NS: Registration resp. for DAVEMACD<00>, Success
199.199.40.124 199.199.41.254 NBT  NS: Query req. for DAVEMACD<1C>
199.199.41.254 199.199.40.124 NBT  NS: Query resp. for DAVEMACD<1C>, Success
199.199.40.124 199.199.41.254 NBT  NS: MultiHomed Name Registration req. for DAVEMAC4<03>
199.199.41.254 199.199.40.124 NBT  NS: Registration resp. for DAVEMAC4<03>, Success
```

[16] The Microsoft WINS server is discussed in more detail in a separate whitepaper (Part No. 098-56544).

This trace shows that the booting client (199.199.40.124) sends a single name registration request to the WINS server, asking to register the computer name (DAVEMAC4<00>) as a unique name for a multi-homed host. The WINS server responds affirmatively. Next, the domain name (DAVEMACD<00>) is registered as a group name. Then a name query is sent to the WINS server, requesting a list of domain controllers (who all register the <domain>[1C] name) so that a logon server can be contacted. One more registration is shown, for DAVEMAC4<03>, which is the name registered by the messenger service. The fully parsed version of the domain name registration is shown below.

```
*****************************************************************************
Source IP       Dest IP         Prot  Description
199.199.40.124  199.199.41.254  NBT   NS: Registration req. for
DAVEMACD<00>

+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ IP: ID = 0x300; Proto = UDP; Len: 96
+ UDP: Src Port: NETBIOS Name Service, (137); Dst Port: NETBIOS Name
       Service (137); Length = 76 (0x4C)
 NBT: NS: Registration req. for DAVEMACD<00>
   NBT: Transaction ID = 32770 (0x8002)
   NBT: Flags Summary = 0x2900 - Req.; Registration; Success
     NBT: 0............... = Request
     NBT: .0101........... = Registration
     NBT: .....0.......... = Non-authoritative Answer
     NBT: ......0......... = Datagram not truncated
     NBT: .......1........ = Recursion desired
     NBT: ........0....... = Recursion not available
     NBT: .........0...... = Reserved
     NBT: ..........0..... = Reserved
     NBT: ...........0.... = Not a broadcast packet
     NBT: ............0000 = Success
   NBT: Question Count = 1 (0x1)
   NBT: Answer Count = 0 (0x0)
   NBT: Name Service Count = 0 (0x0)
   NBT: Additional Record Count = 1 (0x1)
   NBT: Question Name = DAVEMACD<00>
   NBT: Question Type = General Name Service
   NBT: Question Class = Internet Class
   NBT: Resource Record Name = DAVEMACD<00>
   NBT: Resource Record Type = NetBIOS General Name Service
   NBT: Resource Record Class = Internet Class
   NBT: Time To Live = 300000 (0x493E0)
   NBT: RDATA Length = 6 (0x6)
   NBT: Resource Record Flags = 57344 (0xE000)
     NBT: 1............... = Group NetBIOS Name
     NBT: .11............. = Reserved
     NBT: ...0000000000000 = Reserved
   NBT: Owner IP Address = 199.199.40.13
```

Since the domain name is a group name, any number of hosts are allowed to register it.

WINS in a DHCP Environment

WINS is especially helpful on DHCP-enabled networks. One of the DHCP-provided parameters can be the address of a WINS server, so as soon as the client is configured by DHCP, it registers its name(s) and address with the WINS server, and can then be easily located by the other computers on the network. This combination of DHCP and WINS is ideal for dynamic situations.

Domain Name System (DNS)

Microsoft included a Beta version of a DNS server for Windows NT 3.5 in the Windows NT 3.5 resource kit, and one for Windows NT 3.51 in the Windows NT 3.51 resource kit[17]. These versions of the DNS have an interesting feature partial integration of WINS and DNS namespaces.

Integration of the DNS and WINS

The DNS can be configured with a special "$WINS" directive, that instructs it to pass through to a WINS server queries for names that can't be found in the DNS database. The algorithm for passing through queries is:

- Attempt to resolve the name to an IP address using standard DNS records.
- If that fails, the domain suffix is stripped off, and a NetBIOS name is formed by space padding the hostname and appending 0x00 as the 16th character. The WINS server configured for the host running the DNS software is then sent a standard NetBIOS name query for that name.
- The WINS server returns a Name Query Response, and if it was successful the DNS returns a standard DNS response to the original client using the information obtained from WINS.

The following example is presented to clarify this:

- Host "A" sends a DNS query for beetle.microsoft.com to the DNS.
- The DNS checks its records and finds no match.
- The DNS sends a NetBIOS Name Query to its WINS server for "BEETLE<00>" (BEETLE followed by 9 spaces and hex 00).
- The WINS server locates the (dynamically registered) IP address for BEETLE and returns it to the DNS in a NetBIOS Name Query Response.
- The DNS returns the resource record to Host "A" as a DNS response.

Since WINS is a dynamic name service, this feature extends the DNS and can provide relief from some of the manual labor normally associated with DNS administration.

The Browser

The browser was originally designed to be a simple workgroup enumeration tool, but has been enhanced significantly over time. As of Windows NT 3.5, the browser is WAN-aware and can be used to provide configuration-free enumeration of resources that are anywhere in an IP internetwork. The Windows NT Resource Kit Networking Guide contains an overview of the browser, but a few key concepts related to WAN browsing are discussed here.

Master Browser Elections

A Master Browser is elected on each subnet with Microsoft networking computers present. In addition, the PDC for a domain always functions as the Domain Master Browser, which is responsible for replicating the browse lists amongst all Master Browsers within the domain. Each domain has one Master Browser per subnet that it

[17] For information on the latest DNS builds, mail "dnsbeta@microsoft.com".

has member computers on, listening for server announcements from Windows NT-based, Windows for Workgroups-based, and Lan Manager-based systems. It maintains lists of available resources that can be requested by client computers.

As the number of hosts on a subnet grows, the Master Browser will start to replicate the browse list to Backup Browsers. If the master is shut down, an election takes place to determine the new Master Browser. Existing Backup Browsers have an advantage in the election. For this process, workgroups and domains function alike, except that all Windows NT Servers are either a Master Browser or Backup Browser, and Windows NT Workstation and Windows for Workgroups computers aren't allowed to become backups or masters unless specifically configured.

Master Browser elections take place over the special <domain>[1E] NetBIOS name using subnet broadcasts (without using WINS). The election is fully automatic and takes into consideration a number of heuristics: operating system, version number, uptime, role (Workstation, Backup Domain Controller, Primary Domain Controller), etc. In general, the most robust system on the network wins. Elections are forced when:

- A client cannot find its master browser at startup.
- A client detects that a master browser has disappeared.
- A Windows NT Server initializes.

Maintaining Browse Lists

File servers periodically (once every 12 minutes) announce their presence to the special <domain>[1D] NetBIOS name in an IP subnet broadcast. The Master Browser builds a list from these broadcasts. In addition, all Master Browsers register a group name \0x01\0x02__MSBROWSE__\0x02\0x01on the local subnet (not with WINS). Periodically the Master Browsers in the domains and workgroups announce their presence to this special name. Thus, in addition to the workgroup or domain membership lists, Master Browsers also maintain lists of other domains with their associated Master Browsers.

Requesting Browse Lists

When a browse request is made from a client, a "GetBackupListRequest" is sent to the <domain>[1D] name (the Master Browser) that returns a list of browser servers for the local subnet. The "GetBackupListRequest" is also unicast to the Domain Master Browser, which handles the case in which the queried domain has no members on the subnet. The client browser service selects three of the browsers from the list and stores them for future use. Then when further browsing is done, (via the *NetServerEnum* API) one of the three saved names is contacted by the client.

When a client queries its workgroup or domain browser, it first gets back a list of all of the domains and workgroups that the browser has learned about through the \0x01\0x02__MSBROWSE__\0x02\0x01 name as well as the name of the master browser for each. When the user expands a domain or workgroup into a membership list, the client sends a request to <domain>[1D] to get to the list (this is translated to a local subnet broadcast by wins). If this fails, it contacts the master browser for the particular domain or workgroup and fetches the membership list.

The Domain Master Browser

As mentioned earlier, the PDC always acts as the *Domain Master Browser*. Since each locally-elected Master Browser will only hear local membership announcements, there needs to be a mechanism to consolidate all of the members into a single list. This is the role of the Domain Master Browser. Periodically, all of the locally-elected Master Browsers contact the PDC and replicate their membership lists to it. The PDC merges the list with the "master" list for the whole domain and replicates the master list back down. The replication algorithm is smart in that the local Master Browsers only replicate the members that they have learned about locally to the domain master. This whole mechanism allows members in a domain to span subnets and for all clients (eventually) to be able to get complete membership lists.

On WINS-enabled networks, the browser code in NT 3.5*x* periodically connects to WINS and learns all of the systems that have registered any <domain>[1B] name. The browser then does a GetDCName() on each of the <domain>[1B]names (followed by an attempt on <domain>[1C]) and adds the <domain name> <master browser name> to its domain/workgroup list. This allows members of one domain to locate the master browser for another domain even when it is on another subnet and the two domains have no “broadcast area” in common.

Browser Enhancements

Browser code for Windows for Workgroups computers has been enhanced several times over the past year to reduce the dependency on having a BDC per subnet. The updated files are included on Windows NT Server 3.5 and 3.51 CDs under the \clients\wfw directory, and are available from ftp.microsoft.com. Windows 95 systems also contain enhanced browsing code.

Windows NT Workstation and Windows NT Server Services

The workstation and server services are used for file and print sharing. Both use NetBIOS over TCP/IP to communicate with each other; however, *they are not NetBIOS applications . . . they are written to talk directly to NetBT over the TDI interface.* Being direct TDI clients, they are high performance and *not* subject to limitations of the NetBIOS interface, such as the 254 session limit. The Server Message Block (SMB) protocol is used to send commands and responses between clients and servers. Public SMB specifications are available from ftp.microsoft.com.

Logging On

When a user logs on to a Windows NT domain, a name query is sent to the NetBIOS <domain>[1C] name. All domain controllers register this name on the network, typically with the WINS database. If the client logging on is WINS-enabled, then the name query is unicast to its WINS server, which responds with a list of IP addresses for up to 25 domain controllers. The logon process selects one of the addresses from the list and uses it to authenticate the user. All password information is encrypted before being transmitted on the network.

Connecting to Network Resources

When a workstation attempts to connect to a shared resource on the network, the resource is "called" by NetBIOS name. The address resolution is accomplished as illustrated in figures 3 and 4 in the NetBIOS section of this document. Once the IP address of the target host is known, a standard TCP/IP connection is set up, and a NetBIOS session is established over that connection[18]. The user is authenticated using encrypted passwords, and then client/server messages are exchanged using the SMB protocol. The workstation and server use sophisticated caching mechanisms to reduce network traffic and provide high performance. When WINS is used there is no reliance on IP broadcasts, with the single exception of ARPs.

Optimizations

The Windows NT Workstation and Windows NT Server services were designed with many optimizations to minimize network traffic and maximize throughput. The network redirector works closely with the Windows NT Cache Manager to provide read-ahead caching, write-behind caching, and search caching. Various file locking schemes, such as opportunistic locking and local file lock optimization, help to reduce network traffic. The SMB protocol used supports compound commands and responses, such as *LockAndRead* and *WriteAndUnlock*.

Microsoft Remote Access PPP/SLIP Support

Windows NT 3.5x Remote Access Server (RAS) includes client and server support for Point-to-Point Protocol (PPP), and client-only support for Serial Line IP (SLIP)[19]. Microsoft recommends that you use PPP because of its flexibility and its role as an industry standard, and for future flexibility with client and server hardware and software. In addition, Microsoft RAS servers can act as NetBIOS gateways for clients that dial in using the proprietary RAS protocol and NetBEUI, providing access to NetBIOS resources over NetBEUI, IPX, or TCP/IP. Windows NT server RAS supports up to 256 simultaneous remote clients, and Windows NT Workstation RAS supports only one remote client at a time.

RAS Servers

RAS servers act as a "proxy" for their remote TCP/IP clients on the network that they are attached to. They use proxy ARP to respond to ARP requests for their clients, and set up host routes to each of their clients from the network. RAS servers can obtain configuration parameters for their clients from a DHCP server, and then use PPP IPCP (Internet Protocol Control Protocol) as defined in RFC1332 to configure their clients with these parameters dynamically over the RAS link[20].

---

[18] NOTE: Since there are several non-broadcast options for NetBIOS name resolution, the Workstation and Server services can be used over any IP network such as the Internet. You can connect to the \\ftp\data public share on ftp.microsoft.com from another Windows NT system on the Internet with file manager by using the following lmhosts entry:

198.105.232.1 ftp #PRE

[19] RAS is discussed in more detail in a separate whitepaper, Part No. 098-57330.

[20] When a RAS server is configured to use DHCP to obtain TCP/IP configuration parameters for its clients, a pool of leased addresses is obtained from the DHCP server, maintained in the registry, and managed locally by the RAS server. If more addresses are needed, or leases need to be renewed, the RAS server will contact the DHCP server; however it does not check with the DHCP server each time a client dials in. If the RAS server is moved to another subnet, it may have a pool of leases that are not valid for the new subnet still stored in the registry until they expire. See KnowledgeBase Article Q124358 for details on manually removing these leases.

RAS Clients

RAS clients using TCP/IP may be configured to use the default gateway on the remote network while they are connected to a PPP server. If so, then this default gateway overrides any default gateway that is configured for local networks while the RAS connection is established. The override is accomplished by manipulating the IP route table. Any local routes, including the default gateway, get their metric (hop count) incremented by one, and a default route with a metric of 1 hop is dynamically added for the duration of the connection. One-hop routes are also added for the IP multicast address (224.0.0.0), for the local WAN interface, and for the network that the PPP server is attached to. This can present a problem with connecting to resources via the local network default gateway, unless static routes are added at the client. Sample route tables for a Windows NT workstation before and after connecting to a remote network using PPP are shown below:

```
Route table before dialing a PPP Internet provider:

Network Address Netmask         Gateway Address  Interface      Metric
0.0.0.0         0.0.0.0         199.199.40.1     199.199.40.11  1
127.0.0.0       255.0.0.0       127.0.0.1        127.0.0.1      1
199.199.40.0    255.255.255.0   199.199.40.11    199.199.40.11  1
199.199.40.11   255.255.255.255 127.0.0.1        127.0.0.1      1
199.199.40.255  255.255.255.255 199.199.40.11    199.199.40.11  1
224.0.0.0       224.0.0.0       199.199.40.11    199.199.40.11  1
255.255.255.255 255.255.255.255 199.199.40.11    199.199.40.11  1

Route table after dialing a PPP Internet provider:

Network Address Netmask          Gateway Address Interface      Metric
0.0.0.0         0.0.0.0          199.199.40.1    199.199.40.11  2
0.0.0.0         0.0.0.0          204.182.66.83   204.182.66.83  1
127.0.0.0       255.0.0.0        127.0.0.1       127.0.0.1      1
199.199.40.0    255.255.255.0    199.199.40.11   199.199.40.11  2
199.199.40.11   255.255.255.255  127.0.0.1       127.0.0.1      1
199.199.40.255  255.255.255.255  199.199.40.11   199.199.40.11  1
204.182.66.0    255.255.255.0    204.182.66.83   204.182.66.83  1
204.182.66.83   255.255.255.255  127.0.0.1       127.0.0.1      1
224.0.0.0       224.0.0.0        204.182.66.83   204.182.66.83  1
224.0.0.0       224.0.0.0        199.199.40.11   199.199.40.11  1
255.255.255.255 255.255.255.255  199.199.40.11   199.199.40.11  1
```

Using RAS To Route Between Networks

RAS was primarily designed to allow individual network clients to gain access to services on a remote network, not to link networks together. However, in some applications it is possible to use a RAS server to link a small network to a larger one, such as the Internet. See KnowledgeBase article Q121877 for configuration details. Routing support for RAS in more complex networks will be greatly improved when RIP becomes available from Microsoft. Currently RIP is available as a public Beta from ftp.microsoft.com.

Bandwidth Considerations

By default, RAS utilizes effective compression methods to increase the amount of data that may be pumped over a serial link. When designing and installing systems and services using RAS, it is important to do bandwidth planning. As a rule of thumb, transfer rates can be estimated using the 10-bit byte to allow for protocol and timing overhead. For instance, 9600 BPS (without compression) is approximately 1

Kbyte/second, 60Kbytes/minute, and 3.5Mbytes/hour. If the data being transferred compresses fairly well, 5-8 Mbytes per hour throughput might be expected. While this may be an adequate rate for a single workstation, it probably is not feasible as an inter-site link for most applications. ISDN (128Kbits/second or 45 Mbytes/hour, not including compression) might be more realistic. ISDN service in the United States has become more available and economical in the past year.

Simple Network Management Protocol (SNMP) Agent

The SNMP agent in Windows NT provides some programmatic access to the TCP/IP protocol stack. The Windows NT Software Development Kit (SDK) includes the Microsoft Windows NT SNMP Programmers Reference document. There is a MIB compiler (MIBCC.EXE) included in the Windows NT Resource Kit, and the Microsoft KnowledgeBase contains at least one article describing its usage. The resource kit also contains mib_ii.mib, lmmib2.mib, and smi.mib.

TCP/IP Troubleshooting Tools and Strategies

Overview

Many excellent network troubleshooting tools are available for Windows NT. Most are included in the product or the Windows NT Resource Kit. The binaries from the Resource Kit are available at no charge from the Microsoft Internet site. *Microsoft Network Monitor* is an excellent network tracing tool that is included in the Microsoft Systems Management Server product.

When troubleshooting any problem, it's helpful to use a logical approach. Some questions to ask are:

- What does work?
- What doesn't work?
- How are the things that do and don't work related?
- Have the things that don't work ever worked on this computer/network?
- If so, what has changed since it last worked?

Troubleshooting a problem "from the bottom up" is often a good way to quickly isolate it. The tools listed below are organized in this manner.

IPConfig

IPConfig is a command-line utility that prints out the TCP/IP-related configuration of a host[21]. When used with the ***/all*** switch, it produces a detailed configuration report for all interfaces, including any configured serial ports (RAS). Output may be redirected to a file and pasted into other documents as shown below:

```
Windows NT IP Configuration
    Host Name . . . . . . . . . . : davemac1.microsoft.com
    DNS Servers . . . . . . . . . :
    Node Type . . . . . . . . . . : Hybrid
    NetBIOS Scope ID. . . . . . . :
    IP Routing Enabled. . . . . . : No
    WINS Proxy Enabled. . . . . . : No
    NetBIOS Resolution Uses DNS : No

Ethernet adapter Elnk31:
    Description . . . . . . . . . : ELNK3 Ethernet Adapter.
    Physical Address. . . . . . . : 00-20-AF-1D-2B-91
    DHCP Enabled. . . . . . . . . : Yes
    IP Address. . . . . . . . . . : 10.57.9.138
    Subnet Mask . . . . . . . . . : 255.255.248.0
    Default Gateway . . . . . . . : 10.57.8.1
    DHCP Server . . . . . . . . . : 10.54.16.157
    Primary WINS Server . . . . . : 10.54.16.157
    Secondary WINS Server . . . . : 10.54.16.159
    Lease Obtained. . . . . . . . : Sunday, June 25, 1995 11:43:01
    PM
    Lease Expires . . . . . . . . : Wednesday, June 28, 1995
    11:43:01 PM

Ethernet adapter NdisWan5:
    Description . . . . . . . . . :
```

[21] NOTE: Windows 95 TCP/IP includes WINIPCFG in place of IPCONFIG.

```
Physical Address. . . . . : 00-00-00-00-00-00
DHCP Enabled. . . . . : No
IP Address. . . . . : 0.0.0.0
Subnet Mask . . . . . : 0.0.0.0
Default Gateway . . . . . :
```

Ping

Ping is a tool that helps to verify IP-level connectivity. When troubleshooting, the *ping* command is used to send an ICMP echo request to a target name or IP address. First try pinging the IP address of the target host to see if it will respond, as this is the simplest case. If that succeeds, then try pinging the name. Ping uses Windows Sockets-style name resolution to resolve the name to an address, so if pinging by address succeeds, but by name fails, then the problem lies in address resolution, not network connectivity. Type ***ping -?*** to see what command-line options are available. For example, ping allows you to specify the size of packets to use, how many to send, whether to record the route used, what TTL value to use, and whether to set the "don't fragment" flag. See the PMTU discovery section of this document for details on using ping to manually determine the PMTU between two computers.

The following example illustrates how to send two pings, each 1450 bytes in size, to address 10.57.13.152:

```
C:\>ping -n 2 -l 1450 10.57.13.152
Pinging 10.57.13.152 with 1450 bytes of data:

Reply from 10.57.13.152: bytes=1450 time=10ms TTL=32
Reply from 10.57.13.152: bytes=1450 time=10ms TTL=32
```

By default, ping only waits 750ms for each response to be returned before timing out. If the remote system being pinged is across a high-delay link such as a satellite link, responses could take longer to be returned. The **-w (wait)** switch can be used to specify a longer timeout.

ARP

The ***ARP*** command is useful for viewing the ARP cache. If two hosts on the same subnet cannot even ping each other successfully, try running the ***arp -a*** command on each computer to see if they have the correct MAC addresses listed for each other. You can determine a host's MAC address using IPConfig. If another host with a duplicate IP address exists on the network, the ARP cache may have had the MAC address for the other computer placed in it. ***Arp -d*** can be used to delete an entry that may be incorrect. Entries can be added using ***arp -s***.

Tracert

Tracert is a route tracing utility. Tracert uses the IP TTL field and ICMP error messages to determine the route from one host to another through a network. Sample output from the tracert command is shown in the ICMP section of this document.

Route

Route is used to view or modify the route table. ***Route print*** displays a list of current routes known by IP for the host. Sample output is shown in the IP section of this document. ***Route add*** is used to add routes to the table, and ***route delete*** is used to delete routes from the table. Note that routes added to the table are not made permanent unless the ***-p*** switch is specified. Non-persistent routes only last until the computer is rebooted.

In order for two hosts to exchange IP datagrams, they must both have a route to each other, or use default gateways that know of a route. Normally, routers exchange information with each other using a protocol such as Routing Information Protocol (RIP) or Open Shortest Path First (OSPF). Windows NT 3.5*x* did not include support for either of these routing protocols, so when these computers are used as routers it is often necessary to manually add routes. Microsoft is working on RIP and OSPF support for Windows NT.

Netstat

Netstat displays protocol statistics and current TCP/IP connections. ***Netstat -a*** displays all connections, and ***netstat -r*** displays the route table, plus active connections. The **-n** switch tells netstat not to convert addresses and port numbers to names. Sample output is shown below:

```
C:\>netstat -e
Interface Statistics

                           Received      Sent
Bytes                      3995837940    47224622
Unicast packets            120099        131015
Non-unicast packets        7579544       3823
Discards                   0             0
Errors                     0             0
Unknown protocols          363054211

C:\>netstat -a

Active Connections

  Proto Local Address      Foreign Address    State
  TCP   davemac1:1572      10.57.13.152:nbsession ESTABLISHED
  TCP   davemac1:1589      10.57.9.147:nbsession ESTABLISHED
  TCP   davemac1:1606      11.1.105.245:nbsession ESTABLISHED
  TCP   davemac1:1632      10.57.9.213:nbsession ESTABLISHED
  TCP   davemac1:1659      10.55.86.169:nbsession ESTABLISHED
  TCP   davemac1:1714      10.55.80.203:nbsession ESTABLISHED
  TCP   davemac1:1719      10.54.67.36:nbsession ESTABLISHED
  TCP   davemac1:1241      10.57.9.101:nbsession ESTABLISHED
  UDP   davemac1:1025      *:*
  UDP   davemac1:snmp      *:*
  UDP   davemac1:nbname    *:*
  UDP   davemac1:nbdatagram *:*
  UDP   davemac1:nbname    *:*
  UDP   davemac1:nbdatagram *:*

C:\>netstat -s

IP Statistics

  Packets Received            = 5378528
  Received Header Errors      = 738854
  Received Address Errors     = 23150
```

```
 Datagrams Forwarded          = 0
 Unknown Protocols Received   = 0
 Received Packets Discarded   = 0
 Received Packets Delivered   = 4616524
 Output Requests              = 132702
 Routing Discards             = 157
 Discarded Output Packets     = 0
 Output Packet No Route       = 0
 Reassembly Required          = 0
 Reassembly Successful             = 0
 Reassembly Failures               = 0
 Datagrams Successfully Fragmented = 0
 Datagrams Failing Fragmentation   = 0
 Fragments Created                 = 0

ICMP Statistics
                          Received  Sent
 Messages                 693       4
 Errors                   0         0
 Destination Unreachable  685       0
 Time Exceeded            0         0
 Parameter Problems       0         0
 Source Quenchs           0         0
 Redirects                0         0
 Echos                    4         0
 Echo Replies             0         4
 Timestamps               0         0
 Timestamp Replies        0         0
 Address Masks            0         0
 Address Mask Replies     0         0

TCP Statistics

 Active Opens                = 597
 Passive Opens               = 135
 Failed Connection Attempts  = 107
 Reset Connections           = 91
 Current Connections         = 8
 Segments Received           = 106770
 Segments Sent               = 118431
 Segments Retransmitted      = 461

UDP Statistics

 Datagrams Received   = 4157136
 No Ports             = 351928
 Receive Errors       = 2
 Datagrams Sent       = 13809
```

NBTStat

NBTStat is a useful tool for troubleshooting NetBIOS name resolution problems. ***NBTStat -n*** displays the names that were registered locally on the system by applications, such as the server and redirector. ***NBTStat -c*** shows the NetBIOS name cache, which contains name-to-address mappings for other computers. ***NBTStat -R*** purges the name cache and reloads it from the *LMHOSTS* file. ***NBTStat -a <name>*** performs a NetBIOS adapter status command against the computer specified by ***name.*** The adapter status command returns the local NetBIOS name table for that computer plus the MAC address of the adapter card. ***NBTStat -S*** lists the current NetBIOS sessions and their status, including statistics, as shown:

```
NetBIOS Connection Table

Local Name     State      In/Out Remote Host    Input    Output
-------------------------------------------------------------------
DAVEMAC1 <00> Connected Out     CNSSUP1<20>    6MB      5MB
DAVEMAC1 <00> Connected Out     CNSPRINT<20>   108KB    116KB
DAVEMAC1 <00> Connected Out     CNSSRC1<20>    299KB    19KB
DAVEMAC1 <00> Connected Out     STH2NT<20>     324KB    19KB
DAVEMAC1 <03> Listening
```

Performance Monitor

The Windows NT Performance Monitor can be used to view many different TCP/IP-related counters. Since it accesses statistics that have been gathered by the SNMP agent, *the SNMP agent must be installed on computers that are to be monitored*. Counters are available for the NIC, IP, ICMP, UDP, TCP, and NBT. One of the features of Performance Monitor is that it allows counters from various systems to be monitored from a single management window. It also supports setting alert levels for the counters being monitored. The Windows NT Resource Kit book "Optimizing Windows NT" is an excellent resource on using Performance Monitor effectively.

Figure 5 shows a sample Performance Monitor window used for monitoring ftp.microsoft.com. Each of the counters has a scaling factor shown at the left of the counter description. There are 100 sample points on the chart taken at three-second intervals. During the five minutes shown here, the server averaged 1308 packets per second at the NIC (all protocols), and about 1000 IP datagrams per second. There was an average of approximately 450 TCP connections present for the sample period.

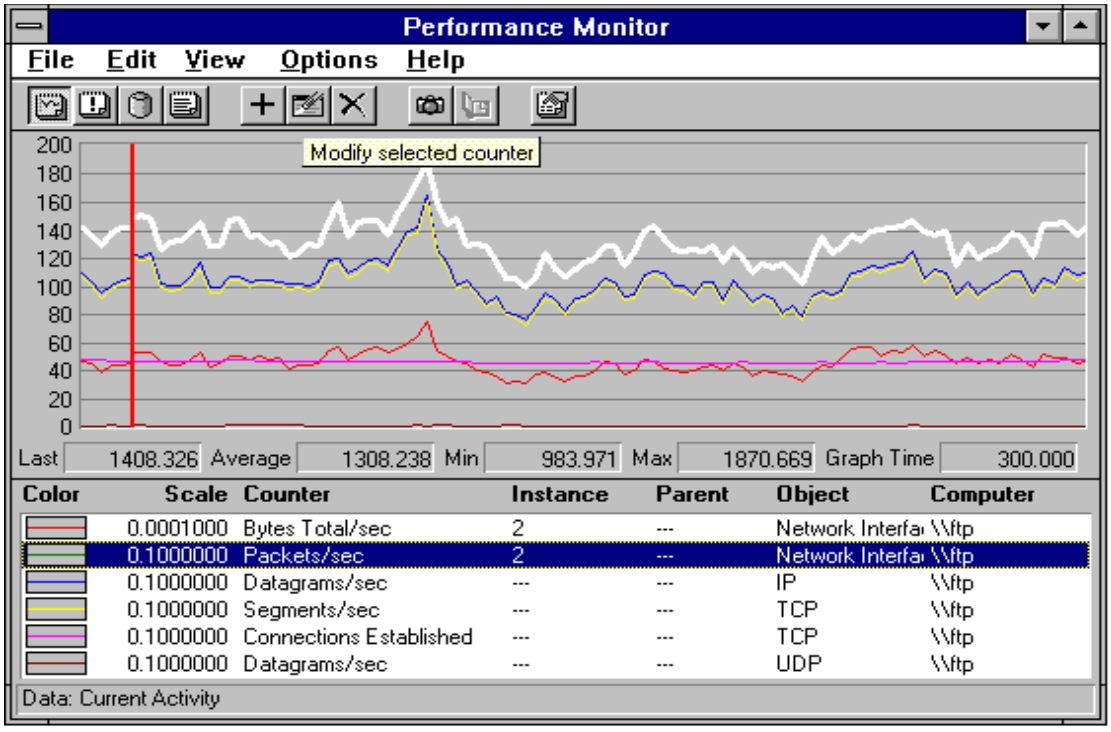

Figure 5: Performance Monitor

Microsoft Network Monitor

Microsoft Network Monitor is a tool developed by Microsoft to make the task of troubleshooting complex network problems much easier and more economical. It is packaged as part of the Microsoft Systems Management Server product but can be used as a standalone network monitor. In addition, Windows NT and Windows 95 distribution media includes *Network Monitor Agent* software. Stations running Network Monitor can attach to stations running the agent software over the network or using dial-up (RAS) to perform monitoring or tracing of remote network segments. This can be a very useful troubleshooting tool.

Network Monitor works by placing the NIC on the capturing host into "promiscuous" mode so that it passes up every frame seen on the wire to the tracing tool. Capture filters can be defined so that only specific frames are saved for analysis. Filters can be defined based on source and destination NIC addresses, source and destination protocol addresses, and pattern matches. Once a capture has been obtained, display filtering can be used to further narrow down a problem. Display filtering allows specific protocols to be selected as well.

Once a capture has been obtained and filtered, Network Monitor protocol parsing interprets the binary trace data into readable terms using parsing DLLs. A sample SMB (Server Message Block) frame is shown fully parsed below:

```
********************************************************************************
Frame Time  Src Other Addr Dst Other Addr Protocol Description
7     0.020 10.57.9.138    10.57.13.152   SMB      C get attributes, File = \temp

 FRAME: Base frame properties
   FRAME: Time of capture = Jun 27, 1995 8:11:11.636
   FRAME: Time delta from previous physical frame: 3 milliseconds
   FRAME: Frame number: 7
   FRAME: Total frame length: 106 bytes
   FRAME: Capture frame length: 106 bytes
   FRAME: Frame data: Number of data bytes remaining = 106 (0x006A)
 ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
   ETHERNET: Destination address : 00608C0E6C6A
     ETHERNET: .......0 = Individual address
     ETHERNET: ......0. = Universally administered address
   ETHERNET: Source address : 0020AF1D2B91
     ETHERNET: .......0 = No routing information present
     ETHERNET: ......0. = Universally administered address
   ETHERNET: Frame Length : 106 (0x006A)
   ETHERNET: Ethernet Type : 0x0800 (IP: DOD Internet Protocol)
   ETHERNET: Ethernet Data: Number of data bytes remaining = 92 (0x005C)
 IP: ID = 0x4072; Proto = TCP; Len: 92
   IP: Version = 4 (0x4)
   IP: Header Length = 20 (0x14)
   IP: Service Type = 0 (0x0)
     IP: Precedence = Routine
     IP: ...0.... = Normal Delay
     IP: ....0... = Normal Throughput
     IP: .....0.. = Normal Reliability
   IP: Total Length = 92 (0x5C)
   IP: Identification = 16498 (0x4072)
   IP: Flags Summary = 2 (0x2)
     IP: .......0 = Last fragment in datagram
     IP: ......1. = Cannot fragment datagram
   IP: Fragment Offset = 0 (0x0) bytes
   IP: Time to Live = 32 (0x20)
   IP: Protocol = TCP - Transmission Control
   IP: CheckSum = 0xC895
   IP: Source Address = 10.57.9.138
   IP: Destination Address = 10.57.13.152
   IP: Data: Number of data bytes remaining = 72 (0x0048)
```

```
TCP: .AP..., len:  52, seq: 344830227, ack:  2524988, win: 8166, src: 1677 dst: (NBT
     Session)
  TCP: Source Port = 0x068D
  TCP: Destination Port = NETBIOS Session Service
  TCP: Sequence Number = 344830227 (0x148DB113)
  TCP: Acknowledgement Number = 2524988 (0x26873C)
  TCP: Data Offset = 20 (0x14)
  TCP: Reserved = 0 (0x0000)
  TCP: Flags = 0x18 : .AP...
    TCP: ..0..... = No urgent data
    TCP: ...1.... = Acknowledgement field significant
    TCP: ....1... = Push function
    TCP: .....0.. = No Reset
    TCP: ......0. = No Synchronize
    TCP: .......0 = No Fin
  TCP: Window = 8166 (0x1FE6)
  TCP: CheckSum = 0xC072
  TCP: Urgent Pointer = 0 (0x0)
  TCP: Data: Number of data bytes remaining = 52 (0x0034)
NBT: SS: Session Message, Len: 48
  NBT: Packet Type = Session Message
  NBT: Packet Flags = 0 (0x0)
    NBT: .......0 = Add 0 to Length
  NBT: Packet Length = 48 (0x30)
  NBT: SS Data: Number of data bytes remaining = 48 (0x0030)
SMB: C get attributes, File = \temp
  SMB: SMB Status = Error Success
    SMB: Error class = No Error
    SMB: Error code = No Error
  SMB: Header: PID = 0xCAFE TID = 0x0800 MID = 0x43C0 UID = 0x0800
    SMB: Tree ID   (TID) = 2048 (0x800)
    SMB: Process ID  (PID) = 51966 (0xCAFE)
    SMB: User ID   (UID) = 2048 (0x800)
    SMB: Multiplex ID (MID) = 17344 (0x43C0)
    SMB: Flags Summary = 24 (0x18)
      SMB: .......0 = Lock & Read and Write & Unlock not supported
      SMB: ......0. = Send No Ack not supported
      SMB: ....1... = Using caseless pathnames
      SMB: ...1.... = Canonicalized pathnames
      SMB: ..0..... = No Opportunistic lock
      SMB: .0...... = No Change Notify
      SMB: 0....... = Client command
    SMB: flags2 Summary = 32771 (0x8003)
      SMB: ...............1 = Understands long filenames
      SMB: ..............1. = Understands extended attributes
      SMB: ..0.............. = No paging of IO
      SMB: .0............... = Using SMB status codes
      SMB: 1................ = Using UNICODE strings
  SMB: Command = C get attributes
    SMB: Word count = 0
    SMB: Byte count = 13
    SMB: Byte parameters
    SMB: Path name = \temp

00000: 00 60 8C 0E 6C 6A 00 20 AF 1D 2B 91 08 00 45 00  .`..lj. ..+...E.
00010: 00 5C 40 72 40 00 20 06 C8 95 9D 39 09 8A 9D 39  .\@r@. ....9...9
00020: 0D 98 06 8D 00 8B 14 8D B1 13 00 26 87 3C 50 18  ...........&.<P.
00030: 1F E6 C0 72 00 00 00 00 00 30 FF 53 4D 42 08 00  ...r.....0.SMB..
00040: 00 00 00 18 03 80 00 00 00 00 00 00 00 00 00 00  ................
00050: 00 00 00 08 FE CA 00 08 C0 43 00 0D 00 04 5C 00  .........C....\.
00060: 74 00 65 00 6D 00 70 00 00 00           t.e.m.p...
```

The parsed output consists of three sections–a summary window, a detailed description window, and the hex output. If traces are to be sent to support personnel at Microsoft, they are most useful in electronic form rather than printed form, as they can be manipulated and scanned electronically. Large printed traces are time-consuming to read.

The Microsoft KnowledgeBase (KB)

The Microsoft KB is an excellent source of information on all aspects of Windows NT. It contains thousands of articles written by the support professionals in the Corporate Network Systems unit at Microsoft. Articles are updated daily, and topics include:

- Installation and configuration information.
- Status on known problems and fixes.
- Service Pack updates.
- Technology discussions
- Troubleshooting tips.
- Hardware-specific information.

The Microsoft KB is available from many different sources, including: the Internet (full-text search capabilities for WWW browsers on www.microsoft.com), several on-line services, and CD-ROM subscription services, such as Microsoft TechNet.

Summary

In summary, when troubleshooting, it is usually best to first verify that a path exists between the hosts in question using ping (by address). Then verify the ability to resolve hostnames using ping by name. Then, if NetBIOS is involved, verify that those names can be resolved, using *net view <servername>* or a similar command. Compile a list of what works and what doesn't work, then study the list to help isolate the failure. If link reliability is in question, try a large number of pings of various sizes at different times of the day, and plot the success rate. When all else fails, a protocol analyzer, such as Microsoft Network Monitor, may be helpful.

Appendix A: TCP/IP Configuration Parameters

Introduction

The TCP/IP protocol suite implementation for Windows NT 3.5*x* reads all of its configuration data from the registry. This information is written to the registry by the Network Control Panel Applet (NCPA) as part of the setup process. Some of this information is also supplied by the Dynamic Host Configuration Protocol (DHCP) client service if it is enabled. This reference 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 the NCPA and DHCP. Optimal default values for all other configurable aspects of the protocols have been encoded into the drivers. There may be some circumstances in customer installations where changes to certain default values are appropriate. To handle these cases, optional registry parameters can be created to modify the default behavior of some parts of the protocol drivers.

> **CAUTION: The Windows NT 3.5 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 one of two different subkeys of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:

- Tcpip\Parameters
- <Adapter Name>\Parameters\Tcpip, in which <Adapter Name> refers to the subkey for a network adapter that TCP/IP is bound to, such as Lance01.

Values under the latter key(s) are specific to each adapter. Parameters for which there may be both a DHCP and statically configured value may or may not exist depending on whether the system/adapter is DHCP configured and/or static override values have been specified. A reboot of the system is required for a change in any of these parameters to take effect.

> **IMPORTANT NOTE: The Windows NT 3.5 Resource Kit documentation was not updated properly from version 3.1, and lists a number of invalid TCP/IP registry parameters. The parameters listed in this document should be used in their place. The Windows NT 3.5 TCP/IP stack was a complete re-write, so many of the old parameters are no longer valid. The Windows NT 3.51 Resource Kit should include the necessary corrections.**

Standard Parameters Configurable using the Registry Editor

The following parameters are installed with default values by the NCPA during the installation of the TCP/IP components. They may be modified using the Registry Editor (regedt32.exe).

DatabasePath

Key: Tcpip\Parameters
Value Type: REG_EXPAND_SZ - Character string
***Valid Range*:** A valid Windows NT file path
Default: %SystemRoot%\system32\drivers\etc
Description: This parameter specifies the path to the standard internet database files (HOSTS, LMHOSTS, NETWORKS, PROTOCOLS, SERVICES). It is used by the Windows Sockets interface.

ForwardBroadcasts

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: Forwarding of broadcasts is not supported. This parameter is ignored.

UseZeroBroadcast

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: If this parameter is set to 1 (True), then IP will use zeros-broadcasts (0.0.0.0) instead of ones-broadcasts (255.255.255.255). Most systems use ones-broadcasts, but some systems derived from BSD implementations use zeros-broadcasts. Systems that use different broadcasts will not interoperate well on the same network.

Optional Parameters Configurable using the Registry Editor

These parameters normally do not exist in the registry. They may be created to modify the default behavior of the TCP/IP protocol driver.

ArpAlwaysSourceRoute *(new in NT 3.51)*

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0,1 (False or True)
Default: 0 (False)
Description: Setting this parameter to 1 will force TCP/IP to transmit ARP queries with source routing enabled on Token Ring networks. By default, the stack transmits ARP queries without source routing first and retries with source routing enabled if no reply was received.

ArpUseEtherSNAP

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean

Valid Range: 0,1 (False or True)
Default: 0 (False)
Description: Setting this parameter to 1 will force TCP/IP to transmit Ethernet packets using 802.3 SNAP encoding. By default, the stack transmits packets in DIX Ethernet format. It will always receive both formats.

DefaultTOS

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number
Valid Range: 0 - 255
Default: 0
Description: Specifies the default Type Of Service (TOS) value set in the header of outgoing IP packets. See RFC 791 for a definition of the values.

DefaultTTL

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number of seconds/hops
Valid Range: 1-255
Default: 32
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 an IP packet may live in the network without reaching its destination. It is effectively a limit on the number of routers an IP packet may pass through before being discarded.

EnableDeadGWDetect

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0,1 (False, True)
Default: 1 (True)
Description: Setting this parameter to 1 causes TCP to perform Dead Gateway Detection. With this feature enabled, TCP will ask IP to change to a backup gateway if it re-transmits a segment several times without receiving a response. Backup gateways may be defined in the Advanced section of the TCP/IP configuration dialog in the Network Control Panel.

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 "Black Hole" routers while doing Path MTU Discovery. A "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 will try to send segments without the Don't Fragment bit set if several re-transmissions of a segment go unacknowledged[22]. If the segment is acknowledged as a result, the MSS will be decreased and the Don't Fragment bit will be set in future packets on the connection. Enabling black hole detection increases the maximum number of re-transmissions performed for a given segment.

EnablePMTUDiscovery

[22] As of Windows NT 3.51 Service Pack 2, the algorithm for PMTUBH Detection has been modified. The new algorithm is to reduce the MTU to 576 bytes (TCP MSS = 536) if no acknowledgment is received after retransmitting a large segment several times. Details are available from the Microsoft Windows NT Knowledgebase.

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0,1 (False, True)
Default: 1 (True)
Description: Setting this parameter to 1 (True) causes TCP to attempt 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 machines on the local subnet.

ForwardBufferMemory

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number of bytes
Valid Range: <network MTU> - <some reasonable value smaller than 0xFFFFFFFF >
Default: 74240 (enough for fifty 1480-byte packets, rounded to a multiple of 256)
Description: This parameter determines how much memory IP allocates to store packet data in the router packet queue. When this buffer space is filled, the router begins discarding packets at random from its queue. Packet queue data buffers are 256 bytes in length, so the value of this parameter should be a multiple of 256. Multiple buffers are chained together for larger packets. The IP header for a packet is stored separately. This parameter is ignored and no buffers are allocated if the IP routing function is not enabled.

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 may only send IP multicast packets. At level 2, the system may send IP multicast packets and fully participate in IGMP to receive multicast packets.

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 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 will be 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 will acknowledge the keep alive transmission. Keep alive packets are not sent by default. This feature may be enabled on a connection by an application.

MTU

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD Number
Valid Range: 68 - <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 packet size in bytes that the transport will transmit over the underlying network. The size includes the transport header. Note that an IP datagram may span multiple packets. Values larger than the default for the underlying network will result in the transport using the network default MTU. Values smaller than 68 will result in the transport using an MTU of 68.

IMPORTANT NOTE: Windows NT 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 PMTU detection discussion in the TCP section of this document for more details.

NumForwardPackets

Key: Tcpip\Parameters
Value Type: REG_DWORD Number
Valid Range: 1 - <some reasonable value smaller than 0xFFFFFFFF>
Default: 50
Description: This parameter determines the number of IP packet headers allocated for the router packet queue. When all headers are in use, the router will begin to discard packets at random from the queue. This value should be at least as large as the ForwardBufferMemory value divided by the maximum IP data size of the networks connected to the router. It should be no larger than the ForwardBufferMemory value divided by 256, since 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 carried on the network and will be somewhere in between these two values. This parameter is ignored and no headers are allocated if routing is not enabled.

TcpMaxConnectRetransmissions

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number
Valid Range: 0 - 0xFFFFFFFF
Default: 3
Description: This parameter determines the number of times TCP will re-transmit a connect request (SYN) before aborting the attempt. The re-transmission timeout is doubled with each successive re-transmission in a given connect attempt. The initial timeout value is three seconds.

TcpMaxDataRetransmissions

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number
Valid Range: 0 - 0xFFFFFFFF
Default: 5

Description: This parameter controls the number of times TCP will re-transmit an individual data segment (not connection request segments) before aborting the connection. The re-transmission timeout is doubled with each successive re-transmission on a connection. It is reset when responses resume. The base timeout value is dynamically determined by the measured round-trip time on the connection.

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 may have open simultaneously.

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 specification for urgent data or the mode used by BSD-derived systems. The two mechanisms interpret the urgent pointer in the TCP header and the length of the urgent data differently. They are not interoperable. Windows NT defaults to BSD mode.

TcpWindowSize

Key: Tcpip\Parameters
Value Type: REG_DWORD - Number of bytes
Valid Range: 0 - 0xFFFF
Default: The smaller of 0xFFFF
OR
(The larger of four times the maximum TCP data size on the network
OR
8192 rounded up to an even multiple of the network TCP data size.)
The default is 8760 for Ethernet.
Description: This parameter determines the maximum TCP receive window size offered by the system. The receive window specifies the number of bytes a sender may transmit without receiving an acknowledgment. In general, larger receive windows will 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).

Parameters Configurable from the NCPA

The following parameters are created and modified automatically by the NCPA as a result of user-supplied information. There should be no need to configure them directly in the registry.

DefaultGateway

Key: <AdapterName>\Parameters\Tcpip
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 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.

Domain

Key: Tcpip\Parameters
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 system. It is used by the Windows Sockets interface.

EnableDhcp

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: If this parameter is set to 1 (True), then the DHCP client service will attempt to configure the first IP interface on this adapter using DHCP.

Hostname

Key: Tcpip\Parameters
Value Type: REG_SZ - Character string
Valid Range: Any valid DNS hostname
Default: The computername of the system
Description: This parameter specifies the DNS hostname of the system, that will be returned by the "hostname" command.

IPAddress

Key: <Adapter Name>\Parameters\Tcpip
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, then the primary interface on the adapter will be configured from DHCP. A system with more than one IP interface for an adapter is called "logically multihomed." There must be a valid subnet mask value in the SubnetMask parameter for each IP address specified in this parameter.

IPEnableRouter

Key: Tcpip\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: Setting this parameter to 1 (True) causes the system to route IP packets between the networks to which it is connected.

NameServer

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 (Blank)
Description: This parameter specifies the DNS name servers to be queried by Windows Sockets to resolve names.

SearchList

Key: Tcpip\Parameters
Value Type: REG_SZ - Space delimited list of DNS domain name suffixes
Valid Range: Any set of valid DNS domain name suffixes
Default: None
Description: This parameter specifies a list of domain name suffixes to append to a name to be resolved via the 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.

SubnetMask

Key: <Adapter Name>\Parameters\Tcpip
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 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, then the primary interface on the adapter will be configured via DHCP. There must be a valid subnet mask value in this parameter for each IP address specified in the IPAddress parameter.

Parameters Configurable via the Route.exe Command in Windows NT 3.51

In Windows NT 3.51, the route.exe command can store persistent IP routes as values under the *Tcpip\Parameters\PersistentRoutes* key. Each route is stored in the value name string as a comma-delimited list of the form:

destination,subnet mask,gateway

For example, the value representing a host route to destination 45.100.23.10 through gateway 131.110.0.1 would be named:

45.100.23.10,255.255.255.255,131.110.0.1

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: <Adapter Name>\Parameters\ Tcpip
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 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.

DhcpIPAddress

Key: <Adapter Name>\Parameters\Tcpip
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, then that value will override this parameter.

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, then it will override this parameter.

DhcpServer

Key: <Adapter Name>\Parameters\Tcpip
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: <Adapter Name>\Parameters\Tcpip
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.

IPInterfaceContext

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD
Valid Range: 0 - 0xFFFFFFFF
Default: None
Description: This parameter is written by the TCP/IP driver for use by the DHCP client service.

Lease

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Time in seconds
Valid Range: 1 - 0xFFFFFFFF
Default: None
Description: This parameter is used by the DHCP client service to store the time in seconds for which the lease on the IP address for this adapter is valid.

LeaseObtainedTime

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Absolute time in seconds since midnight of 1/1/70
Valid Range: 1 - 0xFFFFFFFF
Default: None

Description: This parameter is used by the DHCP client service to store the time at which the lease on the IP address for this adapter was obtained.

LeaseTerminatesTime

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Absolute time in seconds since midnight of 1/1/70
Valid Range: 1 - 0xFFFFFFFF
Default: None
Description: This parameter is used by the DHCP client service to store the time at which the lease on the IP address for this adapter will expire.

LLInterface

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_SZ - NT device name
Valid Range: A legal NT 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 NT device to which IP should bind. This parameter is in conjunction with the RAS component, for example.

T1

Key: <Adapter Name>\Parameters\Tcpip
Value Type: REG_DWORD - Absolute time in seconds since midnight of 1/1/70
Valid Range: 1 - 0xFFFFFFFF
Default: None
Description: This parameter is used by the DHCP client service to store the time at which the service will first try to renew the lease on the IP address for the adapter by contacting the server that granted the lease.

T2

Key: <AdapterName>\Parameters\Tcpip
Value Type: REG_DWORD - Absolute time in seconds since midnight of 1/1/70
Valid Range: 1 - 0xFFFFFFFF
Default: None
Description: This parameter is used by the DHCP client service to store the time at which the service will try 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 has been unable to renew the lease with the original server for some reason.

Appendix B:
NetBT (NetBIOS over TCP) Configuration Parameters

Introduction

All of the NetBT parameters are registry values located under one of two different subkeys of HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services:

- NetBT\Parameters
- NetBT\Adapters\<Adapter Name>, in which <Adapter Name> refers the subkey for a network adapter that NetBT is bound to, such as Lance01.

Values under the latter key(s) are specific to each adapter. If the system is configured via DHCP, then a change in parameters will take effect if the command *ipconfig /renew* is issued in a command shell. Otherwise, a reboot of the system is required for a change in any of these parameters to take effect.

Standard Parameters Configurable from the Registry Editor

The following parameters are installed with default values by the NCPA during the installation of the TCP/IP components. They may be modified using the Registry Editor (regedt32.exe).

BcastNameQueryCount

Key: Netbt\Parameters
Value Type: REG_DWORD - Count
Valid Range: 1 to 0xFFFF
Default: 3
Description: This value determines the number of times NetBT broadcasts a query for a given name without receiving a response.

BcastQueryTimeout

Key: Netbt\Parameters
Value Type: REG_DWORD - Time in milliseconds
Valid Range: 100 to 0xFFFFFFFF
Default: 0x2ee (750 decimal)
Description: This value determines the time interval between successive broadcast name queries for the same name.

CacheTimeout

Key: Netbt\Parameters
Value Type: REG_DWORD - Time in milliseconds
Valid Range: 60000 to 0xFFFFFFFF
Default: 0x927c0 (600000 milliseconds = 10 minutes)
Description: This value determines the time interval that names are cached in the remote name table.

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 will send name service related packets, such as name queries and name registrations, to WINS. The Microsoft WINS Server listens on port 0x89. NetBIOS name servers from other vendors may listen on different ports.

NameSrvQueryCount

Key: Netbt\Parameters
Value Type: REG_DWORD - Count
Valid Range: 0 - 0xFFFF
Default: 3
Description: This value determines the number of times NetBT sends a query to a WINS server for a given 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 given name.

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 keepalive transmissions on a session. Setting the value to 0xFFFFFFF disables keepalives.

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 used to store local and remote names. In general, a setting of Small is adequate. If the system is acting as a proxy nameserver, then the value is automatically set to Large to increase the size of the name cache hash table. Hash table buckets are sized as follows:

Large: 256 Medium: 128 Small: 16

Optional Parameters Configurable from the Registry Editor

These parameters normally do not exist in the registry. They may be created to modify the default behavior of the NetBT protocol driver.

BroadcastAddress

Key: Netbt\Parameters
Value Type: REG_DWORD - Four byte, little-endian encoded IP address
Valid Range: 0 - 0xFFFFFFFF
Default: The ones-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 ones-broadcast address appropriate for each net (i.e., 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 zeros-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. Note that this parameter is global and will be used on all subnets that NetBT is bound to.

EnableProxyRegCheck

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: If this parameter is set to 1 (True), then the proxy name server will send 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. The hazard of enabling this feature is that it 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.

InitialRefreshTimeout

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 timeout used by NetBT during name registration. NetBT tries to contact the WINS servers at 1/8th of this time interval when it is first registering names. When it receives a successful registration response, that response will contain 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 timeout value for LMHOSTS and DNS name queries. The timer has a granularity of the timeout value, so the actual timeout could be as much as twice the value.

MaxDgramBuffering

Key: Netbt\Parameters
Value Type: REG_DWORD - Count of bytes
Valid Range: 0 - 0xFFFFFFFF
Default: *0x*20000 (128 Kb)
Description: This parameter specifies the maximum amount of memory that NetBT will dynamically allocate for all outstanding datagram sends. Once this limit is reached, further sends will fail due to insufficient resources.

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 will use 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 via LMHOSTS and/or DNS, if enabled, will follow these methods. If this key is present it will override 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.

RandomAdapter

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: This parameter applies to a multihomed host only. If it is set to 1 (True), then NetBT will randomly choose 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 that the query arrived on. This feature would be used by a server with two interfaces on the same network for load balancing.

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 in name refresh packets. The specification for the NetBT protocol is somewhat ambiguous in this area. Although the default of 8 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 to interoperate.

SingleResponse

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: This parameter applies to a multihomed host only. If this parameter is set to 1 (True), then NBT will only supply an IP address from one of its bound interfaces in name query responses. By default, the addresses of all bound interfaces are included.

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 NetBT will wait before again trying to use WINS 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 timeout out each WINS name registration or query individually.

Parameters Configurable from the NCPA

The following parameters can be set via the NCPA. There should be no need to configure them directly.

EnableDns

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: If this value is set to 1 (True), then NetBT will query the DNS for names that cannot be resolved by WINS, broadcast, or the LMHOSTS file.

EnableLmhosts

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 1 (True)
Description: If this value is set to 1 (True), then NetBT will search 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 will be taken. This value is written by the Advanced TCP/IP Configuration dialog of the NCPA.

EnableProxy

Key: Netbt\Parameters
Value Type: REG_DWORD - Boolean
Valid Range: 0 or 1 (False or True)
Default: 0 (False)
Description: If this value is set to 1 (True), then the system will act 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 nameserver allows a network of b-node implementations to connect to servers on other subnets that are registered with WINS.

NameServer

Key: Netbt\Adapters\<Adapter Name>
Value Type: REG_SZ - Dotted decimal IP address (i.e. 10.101.1.200)
Valid Range: Any valid IP address
Default: blank (no address)

Description: This parameter specifies the IP address of the primary WINS server. If this parameter contains a valid value, it overrides the DHCP parameter of the same name.

NameServerBackup

Key: Netbt\Adapters\<Adapter Name>
Value Type: REG_SZ - Dotted decimal IP address (i.e. 10.101.1.200)
Valid Range: Any valid IP address.
Default: blank (no address)
Description: This parameter specifies the IP address of the secondary WINS server. If this parameter contains a valid value, it overrides the DHCP parameter of the same name.

ScopeId

Key: Netbt\Parameters
Value Type: REG_SZ - Character string
Valid Range: Any valid DNS domain name consisting of two dot-separated parts, or a "*".
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 will override the DHCP parameter of the same name. A blank value (empty string) will be ignored. Setting this parameter to the value "*" indicates a null scope and will override the DHCP parameter.

Non-Configurable Parameters

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

DhcpNameServer

Key: Netbt\Adapters\<Adapter Name>
Value Type: REG_SZ - Dotted decimal IP address (i.e. 10.101.1.200)
Valid Range: Any valid IP address
Default: None
Description: This parameter specifies the IP address of the primary WINS server. It is written by the DHCP client service, if enabled. A valid NameServer value will override this parameter.

DhcpNameServerBackup

Key: Netbt\Adapters\<Adapter Name>
Value Type: REG_SZ - Dotted decimal IP address (i.e. 10.101.1.200)
Valid Range: Any valid IP address
Default: None
Description: This parameter specifies the IP address of the secondary WINS server. It is written by the DHCP client service, if enabled. A valid NameServerBackup value will override this parameter.

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 will override 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 (AFD.SYS) Registry Parameters

Introduction

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

- The first value is the default for "small" computers (<12.5 MB).
- The second value is the default for "medium" computers (12.5 to 20MB).
- The third value is the default for large computers (> 20 MB).

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

Performance-Related Values

The following values may be set under:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Afd\Parameters:

LargeBufferSize

Value Type: REG_DWORD
Default: 4096
Description: The size in bytes of large buffers used by AFD. Smaller values use less memory, larger values can improve performance.

InitialLargeBufferCount

Value Type: REG_DWORD
Default: 0/2/10
Description: The count of large buffers allocated by AFD at system startup. More buffers can be allocated to improve performance at the cost of physical memory.

MediumBufferSize

Value Type: REG_DWORD
Default: 1504
Description: The size in bytes of medium buffers used by AFD.

InitialMediumBufferCount

Value Type: REG_DWORD
Default: 2/10/30
Description: The initial count of medium buffers.

SmallBufferSize

Value Type: REG_DWORD
Default: 64
Description: The size in bytes of small buffers used by AFD.

InitialSmallBufferCount

Value Type: REG_DWORD
Default: 5/20/50
Description: The initial count of small buffers.

FastSendDatagramThreshold

Value Type: REG_DWORD
Default: 1024
Description: Datagrams smaller than this get buffered on send, and larger ones are pended (locked down in memory, and held until the datagram is actually sent). The default value was found by testing to be the best overall value for performance. Changing this value is not recommended.

StandardAddressLength

Value Type: REG_DWORD
Default: 24
Description: The length of TDI addresses typically used for the computer. When using an alternate transport protocol such as TP4, which uses very long addresses, increasing this value will result in a slight performance improvement.

DefaultReceiveWindow

Value Type: REG_DWORD
Default: 8192
Description: The number of receive bytes AFD will buffer on a connection before imposing flow control. For some applications, a larger value here will give slightly better performance at the expense of increased resource utilization. Note that applications can modify this value on a per-socket basis with the SO_RCVBUF socket option.

DefaultSendWindow

Value Type: REG_DWORD
Default: 8192
Description: Similar to DefaultReceiveWindow, but for the send side of connections.

BufferMultiplier

Value Type: REG_DWORD
Default: 512
Description: DefaultReceiveWindow and DefaultSendWindow get divided by this value to determine how many messages can be sent/received before flow control is imposed.

PriorityBoost

Value Type: REG_DWORD
Default: 2
Description: The priority boost AFD gives to a thread when it completes I/O for that thread. If a multithreaded application experiences starvation of some threads, reducing this value may remedy the problem.

IrpStackSize

Value Type: REG_DWORD
Default: 4
Description: The count of IRP stack locations used by default for AFD. Changing this value is not recommended.

TransmitIoLength *(new in 3.51)*

Value Type: REG_DWORD
Default: PAGE_SIZE, PAGE_SIZE*2, 65536
Description: The default size for I/O (reads and sends) performed by TransmitFile(). Note that, for Windows NT workstation, the default I/O size is exactly one page.

Service Resolution and Registration Parameters

The following keys are used by the RNR (service resolution and registration) APIs in Winsock. These are all just "pointers" to other items in the registry. Changing these values is not recommended.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ServiceProvider\Order

ExcludedProviders:

Value Type: REG_MULTI_SZ
Default: Empty Set
Description: Contains decimal values corresponding to name space providers that should be excluded. Some name space provider decimal values include:

NS_SAP	(1)
NS_NDS	(2)
NS_TCPIP_LOCAL	(10)
NS_TCPIP_HOSTS	(11)
NS_DNS	(12)
NS_NETBT	(13)
NS_WINS	(14)
NS_NBP	(20)
NS_MS	(30)
NS_STDA	(31)
NS_CAIRO	(32)
NS_X500	(40)
NS_NIS	(41)

For example, setting ExcludedProviders to "1" "12" means that GetAddressByName() will not attempt to use SAP or DNS when doing typical name resolution operations.

ProviderOrder

Value Type: REG_MULTI_SZ
Default: Varies with installed protocols
Description: Contains strings corresponding to keys under CurrentControlSet\Services. These keys must have a ServiceProvider subkey that provides information about the name space provider, especially Class and ProviderPath values.

TCP/IP Name Resolution Parameters

These parameters are used by the gethostbyname() and GetAddressByName() APIs. They are found under:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\ServiceProvider:

Class

Value Type: REG_DWORD
Default: 8
Description: This should not be changed–it indicates that TCPIP is a name service provider.

DnsPriority

Value Type: REG_DWORD
Default: 0x7D0
Description: These priority values are used to determine the order of name resolution. Low priority mechanisms are used first, so **the default order is: local, hosts, dns, NetBT**. To alter name resolution order, re-adjust the priority values as needed. Note that values under 1000 decimal are considered "fast" name resolution providers, so *putting network-based resolution mechanisms like dns and NetBT at values under 1000 may have undesirable effects.*

HostsPriority

Default: 0x1F4
Description: See DnsPriority.

LocalPriority

Default: 0x1F3
Description: See DnsPriority.

NetbtPriority

Default: 0x7D1
Description: See DnsPriority.

Name

Value Type: REG_SZ
Default: "TCP/IP"
Description: Transport name. There is no need to change this.

ProviderPath

Value Type: REG_SZ
Default: "%SystemRoot%\System32\wsock32.dll"
Description: Points to the DLL that does TCP/IP name resolution. There is no need to change this.

Appendix D:
Microsoft FTP Server Configuration Parameters

Introduction

All of the FTP Server parameters are registry values located under the following registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FtpSvc\Parameters

Configurable Parameters

AllowAnonymous

Value Type: REG_DWORD
Default: 1 (allow anonymous logons)
Description: If the value is non-zero, then anonymous logons are allowed. Otherwise, (if the value IS zero), anonymous logons are not allowed.

AllowGuestAccess *(new in 3.51)*

Value Type: REG_DWORD
Default: 1 (allow guest-access)
Description: If this value is non-zero, then guest-access logons are allowed. Otherwise, (if the value is zero), guest-access logons are not allowed. (Note that guest-access has nothing to do with the "Guest" account; guest-access is granted if the local computer's user rights policy states that "Everyone" may access the host from the network. Any user that tries to logon with an unknown account will be granted guest-access.)

AnnotateDirectories

Value Type: REG_DWORD
Default: 0 (don't annotate directories)
Description: If this value is non-zero, then every time a user changes directories (sends the server a CWD command) an attempt is made to open a file called "~FTPSVC~.CKM" in the new directory. If this file is found, its contents are sent to the user as part of the successful reply to the CWD command. This may be used to attach "annotations" to specific directories. This value is used as a default for new users. Users can toggle their own personal "annotate directories" flag with the site-specific CKM command (SITE CKM).

AnonymousOnly

Value Type: REG_DWORD
Default: 0 (non-anonymous logons allowed).
Description: If this value is non-zero, then only anonymous logons are allowed. Otherwise, (if the value is zero), then non-anonymous logons are allowed as well.

AnonymousUserName

Value Type: REG_SZ
Default: "Guest"
Description: Anonymous logon alias. When a user attempts an anonymous logon, the username specified ("anonymous") is mapped to this registry value for the purposes of authentication and impersonation. The password for this

account is stored in an LSA secret object named "FTPD_ANONYMOUS_DATA".

ConnectionTimeout

Value Type: REG_DWORD
Default: 600 (10 minutes)
Description: The time (in seconds) allotted to allow clients to remain idle before forcibly disconnecting them. This prevents idle clients from consuming server resources indefinitely. This value may be set to zero if time-outs are not to be enforced. If set to zero, idle clients can remain connected indefinitely.

DebugFlags

Value Type: REG_DWORD
Default: 0 (no debug output)
Description: This value is used only by the debugging (checked) builds of the FTP Server. It controls the output of various debugging information. This value is unused by retail builds.

DefaultLogonDomain *(new in 3.51)*

Value Type: REG_SZ
Default: NULL (use the local computer's primary logon domain)
Description: The domain name to use when validating user logon requests if the user did not specify a domain. If this value does not exist in the registry, then the FTP Server will use the local computer's primary logon domain instead.

ExitMessage

Value Type: REG_SZ
Default: "Goodbye."
Description: This is the signoff message sent to a client upon receipt of a QUIT command.

GreetingMessage

Value Type: REG_MULTI_SZ
Default: NULL (no special greeting)
Description: This message (if it exists in the registry) is sent to new clients after their account has been validated. In accordance with "de facto" Internet behavior, if a client logs on as anonymous and specifies an identity starting with “-” (minus), then this greeting message is NOT sent.

HomeDirectory

Value Type: REG_EXPAND_SZ
Default: "C:\"
Description: This is the initial "home" directory for new clients. After a new client is validated, an attempt is made to CHDIR to this directory. If this directory is inaccessible, the client is refused FTP services. If the CHDIR is successful, then an attempt is made to CHDIR to a directory with the same name as the client's username. If this fails, an attempt is made to CHDIR to a directory called "Default". If this fails, the current directory is left at "home." If a user finds that the home directory is inaccessible, then an event is written to the event log indicating such.

ListenBacklog *(new in 3.51)*

Value Type: REG_DWORD
Default: 5
Maximum: 100

Description: This is the "backlog" parameter passed into the *listen()* API. This sets the maximum number of unaccepted connections that can be queued against the socket that listens on the main FTP port.

LogAnonymous

Value Type: REG_DWORD
Default: 0 (don't log successful anonymous logons)
Description: If this value is non-zero, then all successful anonymous logons are logged in the system event log. Otherwise, (if the value is zero), successful anonymous logons are not logged.

LogFileAccess

Value Type: REG_DWORD
Default: 0 (don't log file accesses)
Description: This value controls the logging of file accesses. This value can be one of the following:

- 0 = Don't log file accesses
- 1 = Log file accesses to FTPSVC.LOG
- 2 = Log file accesses to *FTyymmdd.LOG*, where *yy* is the year, *mm* is the month, and *dd* is the day. A new log file will be opened every day as necessary.

LogFileDirectory

Value Type: REG_SZ
Default: %SystemRoot%\System32
Description: This value specifies the target directory for log files. This value is only used if LogFileAccess is not 0.

LogNonAnonymous

Value Type: REG_DWORD
Default: 0 (don't log successful nonanonymous logons)
Description: If this value is non-zero, then all successful non-anonymous logons are logged in the system event log. Otherwise, (if the value is zero), successful non-anonymous logons are not logged.

LowercaseFiles

Value Type: REG_DWORD
Default: 0 (don't map filenames to lowercase)
Description: If this value is non-zero, then all file names returned by LIST and NLST commands for non-case-preserving file systems will be mapped to lowercase. If this value is zero, then all file names will be unaltered.

MaxClientsMessage

Value Type: REG_SZ
Default: "Maximum clients reached, service unavailable."
Description: This message (if it exists) is sent to a client if the maximum number of clients has been reached/exceeded. This indicates that the server is currently servicing the maximum number of simultaneous clients and is refusing additional clients.

MaxConnections

Value Type: REG_DWORD
Default: 20
Description: This is the maximum number of simultaneous clients the server will service. This value may be set to zero if there is to be no limit on simultaneous clients.

MsdosDirOutput

Value Type: REG_DWORD
Default: 1 (directory listings like MS-DOS®)
Description: If this value is non-zero, then the output of the LIST command (usually sent as a result of a DIR from the client) will look like the output of the MS-DOS DIR command. Otherwise, (if the value IS zero), then the output of the LIST command will look like the output of the UNIX LS command. This value also controls "slash flipping" in the path sent by the PWD command. If this value is non-zero, the path will contain backward "\" slashes. If this value is zero, the path will contain forward "/" slashes.

ReadAccessMask

Value Type: REG_DWORD
Default: 0 (all read access denied)
Description: This value is a bitmask and controls the "readability" of the various disk volumes in the system. Drive A: corresponds to bit zero, drive B: is bit 1, drive C: is bit 2, etc. A user may only read from a specific volume if the corresponding bit is set.

WriteAccessMask

Value Type: REG_DWORD
Default: 0 (all write access denied)
Description: This value is a bitmask and controls the "writability" of the various disk volumes in the system. Drive A: corresponds to bit zero, drive B: is bit 1, drive C: is bit 2, etc. A user may only write to a specific volume if the corresponding bit is set.

Optional Key: AccessCheck

There is an additional (optional) key that may exist under the Parameters key. After a user's account/password has been validated and the server is impersonating that user, an attempt is made to open a key named "AccessCheck." If this key exists, and the user cannot open it, then the user is denied access to the FTP Server. If this key exists, and the user can only open it for read access, then the user is given read-only access to the FTP Server. This way, an administrator can create this "AccessCheck" key and attach specific ACLs to the key. These ACLs will then control access to the FTP Server.