

Chapter 3: Windows 2000 TCP/IP Implementation Details

Windows 2000 uses TCP/IP as the main communication protocol for building enterprise networks, intranets, and connectivity to the Internet. Because the Windows TCP/IP protocol stack is a multithreaded stack, protocol performance can be improved by the thread scheduler that can parallelize TCP/IP operations when run on a multiprocessor computer. This chapter discusses some of the TCP/IP protocol implementation details of TCP/IP in the Windows 2000 operating system.

Components of Windows 2000 TCP/IP

The architecture of the Windows 2000 TCP/IP stack and its relationship to the Open Systems Interconnection (OSI) model is described in [Figure 3.1](#). The Windows 2000 TCP/IP stack implements a number of major protocols, such as TCP, UDP, IP, ICMP, ARP, IGMP, IP forwarding, and IP filtering. The *Transport Driver Interface Specification (TDI)* provides an interface to layer 4 of the OSI model and is used by applications. The *Network Device Interface (NDIS)* is the interface between layer 2 and layer 3 of the OSI model and is used by network adapter drivers. A number of higher-level interfaces are available to user-mode applications, such as Windows Sockets (WINSOCK), Remote Procedure Call (RPC), NetBIOS, Wnet, and WinInet.

Figure 3.1

Windows 2000 TCP/IP components.

Understanding NDIS Features

The Network Driver Interface Specification (NDIS) has gone through an evolutionary process from versions 3.1 to 5.0 with new features and capabilities added as Microsoft operating systems have become more complex.

NDIS 3.1 supported a set of basic services that enabled a protocol module (TCP/IP and IPX, for example) to send raw packets over a network device and allowed that same module to be notified of incoming packets received by a network device. NDIS 3.1 was used as a simple switch/multiplexor between the protocol modules and the network adapters.

NDIS 4 was introduced in Windows NT 4 and has the capability of NDIS 3.1, but added a number of new features, including out-of-band data support, media sense, local packet filtering support, and new NDIS system functions for miniport binary compatibility across the Windows family of products. The media sense feature enables network disconnection events to be reported to protocol stacks, and the local filtering prevents the Network Monitor protocol analyzer from monopolizing the CPU.

NDIS 5 was introduced with Windows 2000 and is a superset of NDIS 4. Additionally, NDIS 5 includes support for power management, plug-and-play, Windows Management Instrumentation (WMI), improved miniport performance, protocol task offload mechanisms, Asynchronous Transfer Mode (ATM), Quality of Service (QoS) parameters, Asymmetric Digital Subscriber Line (ADSL), and Windows Driver Model—Connection Streaming Architecture (WDM-CSA).

NDIS 5 power management is particularly useful for laptop computers. Power management can be used to conserve battery power by powering down network adapters that are not in use. The power-

management system requests a power level change when the cable is disconnected or after a timeout period. NDIS power management requires that the network adapter hardware support this feature. There must be no network activity before NDIS invokes its power-management functions. No network activity means that all network components that are affected by powering down the network adapter must be in an idle state. If there are active sessions or open files over the network, the power-down request can be refused by any of the affected components.

If the computer is in sleep mode to conserve power, it can be awakened by network events, such as a change in the network link state, receipt of a network wakeup frame, or receipt of a special packet.

When the NDIS driver initializes, it queries the network adapter and determines which of the network adapter power-management functions are supported. NDIS then determines the lowest required power state for each wakeup method, thereby enabling the protocols to set the wakeup policy.

NDIS 5 provides support for task offloading whereby network adapters can be used to perform some computation chores. Task offloading is automatically enabled and does not require any additional configuration. Windows 2000 TCP uses this task off-loading feature by allowing the network adapter to perform the TCP checksum calculations. Offloading the checksum calculations to hardware can result in performance improvements in high-throughput environments. (Please note that TCP/IP refers to the entire TCP/IP stack. TCP refers to only the TCP module in the stack.)

Data Link Layer Implementation

The Hardware Link layer (such as Ethernet and Token Ring) serves as a filter. This layer rejects all incoming frames except those containing the specific address of the adapter, the "all 1s" broadcast address (FF-FF-FF-FF-FF-FF), and multicast addresses (class D addresses).

Modern network adapters discard any frames that do not meet the filter criteria. They do so without using up CPU processing power. After a frame passes through the hardware filter, it is forwarded by NDIS to the appropriate protocol module via a hardware interrupt. Hardware interrupts dealing with packet arrival are processed by entry points in the NDIS driver. The NDIS driver code is executed by the computer's CPU. After the packet passes through the hardware filter, CPU cycles are expended. In older network adapters, even the hardware filter functions were performed by the computer's CPU. This resulted in CPU cycles being stolen by excessive network traffic.

After the network adapter driver passes the frame from the network adapter into system memory, the frame is forwarded to the appropriate transport protocol module (implemented as a transport protocol driver). After that step, the frames are passed up to all bound transport drivers in the order in which they are bound.

Each network type has a maximum transmission unit (MTU) size that cannot be exceeded. *MTU* is the amount of data that can be sent in a Data Link layer frame; it is the data "payload." For example, Ethernet has an MTU of 1500 octets; Token Ring has an MTU of 1440 octets (4Mbps), or 17940 octets (16Mbps); and Fiber Distributed Data Interface (FDDI) has an MTU of 4352 octets.

The Data Link layer is responsible for discovering this MTU and reporting it to the protocols that are higher in the network OSI model. NDIS drivers might be queried by protocol modules for the local MTU. The MTU size for an interface is used by upper-layer protocols, such as TCP, that optimize packet sizes for each media automatically.

If an NDIS driver for a network such as ATM uses LAN emulation mode, the MTU might be higher than normal for that network type. Windows 2000 accepts the MTU size reported by the adapter, even when the size is greater than or less than the normal MTU size for a given media type. The procedure for setting the MTU size is dependent on the LANE driver software.

Understanding Windows 2000 TCP/IP Implementation Issues

The Windows 2000 core protocol stack components are those shown between the NDIS and Transport Driver Interface (TDI) interfaces in [Figure 3.1](#). The Windows 2000 TCP/IP protocol elements are implemented in the TCPIP.SYS driver. The following sections discuss different issues of interest regarding the Windows 2000 TCP/IP implementation.

Windows 2000 Internet Protocol (IP) Implementation

Windows 2000 implements the Internet Protocol described in STD 5. The Internet Protocol is discussed in detail in Chapter 2 "TCP/IP Protocols Infrastructure for Windows Networks."

The Internet Protocol in Windows 2000 TCP/IP is part of the TCPIP.SYS driver. The IP module sends incoming IP datagrams to upper-layer protocols (TCP and UDP, for example) above from the NDIS driver below and sends outgoing datagrams to the NDIS driver from the upper-layer protocol. The Windows 2000 IP implementation closely follows the standards (STD 5) implementation.

The function of the Internet Protocol layer is to define a network address (in this case, the IP address) for each network interface and then perform IP forwarding and routing. The overall objective of the IP layer is to provide a logical abstraction of the underlying physical networks that is independent of the physical-layer characteristics of the network (see [Figure 3.2](#)) such as MTU size and network hardware mechanisms. In [Figure 3.2](#), the upper-protocol layers (such as TCP) above IP do not have to be concerned about media-dependent packet sizes and other operating mechanisms.

Figure 3.2

The IP provides a logical abstraction of physical networks.

Windows 2000 IP implementation also supports IP multicasting to provide multicast services to clients that might not be located on the same network segment. For example, Windows applications can join a multicast group to participate in a wide-area conference. Windows 2000 implements the Internet Group Management Protocol (IGMP) for managing IP multicasting. Windows 2000 is level-2 (send and receive) -compliant with RFC 1112, "Host Extensions for IP Multicasting."

The IP layer in each node performs IP forwarding on the network interfaces that it knows about for packets generated within the node. When IP forwarding is done for packets originating on other parts of the network, it is called *routing*. In either case, the IP examines the destination IP address on each datagram, compares it to a locally maintained host route table, and decides whether it needs to forward the datagram to the local host or to one of the network interfaces or discard the datagram.

Multicasting Versus Broadcasting

Broadcasts are received by all computers on a physical network and might cause the computers on the network to expend CPU cycles in processing the broadcasted packets. Because multicast packets are received only by those computers that are part of a multicast group, multicasting is more efficient than broadcasting. The multicast group is identified by a class D IP address.

Each IP node maintains a host route table that contains host entries, network entries, and default route entries. The destination address of the IP datagram is matched first against a host entry; next, against a network entry; and then against the default entry (see [Figure 3.3](#)). Host-specific entries are specified for important hosts and specify how datagrams to the specified entry are routed. Network entries represent groups of hosts on a network and aggregate forwarding for hosts on a network into a single network entry in the host route table. If no match occurs for a host-specific entry or network entry, the default entry is tried.

Figure 3.3

Host routing logic.

Loopback Address

A special IP address, 127.0.0.1, is used as the loopback address. Packets sent to the loopback address are sent back to the process that originated the packet without leaving the network interface.

You can use the loopback address for diagnostic and testing purposes. For example, if a Web client and Web server are running on the same machine, you can use the Web client to connect to the Web server using the URL `http://127.0.0.1`.

Although 127.0.0.1 is typically used as a loopback address, any address in the form 127.x.x.x will suffice.

You can use the `route print` command to view the route table from the command prompt:

```
>route print
=====
Interface List
0x1 ..... MS TCP Loopback interface
0x2 ...00 60 97 8b 74 5e ..... EL3C589 Ethernet Adapter
=====
Active Routes:
Network Destination        Netmask          Gateway           Interface         Metric
0.0.0.0                    0.0.0.0          63.224.13.118    63.224.13.116     1
10.0.0.0                   255.0.0.0        10.0.0.116       63.224.13.116     1
10.0.0.116                 255.255.255.255  127.0.0.1        127.0.0.1         1
10.255.255.255             255.255.255.255  10.0.0.116       63.224.13.116     1
63.224.13.112              255.255.255.248  63.224.13.116    63.224.13.116     1
63.224.13.116              255.255.255.255  127.0.0.1        127.0.0.1         1
63.255.255.255             255.255.255.255  63.224.13.116    63.224.13.116     1
```

```

        127.0.0.0          255.0.0.0          127.0.0.1          127.0.0.1          1
        224.0.0.0          224.0.0.0          63.224.13.116      63.224.13.116      1
    255.255.255.255  255.255.255.255  63.224.13.116      63.224.13.116      1
Default Gateway:      63.224.13.118
=====
Persistent Routes:
None

```

This route table is for a computer with the class A IP addresses 10.0.0.116 and 63.224.13.116 assigned to the same Network Interface Card (NIC). The route table contains the following entries:

1. Network Destination 0.0.0.0 is the default route. Notice that the subnet mask is 0.0.0.0.
2. Network Destination 10.0.0.0 is for the subnet 10.0.0.0, on which this computer resides. The route to 10.0.0.0 is going through the 63.224.13.116 interface because the two addresses are for the same NIC.
3. Network Destination 10.0.0.116 is a host-specific route for the local host. Because it specifies the loopback address (127.0.0.1), a datagram bound for the local host should be looped back internally.
4. Network Destination 10.255.255.255 is for the directed network broadcast address.
5. Network Destination 63.224.13.112 is for the subnet 63.224.13.112 with subnet mask 255.255.255.248, on which this computer resides.
6. Network Destination 63.224.13.116 is a host-specific route for the local host. Because this destination specifies the loopback address (127.0.0.1), a datagram bound for the local host should be looped back internally.
7. Network Destination 63.255.255.255 is for the directed network broadcast address for network 63.0.0.0.
8. Network Destination 127.0.0.0 is for the loopback address 127.0.0.0.
9. Network Destination 224.0.0.0 is for IP multicasting.
10. Network Destination 255.255.255.255 is for the limited broadcast (all 1s) address that does not cross router boundaries and is confined to the local network.
11. The Default Gateway is the active default gateway. This information is useful to know when multiple default gateways are configured. A computer can have only one active default gateway at any time.

255.255.255.255 Host Routes

You can tell that a route is a host route when the corresponding subnet mask is 255.255.255.255. All the bits in the destination are therefore significant and are used for comparison purposes.

Routing concepts involving subnetting, supernetting, Variable-Length Subnet Masks (VLSMs), subnet number assignment optimization, and Classless Internet Domain Routing (CIDR) concepts are discussed in detail in Chapter 9 "Routing and Remote Access with Microsoft TCP/IP."

If a routing protocol such as Routing Information Protocol (RIP) is started on a Windows 2000 computer or if the Windows 2000 TCP/IP host acts as a router, the Windows 2000 computer will learn about other routes through the configured routing protocols. Windows 2000 computers can also learn about routers using the ICMP redirect message or the ICMP router discovery messages.

Using Subnet Masks to Match Route Table Entries

For each of the route entries, a subnet mask column is used to determine the number of most significant bits in the destination IP address that should be used for matching. The host entries use a subnet mask of 255.255.255.255. The subnets use either the default subnet mask (255.0.0.0 for class A network, 255.255.0.0 for class B network, 255.255.255.0 for class C network) if no subnetting is being used or some other subnet mask to indicate that subnetting is being used. The default route uses a subnet mask of 0.0.0.0 so that a match will always take place. Using 0.0.0.0 is like saying "Don't use any bits for matching—it will always match." A match is performed by computing the bitwise AND of the destination IP address with the subnet mask and comparing it with the routing entry that is bitwise ANDed with the subnet mask. The bitwise AND operation is performed by converting the IP address and subnet mask into bit representations and logically performing the AND operation. The logical AND is described as follows:

Operand1	Operand2	Result
0	0	0
0	1	0
1	0	0
1	1	1

The metric column is a measure of the cost of the route path specified for that routing entry. The higher the metric, the greater the route path cost. Routing entries with lower metric values are preferred over those with larger values.

ICMP redirect messages can specify redirection for hosts or networks and indicate that host routes on Windows 2000 computers need to be changed to more efficient routes. When a Windows 2000 computer receives an ICMP redirect, it performs a check to see whether the ICMP message came from the router on the directly connected network. If it did, ten minutes are added to the lifetime for that route. If the ICMP message is not from a router on the directly connected network, the ICMP message is ignored. ICMP redirect messages sent to a host let the host know that a more efficient route to the destination exists. The host should adjust its routing table to contain the more efficient

route.

The ICMP router discovery (specified by RFC 1256, "ICMP Router Discovery Messages") is another way that a Windows 2000 computer can learn about and configure default gateways. Windows 2000 computers can discover routers on their subnet using ICMP router discovery. When a host supporting ICMP router discovery initializes, it joins the all-systems IP multicast group (a class D address of 224.0.0.1) that is used by routers to send messages to that group. Windows 2000 initiates router discovery by sending as many as three solicitations to this multicast group at 600-millisecond intervals. The ICMP router discovery is controlled by the `PerformRouterDiscovery` and `SolicitationAddressBCast` Registry parameters under the `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\GUID` key. [Figure 3.4](#) shows the setting of the `PerformRouterDiscovery` parameter as well as others. If a parameter is not listed, its default value is assumed. Normally router solicitations are multicast rather than broadcast. By setting `SolicitationAddressBCast` to 1, the router solicitations are sent as broadcasts rather than as multicasts.

Figure 3.4

TCP/IP parameters for an interface with `PerformRouterDiscovery` highlighted.

You can also manually add routes using the `route` command with the `-p` option, which makes the routes persistent. For example, to add a static route for network 192.12.33.0 through router 63.224.13.118 with a metric of 4 on interface 2, use the following command:

```
route -p add 192.12.33.0 mask 255.255.255.0 63.224.13.118 metric 4 if 2
```

Manually added routes are called *static* routes. *Persistent* routes are stored in the Registry under the Registry key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\PersistentRoutes`.

Windows 2000 TCP/IP uses dead gateway detection to skip routers that are down. If a default router does not seem to be working, Windows 2000 triggers a dead gateway detection mechanism, which causes a switch to the next lowest metric default route in the default router list. Dynamic Host Configuration Protocol (DHCP) servers use a base metric with the list of default gateways and do not override statically configured default gateways. As an example, if a DHCP server provides a base of 50 and a list of default gateways, the gateways will be assigned metrics of 50, 51, and so on.

Windows 2000 TCP/IP uses metrics for default gateways with a 1 default value. A lower value for the default metric makes it the preferred default gateway. You can use the Advanced TCP/IP Settings Configuration screen to set the metric for default routes (see [Figure 3.5](#)).

Figure 3.5

Advanced TCP/IP settings.

Autonomous System (AS) routers use an Interior Gateway Routing Protocol, such as RIP version 1 and 2 or Open Shortest Path First (OSPF), to exchange routing tables with each other. Windows 2000 Server and Advanced Server include support for RIP and OSPF. Windows 2000 Professional (in addition to Windows 2000 Server and Advanced Server) includes support for passive RIP, or silent RIP. *Passive* RIP listens to RIP messages to learn about routes but does not actively participate in route path computation and the generation of RIP messages.

By default, Windows 2000–based systems do not behave as routers and do not forward IP datagrams between interfaces unless they are configured to do so. The Routing and Remote Access Service (RRAS), included in Windows 2000 Server, can be enabled and configured to provide full multiprotocol routing services. Chapter 9, "Routing and Remote Access with Microsoft TCP/IP" discusses this subject in more detail.

Proxy-ARP Support in Windows 2000 TCP/IP

Windows 2000 includes support for proxy-ARP environments. For example, if you have multiple logical subnets on the same physical network, the following command can be used to tell IP to treat all subnets as local:

```
route add 0.0.0.0 MASK 0.0.0.0 localIPAddress
```

Windows 2000 TCP/IP then uses ARP directly for the destination. If the previous command is executed, packets destined for nonlocal subnets are assumed to be on the directly connected network instead of being sent to a router. Proxy-ARP can be useful where several logical networks are used on one physical network with no router to the outside world or in a proxy-ARP environment (see [Figure 3.6](#)).

Protocols

RIP is an example of a distance-vector routing protocol, and OSPF is an example of a Link State protocol. Link State routing protocols are generally considered superior to distance-vector routing protocols. Some reasons are that Link State protocols converge faster in their route path computation and scale better for larger networks when compared with distance-vector protocols.

Figure 3.6

Proxy-ARP environment.

In a proxy-ARP environment, the router on the network must be configured to act as a proxy-ARP server. In [Figure 3.6](#) the Windows 2000 computer at the IP address 132.12.6.1 had been enabled for proxy-ARP operation. A subnet mask of 255.255.0.0 is assumed for the nodes so that the nodes think that they are on the same physical network when, in reality, they are divided into two separate networks on the two router ports. The router uses a 255.255.255.0 subnet mask so that it knows that networks 132.12.6.0 and 132.12.7.0 are separate subnets.

When the Windows 2000 computer issues an ARP request to discover the hardware address of the IP node at 132.12.7.3, the proxy-ARP at the router replies with the hardware address of the router's port. The Windows 2000 station then sends subsequent IP datagrams to the router port. The proxy-ARP forwards these datagrams to destination 132.12.7.3. To forward the datagrams to the destination 132.12.7.3, the router might issue an ARP request to discover the destination's hardware address if this information is not already in the router's ARP cache.

ARP in Legacy Networks

Proxy-ARP is useful in situations where the hosts cannot be configured with subnet masks (because they use old TCP/IP software) but the routers are configured to understand subnets.

If a Windows 2000 computer is configured for proxy-ARP and the router is not configured as proxy-ARP, the Windows 2000 computer cannot get to anything beyond its local subnet.

Duplicate IP Address Detection in Windows 2000 TCP/IP

Duplicate IP addresses can cause havoc on your network. Consequently the detection of duplicate IP addresses is an important feature in Windows 2000 TCP/IP.

When the TCP/IP stack is first initialized, ARP requests are broadcast, requesting IP address resolution for the IP addresses of the local host. If another host replies to any of these ARP request packets, it indicates that the host is already using that IP address. This is called the Duplicate Address Test (DAT).

When a duplicate IP address is detected, the Windows 2000 computer still boots, but IP for the duplicate address is disabled and an error message is displayed. To fix the possible corruption of ARP cache tables on other computers, the computer sending the ARP request rebroadcasts another ARP but fills the Source Hardware Address (HA) field in the ARP request with the hardware address of the computer that sent the ARP response. This technique hopefully fixes the ARP cache table corruption of other computers. The Windows 2000 DAT operation is illustrated in [Figure 3.7](#).

Figure 3.7

Windows 2000 DAT test.

Registry Setting for ARP Broadcast

The number of ARP request broadcasts that are sent is controlled by the `ArpRetryCount` Registry parameter in the `HKEY_LOCAL_MACHINE\SYSTEM\Services\Tcpip\Parameters` key (set to a default of 3).

DHCP Conflicts

DHCP-enabled clients use the DHCP Decline Support mechanism to handle IP address conflicts. It handles an IP address conflict by informing the DHCP server that it is declining the address that caused the duplicate address conflict and requesting a new address from the DHCP server. The DHCP server flags the conflicting address as

unusable.

Windows 2000 TCP/IP Support for Classless Interdomain Routing

Classless Interdomain Routing (CIDR) removes the concept of class from the IP address assignment and management process. The class A, B, C scheme used for network number assignment can be inefficient because addresses are allocated in large blocks of 16MB, 64KB, and 256KB, respectively. The use of subnetting alleviates this problem by subdividing a class A, B, or C network.

CIDR eliminates the artificial class boundaries of class A, class B, and class C addresses by specifying a variable number of bits for the network portion. For example, you can specify that the first 23 bits of the network 207.22.128.0 be used to identify the network rather than the standard 24 bits because this address is a class C address. It is expressed in the *net prefix*, or *slash*, notation:

```
207.22.128.0/23
```

In this example, the use of 23 bits for the network portion describes two class C networks: 207.22.128.0 and 207.22.129.0. This form of supernetting is different from subnetting. In subnetting you extend the network bits into the host ID of the address, whereas in supernetting the network bits retreat into the network ID of the address. Recall that an IP address has a network ID and also a host ID whose size is determined by the most significant bits in the IP address. CIDR can be used to supernet class C addresses, and it summarizes them using a single entry in the routing table, thus reducing the size of the routing table. CIDR is used by Internet Service Providers (ISPs) and in the Internet backbone to reduce the number of routing table entries. ISPs are given blocks of class C addresses by the Internet Assigned Numbers Authority (IANA), and they can use CIDR for reducing the number of router table entries. Many commercial IP routers support supernetting.

Using CIDR to Supernet Class A, B, and C Networks

CIDR is described in RFCs 1518 and 1519.

Although subnetting subdivides a class A, B, or C network, CIDR can be used to combine class A, B, or C networks into supernets. CIDR is used primarily to combine a block of class C addresses into a "super class C" network.

Table 3.1 shows the groupings of classless addresses described in RFC 1878, "Variable Length Subnet Table For IPv4."

Table 3.1 Groupings of Classless Addresses

Mask Value (Hex)	CIDR	Decimal	# of Addresses	Classful
80.00.00.00	/1	128.0.0.0	2048MB	128 A
C0.00.00.00	/2	192.0.0.0	1024MB	64 A
E0.00.00.00	/3	224.0.0.0	512MB	32 A
F0.00.00.00	/4	240.0.0.0	256MB	16 A
F8.00.00.00	/5	248.0.0.0	128MB	8 A
FC.00.00.00	/6	252.0.0.0	64MB	4 A
FE.00.00.00	/7	254.0.0.0	32MB	2 A
FF.00.00.00	/8	255.0.0.0	16MB	1 A
FF.80.00.00	/9	255.128.0.0	8MB	128 B
FF.C0.00.00	/10	255.192.0.0	4MB	64 B
FF.E0.00.00	/11	255.224.0.0	2MB	32 B
FF.F0.00.00	/12	255.240.0.0	1024KB	16 B
FF.F8.00.00	/13	255.248.0.0	512KB	8 B
FF.FC.00.00	/14	255.252.0.0	256KB	4 B
FF.FE.00.00	/15	255.254.0.0	128KB	2 B
FF.FF.00.00	/16	255.255.0.0	64KB	1 B
FF.FF.80.00	/17	255.255.128.0	32KB	128 C
FF.FF.C0.00	/18	255.255.192.0	16KB	64 C
FF.FF.E0.00	/19	255.255.224.0	8KB	32 C
FF.FF.F0.00	/20	255.255.240.0	4KB	16 C
FF.FF.F8.00	/21	255.255.248.0	2KB	8 C
FF.FF.FC.00	/22	255.255.252.0	1KB	4 C
FF.FF.FE.00	/23	255.255.254.0	512	2 C
FF.FF.FF.00	/24	255.255.255.0	256	1 C
FF.FF.FF.80	/25	255.255.255.128	128	1/2 C
FF.FF.FF.C0	/26	255.255.255.192	64	1/4 C
FF.FF.FF.E0	/27	255.255.255.224	32	1/8 C
FF.FF.FF.F0	/28	255.255.255.240	16	1/16 C
FF.FF.FF.F8	/29	255.255.255.248	8	1/32 C
FF.FF.FF.FC	/30	255.255.255.252	4	1/64 C
FF.FF.FF.FE	/31	255.255.255.254	2	1/128 C
FF.FF.FF.FF	/32	255.255.255.255	1	A single host route

Routing decisions with CIDR addresses are based on using the number of network bits specified in the net prefix (/n) for that address.

Flow Control Considerations Using ICMP

If a host is receiving datagrams faster than it can process them, the host might send an ICMP Source Quench message requesting the sender to slow down. Most implementations of TCP/IP ignore the Source Quench message because they believe that the problem of flow control should be solved by the Transport layer, such as TCP. TCP indeed solves the flow control problem using sequence numbers to keep track of bytes sent and a window size whose value is set to the number of bytes that can be sent without waiting for an acknowledgment.

RFC 1122, "Requirements for Internet Hosts—Communication Layers," states that ICMP Source Quench messages must be reported to the Transport layer, which should implement a mechanism to respond to the source quench.

The TCP/IP stack in Windows 2000 honors the Source Quench message, as long as it is sent in response to a message sent in an active connection by the Windows 2000 computer. Windows 2000 computers that act as routers do not send ICMP Source Quench messages. If a Windows 2000-based router cannot handle incoming datagrams, it simply drops those datagrams that exceed its buffer capacity—no ICMP Source Quench messages are sent.

Use of Internet Group Management Protocol (IGMP) in Windows 2000

IGMP allows a group of IP nodes to be identified and addressed using a special IP multicast address. *IP multicasting* is the transmission of an IP datagram to a *host group* that is identified by a single IP destination address (class D address for IPv4).

Windows 2000 implements IGMP version 2, as described in RFC 1112 and RFC 2236. Some Windows NT and Windows 2000 components use IGMP. For example, ICMP router discovery is done using multicasts by default. Windows Internet Naming Service (WINS) servers use multicasting in attempting to locate replication partners.

In IGMP, host group membership is dynamic—hosts can join and leave groups at any time and do not have any restrictions on their location or number. Any host can send a datagram to a multicast group.

TCP Compliance Standards

Although RFCs make recommendations and exhortations about compliance to standards, not all TCP implementations conform to standards. Most of the noncompliance issues are, fortunately, minor and result in peculiarities of TCP implementation rather than in any major incompatibility.

Table 3.2 lists the standard multicast address assignments used with IGMP.

Table 3.2 Standard Multicast Address Assignments

Multicast Address	Description
224.0.0.0	Base address (Reserved)
224.0.0.1	All systems on this subnet
224.0.0.2	All routers on this subnet
224.0.0.3	Unassigned
224.0.0.4	DVMRP Routers
224.0.0.5 OSPFIGP	OSPFFIGP all routers
224.0.0.6 OSPFIGP	OSPFFIGP designated routers
224.0.0.7	ST routers
224.0.0.8	ST hosts
224.0.0.9	RIP2 routers
224.0.0.10	IGRP
224.0.0.11	Mobile-Agents
224.0.0.12- 224.0.0.255	Unassigned
224.0.1.0	VMTP managers group
224.0.1.1	Network Time Protocol (NTP)
224.0.1.2	SGI-Dogfight
224.0.1.3	Rwhod
224.0.1.4	VNP
224.0.1.5	Artificial Horizons — Aviator
224.0.1.6	NSS — Name Service Server
224.0.1.7	AUDIONEWS — Audio News Multicast
224.0.1.8	SUN NIS+ Information Service
224.0.1.9	Multicast Transport Protocol (MTP)
224.0.1.10	IETF-1-LOW-AUDIO
224.0.1.11	IETF-1-AUDIO
224.0.1.12	IETF-1-VIDEO

224.0.1.13	IETF-2-LOW-AUDIO
224.0.1.14	IETF-2-AUDIO
224.0.1.15	IETF-2-VIDEO
224.0.1.16	MUSIC-SERVICE
224.0.1.17	SEANET- TELEMETRY
224.0.1.18	SEANET-IMAGE
224.0.1.19	MLOADD
224.0.1.20	Any private experiment
224.0.1.21	DVMRP on MOSPF
224.0.1.22	SVRLOC
224.0.1.23	XINGTV
224.0.1.24	microsoft-ds
224.0.1.25	nbc-pro
224.0.1.26	nbc-pfn
224.0.1.27- 224.0.1.255	Unassigned
224.0.2.1	"rwho" group (BSD) (unofficial)
224.0.2.2	SUN RPC PMAPPROC_CALLIT
224.0.3.000- 224.0.3.255	RFE generic service
224.0.4.000- 224.0.4.255	RFE individual conferences
224.0.5.000- 224.0.5.127	CDPD groups
224.0.5.128- 224.0.5.255	Unassigned
224.0.6.000- 224.0.6.127	Cornell ISIS Project
224.0.6.128- 224.0.6.255	Unassigned
224.1.0.0- 224.1.255.255	ST multicast groups
224.2.0.0- 224.2.255.255	Multimedia conference calls
224.252.0.0- 224.255.255.255	DIS transient groups

232.0.0.0- 232.255.255.255	VMTP transient groups
-------------------------------	--------------------------

In Windows 2000, IP multicasting is supported on AF_INET sockets of type SOCK_DGRAM and SOCK_RAW, which is adequate for most TCP-based applications. IP multicast datagrams are normally sent with a time to live (TTL) of one second, which limits the scope of the multicast datagram to the same subnet.

Time-To-Live (TTL) Value

The TTL value is used to prevent datagrams from circulating uselessly on the network in routing loops. The TTL value is measured in seconds and should be decreased by the amount of time the router takes to forward the datagram. Because routers typically process a datagram in a fraction of a second, they simply decrease the TTL by 1. Thus, TTL, although measured in seconds, acquires the characteristics of a hop count.

The TTL field in IP datagrams is used in tools, such as the Windows 2000 `tracert`, to define an expanding scope of search to find intermediate routers to a final destination. The expanding scope of search is implemented by setting the TTL field to a value of 1 and sending a datagram to the destination. The first router to process the datagram will decrease the TTL field to 0 and send an ICMP message to the sender saying that the TTL field has expired. From this the sender determines the IP address of the first router that is on the path to the destination. Next, the sender sends a datagram to the destination with the TTL field incremented to 2. The first router on the path to the destination decrements the TTL field to 0, at which point another ICMP message is sent from the second router to the sender informing the sender that the TTL field has expired. From this message the sender knows the IP address of the second router on the network path. This process continues until the sender discovers all the routers on the network path to the destination.

An application can change the TTL value and, therefore, the scope of the multicast datagram, by calling the Windows Socket `setsockopt` function. Table 3.3 shows how multicast routers use TTL thresholds to limit the scope of the multicast datagrams.

Table 3.3 TTL Thresholds for Multicast Datagrams

TTL Threshold	Multicast Datagram Restriction
0	Restricted to the same host
1	Restricted to the same subnet
32	Restricted to the same site
64	Restricted to the same

	region
255	Restricted to the same continent

In Windows 2000, an additional route is defined on the host with the multicast destination address 224.0.0.0. This route is added by default on TCP/IP stack initialization and specifies that if a datagram is being sent to a multicast host group, it should be sent to the IP address of the host group through the local interface card, and not forwarded to the default gateway (destination 0.0.0.0). The following route, as seen with the `route print` command, shows the multicast entry:

```

Network Address  Netmask      Gateway Address  Interface      Metric
224.0.0.0       224.0.0.0    63.224.13.116   63.224.13.116  1

```

When a multicast is sent to a broadcast network, such as a LAN, the LAN's multicasting hardware capabilities are used. Both Ethernet and IEEE 802 LANs have a multicasting bit that can be set in the most significant bit of the Media Access Control (MAC) address. The 23 low-order bits of the IP multicast address are placed in the low-order 23 bits of the Ethernet or IEEE 802 net multicast address.

The physical layer multicast address that is used has the following bit pattern:

```
10000000 00000000 011110010 xxxxxxxx xxxxxxxx xxxxxx
```

Bits marked as `x` can be any bit values. Because the first `x` bit is set to 0, the physical layer multicast now becomes

```
10000000 00000000 011110010 0xxxxxxx xxxxxxxx xxxxxxxx
```

The IP multicast is a class D address whose most significant bits are 1110. The bit representation for any multicast address is

```
1110xxxx xxxxxxxx xxxxxxxx xxxxxxxx
```

Relationship Between MAC and IP Addresses

IP addresses are 32-bit values defined at the Network layer of the OSI model. MAC addresses are 48-bit values defined at the Data Link layer of the OSI model.

In IP, the ARP is used to provide the correspondence between IP and MAC addresses.

The `x` positions can be any bit values. Only the low-order bits of the IP multicast address are used to form the physical multicast address. These low-order bits are marked with `y` in the following multicast address format:

```
1110xxxx xxxxxxxx yyyyyyyy yyyyyyyy yyyyyyyy
```

The `x` bits in the previous class D multicast address are ignored in forming the physical multicast

address, which now becomes

```
10000000 00000000 011110010 0YYYYYYY YYYYYYYY YYYYYYYY
```

The translations described in the process are done so that it is easy to translate an IP multicast address to a physical multicast address on a LAN.

Another point to note in the previous discussion is that because only the lower 23 bits of the IP multicast address are used in the physical multicast address, many IP multicast addresses would map to the same physical IP multicast address. Consider the following IP multicast address:

```
1110xxxx xYYYYYYY YYYYYYYY YYYYYYYY
```

Regardless of the values of the *x* bits, the IP multicast address maps to the following single physical IP multicast address:

```
10000000 00000000 011110010 0YYYYYYY YYYYYYYY YYYYYYYY
```

Because several IP multicast addresses map to the same physical multicast address, the host might pick up IP multicast datagrams that it does not want. For this reason, the *x* bits in the IP multicast are typically set to zero. With the *x* bits set to zero, the IP multicast addresses have the following format:

```
1110000 0YYYYYYY YYYYYYYY YYYYYYYY
```

In dotted decimal notation, these are IP addresses in the following range:

```
224.0.0.0 to 224.127.255.255
```

Windows 2000 Media Sense Technology

Windows 2000 includes a *media sense* feature that is available via NDIS 5.0. Media sense enables a network adapter to notify the protocol stack of media-connect and media-disconnect events. Without media sense, if a computer (such as a laptop) is from a network and connected to another network without rebooting, the protocol stack is unaware of the move and the existing IP configuration parameters might be invalid. Under these conditions, any attempt to use the network will fail and possibly cause problems on the new network.

Without media sense, even if the computer is shut off and reconnected to the new network, problems will occur. The TCP/IP stack is not aware that the NIC was disconnected, and the old parameters will be used unless they are explicitly reconfigured or DHCP is used to obtain a new lease for the TCP/IP parameters.

Media sense enables the protocol stack to handle media disconnect events and perform actions such as invalidating stale IP parameters. If the Windows 2000 computer is disconnected from a network, after an interval of 20 seconds the protocol stack invalidates the TCP/IP parameters for the network interface. In Windows 2000, you can make the media connection status visible on the taskbar by selecting the option labeled Show icon in taskbar when connected in the Local Area Connection Properties dialog box (see [Figure 3.8](#)). One way to get to this dialog box is to right-click on My Network Places and select Properties. Then right-click on Local Area Connection and select Properties.

Figure 3.8

Using media sense to make the network connection status visible.

Placing the cursor on the network icon shows the current status and number of packets sent and received in the lower right corner (see [Figure 3.9](#)). When the network is unplugged, you see the disconnect message reported by media sense (see [Figure 3.10](#)).

Figure 3.9

Connection status.

Figure 3.10

Offline status.

What happens when you reconnect the Windows 2000 computer's network cable when media sense is enabled? In this case the Windows 2000 computer first tries to ping the DHCP server if it knows one. If the Windows 2000 computer gets no response from a DHCP server, the computer tries to ping the default router it knows about from the previous configuration. If it gets a response from a default router, the Windows 2000 computer keeps its previous address and assumes that the DHCP server is unavailable. If the Windows 2000 computer gets no response from the default router, the computer assigns an address from the unregistered private IP address in the address block 169.254.0.0/255.255.0.0.

Transmission Control Protocol (TCP) Implementation Issues

Windows 2000 supports the most recent developments and improvements in the performance of the Transmission Control Protocol. Performance improvements include the following:

- Windows 2000 TCP Keep-Alive Messages
- Windows 2000 TCP TIME-WAIT Delay
- Windows 2000 TCP Connections for Multihomed Computers
- Windows 2000 TCP Receive Window Size Calculation and Window Scaling
- Windows 2000 TCP Delayed Acknowledgments
- Windows 2000 TCP Selective Acknowledgment
- Windows 2000 TCP Timestamps
- Windows 2000 TCP Path Maximum Transmission Unit (PMTU) Discovery
- Windows 2000 TCP Dead Gateway Detection
- Windows 2000 TCP Retransmission Behavior

- Windows 2000 TCP Slow Start, Congestion Avoidance Algorithm
- Windows 2000 TCP's Avoidance of Silly Window Syndrome (SWS)

Keep-Alive Messages

Like many TCP implementations, Windows 2000 TCP sends keep-alive messages to verify that the computer at the remote end of a connection is still available.

In a TCP virtual circuit, a sequence number is used at the start of the connection to keep track of the number of octets that have been sent. The remote TCP host sends acknowledgments that contain the sequence number of the octet that has been received successfully so far. This number is used by the sender to determine which octets have been successfully sent. A TCP keep-alive message is a TCP ACK packet with the sequence number set to one less than the current sequence number for the TCP virtual circuit. A host receiving such an ACK responds with a TCP ACK message that contains the current sequence number.

In Windows 2000, TCP keep-alives are disabled by default. When the TCP keep-alive messages are enabled, they can be controlled by Registry settings. TCP keep-alives can be sent once every `KeepAliveTime` milliseconds (the default is set to 7,200,000 milliseconds = 2 hours) if no data has been sent over the TCP connection during this period. If the TCP connection is still valid, a response is received immediately. However, if the keep-alive message gets no response, the message is repeated once every `KeepAliveInterval` seconds (the default is 1 second). An application can use the Windows Socket `setsockopt` function to enable TCP keep-alives. The TCP/IP Registry parameters can be found under the Registry key `HKEY_LOCAL_MACHINE\SYSTEM\Services\Tcpip\Parameters`.

NetBIOS over TCP/IP (NBT) connections also use keep-alive messages. A NetBIOS keep-alive is sent by default once per hour and can be controlled by the `SessionKeepAlive` Registry parameter under the `HKEY_LOCAL_MACHINE\SYSTEM\Services\Tcpip\Parameters` key. The NBT keep-alive messages appear as data for the TCP connection because they are encapsulated in TCP headers. For this reason TCP keep-alives are not needed for a NBT connection. Note that because NBT can be disabled in a native mode Windows 2000 environment, in this case you do not see any NBT keep-alive messages. However, because NBT is not disabled by default, you see the occurrence of NBT keep-alive messages.

TIME-WAIT Delay

The operation of TCP can be described in terms a *finite state machine*, which is a logical model of the behavior of a system that changes with the occurrence of specified events.

After entering the TIME-WAIT state when closing the connection, a delay of twice the Maximum Segment Lifetime (MSL) time interval occurs. This delay is implemented to avoid duplicate segment numbers.

In Windows 2000, a delay in the TIME-WAIT state is set to four minutes to avoid the reuse of the full association parameters (the source IP address, source port number, destination IP address, and destination port number) for the TCP virtual circuit that was just closed. The TIME-WAIT state does

not affect other TCP connections with different full association parameters, which can be opened immediately.

With a wait of four minutes as the default setting, some network applications that open many connections in a short time might use up all available port numbers before they are marked for reuse. Windows 2000 use two Registry parameters to help with this situation:

- `TcpTimedWaitDelay`—Controls the wait in the TIME-WAIT state and can be set as low as 30 seconds.
- `MaxUserPorts`—Controls the number of application-accessible temporary ports that can be used as source port numbers. When a Windows 2000 application requests a port number for opening a TCP connection, an available port from the range 1024 through 5000 is used.

The `MaxUserPorts` parameter can be used to set the maximum port number an application can use. Setting this number to a higher value can make more port numbers available. For example, setting `MaxUserPorts` to 20000 makes 20000 – 1024 (the first temporary port number that can be assigned) equal to 18,976 available ports.

Windows 2000 also maintains a `MaxFreeTcbs` Registry parameter. It controls the number of cached (preallocated) Transport Control Blocks (TCBs) that are available. A *Transport Control Block* is a data structure that is maintained for each TCP connection. You might want to increase it if applications are running out of TCBs because TCP connections are being created rapidly, with an insufficient amount of time for the TCBs to be recycled.

The TCBs are quickly found and accessed by using a hash table. If you increase `MaxFreeTcbs`, you should also increase the Registry parameter `MaxHashTableSize`. The `MaxHashTableSize` value should be set to a power of 2 (for example, 1024, 2048, or 4096) and has a default value of 512. If this value is not a power of 2, the system configures the hash table to the next higher power of 2 value. This value controls how fast the system can find a TCP control block.

TCP and UDP Port Numbers

RFC 1700, "Assigned Numbers" describes the ranges of TCP and UDP port numbers that are available. TCP and UDP port numbers are 16 bits in size and range from 0 to 65535. Port number 0 is reserved. Port numbers up to 1023 are well-known port numbers, documented in RFC 1700. Applications assign temporary port numbers starting from 1024.

Running Out of TCBs

If an application is creating many TCP connections (such as TCP connections being created by many threads running in parallel) and they are not being closed because of delays or unexpected conditions on the network, you run the risk of running out of TCBs (an error message lets you know). Monitor the system log using Event Viewer to detect

this condition.

Connections for Multihomed Computers

There are two situations for TCP connections with multihomed machines:

- TCP connections to a multihomed computer
- TCP connections from a multihomed computer

These situations are described in the following subsections.

TCP Connections to a Multihomed Computer

A computer configured with more than one IP address is a *multihomed* system. Multihoming generally implies that multiple network adapters are on different physical networks. However, multihoming can also be implemented by assigning multiple IP addresses for a network adapter.

Because a multihomed computer has more than one IP address associated with it, the computer can be reached by more than one IP address. If a connection is made by an explicit IP address, that address is used. If a connection is made using a name, the name has to resolve to an IP address. The name can resolve to more than one IP address using WINS or Domain Name System (DNS), in which case one of those IP addresses must be selected.

Regardless of whether a WINS client or DNS client—also called the Domain Name Resolver (DNR)—is used, Windows 2000 attempts to determine whether any of the destination IP addresses for the destination is on the same subnet as any of the interfaces in the local computer. If the destination addresses are on the same subnet as one of the interfaces, they are given a higher priority because the destination can be reached by a directly connected network. These addresses are sorted and appear at the beginning of the list so that the application can try them before trying addresses that are not on the same subnet.

If none of the destination addresses is on a common subnet that connects to the local computer, the behavior depends on whether WINS or DNS was used to resolve the name.

WINS servers return a list of addresses that are in the same order. If WINS is used, therefore, the client is responsible for randomizing or load balancing between the destination IP addresses, by randomly picking an address from the list returned by the WINS server.

Using DHCP to Obtain an IP address

If you add an additional network adapter and do not configure it, Windows 2000 TCP/IP uses DHCP to obtain an IP address for the interface. If there is no DHCP response, an unregistered private IP address, such as from the address block 169.254.0.0/255.255.0.0, is used.

DNS servers return a list of addresses in a round robin fashion. If DNS is used, therefore, the DNR does not attempt to further randomize the addresses.

In some situations you might want to connect to a specific interface on a multihomed computer. You can accomplish this by providing each interface with its own DNS entry. For example, a computer named `embers` could have one DNS entry listing both IP addresses. An example would be the following zone data records in the DNS zone data file:

```
embers.xyz.com.      IN  A    63.2.6.9
embers.xyz.com.      IN  A    192.45.22.2
```

The DNS has two separate records with the same name. When `embers.xyz.com` is resolved, the answer contains two address records. However, at times you might want to treat the interfaces on the multihomed computer differently. For example, you can call the first interface `embers-1` and the second interface `embers-2` and associate records for their separate addresses. Using separate names for the interfaces allows you to query for them separately. As an example, you might want to do a diagnostics test on the second interface by pinging `embers-2.xyz.com`.

```
embers-1.xyz.com.    IN  A    63.2.6.9
embers-2.xyz.com.    IN  A    192.45.22.2
```

TCP Connections from a Multihomed Computer

When TCP connections are made from a multihomed Windows 2000 computer, the application makes the request, but the IP layer has to decide through which of its interface addresses the TCP connection should be made.

If DNS is used to resolve the destination name, TCP attempts to connect using the host routing table. The IP address that is used to match entries in the host routing table is selected from the list of addresses returned by the DNS. If an interface in the host routing table is on the same subnet as the target IP address, the connection originates from that interface to the target IP address. This is done to minimize the number of hops a datagram has to traverse. If there is no best interface to use, the system selects any one of the network interfaces to originate the connection.

DNR Sorting

The `PrioritizeRecordData` TCP/IP Registry parameter can be used to prevent the DNR component from sorting local subnet addresses to the beginning of the list. This parameter controls whether the Domain Name Resolver sorts the addresses that are returned in response to a query for a multihomed host. By default, the DNR sorts addresses that are on the same subnet as one of the interfaces in the querying computer to the top of the list. This is done to give preference to a common-subnet (nonrouted) IP address whenever possible.

If the TCP connection is made by NBT, the NetBIOS application does not have routing information. The reason is that NetBIOS is an application-level protocol and has no direct knowledge of IP. The

network redirector component attempts to open connections on all the transports bound to NetBIOS by using an IP address for each interface as its source IP address. If more than one TCP call succeeds, the redirector cancels all but one. The choice of which TCP connections to cancel depends on the redirector `ObeyBindingOrder` Registry parameter. If `ObeyBindingOrder` is set to 0 (the default value), the transport protocol that is first in the binding order has the highest preference. If `ObeyBindingOrder` is set to 1, the binding order is ignored. In this case the redirector selects the first connection that succeeds and cancels all connections made through other transport protocols.

Windows TCP Receive Window Size Calculation and Window Scaling

TCP uses a sliding-window mechanism to handle flow control. Each endpoint of the TCP connection maintains a window size that it advertises to the remote endpoint. The TCP window size at any time equals the number of additional octets the receiver can accept without overflowing its buffer. Each sending side can send data up to the window size advertised by the receiver. Only when an acknowledgment is received for data already sent can the window move up. The Windows 2000 TCP/IP stack uses larger default window sizes than earlier Windows versions and is set to even increments of the maximum segment size (MSS) when a TCP connection is established. The default window size is calculated as follows:

1. The connection request advertises an initial receive window size of 16KB (16,384).
2. When the connection is accepted by the remote end, the TCP MSS is negotiated. The receive window size is rounded up to an even increment of the negotiated maximum TCP segment size (MSS).
3. If the receive window size adjusted in step 2 is not at least four times the MSS, it is adjusted to 4 x MSS, with a maximum size of 64KB unless a window scaling option as specified in RFC 1323 is in effect.

The receive window can be set using the `TcpWindowSize` Registry parameter or by calling the Windows Sockets function `setsockopt` for each connection.

TCP Window Size

For Ethernet, the window is normally set to 17,520 bytes, which is 16KB rounded up to twelve 1460-byte segments.

The Window size is sent in a 16-bit field in the TCP header and can describe a maximum window size of 64KB. Windows 2000 implements the features described in RFC 1323, "TCP Extensions for High Performance," for specifying larger window sizes than 64KB. Larger window sizes can improve performance on high-bandwidth, high-delay networks. RFC 1323 details a method for supporting scalable windows by allowing TCP to negotiate a scaling factor for the window size when a connection is established. This factor allows for an actual receive window of as much as 1 gigabyte (GB). A new 3-byte TCP Window Scale option might be sent in the open connection TCP packet. The Window Scale option indicates that the TCP is prepared to do both send and receive window scaling, and the scale factor to be applied to its receive window is sent. A TCP endpoint that is prepared to scale windows should send the option, even if its own scale factor is 1. The scale factor is

limited to a power of 2, and only the exponent is encoded, so it might be implemented by binary shift operations. [Figure 3.11](#) shows the TCP Window Scale option (or WSopt).

Figure 3.11

TCP Window Scale option (WSopt).

Both endpoints of the TCP connection must send the Window Scale options in their SYN TCP segments to enable window scaling in either direction. If window scaling is enabled, the TCP endpoint right-shifts the actual window size value by `shift.cnt` bits and places this value in the Window size field. The value `shift.cnt` might be 0, which means that it is offering to scale but using a scale factor of 1 to the receive window. A left shift by `shift.cnt` bits multiplies the window size by 2.

The Windows scaling option might be sent in an initial SYN segment (open connection TCP packet) or in a segment with the SYN bit on and the ACK bit on (passive open). A Window Scale option in a segment with the SYN bit set to 0 should be ignored. (TCP data packets and acknowledgment packets have the SYN bit set to 0.) The Window size in a SYN or SYN/ACK segment is never scaled. The scaling begins after the Windows Scaling option has been accepted.

WSopt:

```
00000101 00000101 00000110
Kind=3      Length=3 shift.cnt=6
```

If the Windows size (16 bit) that is sent in a TCP segment has a size of 32KB, the scaled window size is 2,097,088 bytes, which is obtained by left-shifting the number 6 bits or multiplying it by 2^6 , or 64. Thus, the window size of 32767 multiplied by 64 is 2,097,088:

```
Window size          =          111 1111 1111 1111
Left-shift 6 bits = 1 1111 1111 1111 1100 0000 = (2,097,088 bytes)
```

The scale factor is not required to be symmetrical and can be different for each direction of data flow. Windows 2000 uses window scaling automatically if `TcpWindowSize` is set to a value greater than 64KB and the `Tcp1323Opts` Registry parameter is set appropriately. The `Tcp1323Opts` Registry parameter is normally set to Off by default.

Windows 2000 TCP Delayed Acknowledgments (RFC 1122)

Windows 2000 TCP uses delayed ACKs specified in RFC 1122 to reduce the number of packets sent. This approach was originally popularized in the BSD UNIX implementation of TCP, where acknowledgments were sent for every other TCP segment. In this approach, TCP sends an acknowledgment back only if no ACK message is sent for the previous segment received or if a segment is received but no other segment arrives within 200 milliseconds (the value of the ACK timer) for that connection.

These conditions imply that an ACK is sent for every other TCP segment received on a connection, unless no segment is sent for 200 milliseconds. The delayed ACK timer can be adjusted through the `TcpDelAckTicks` Registry parameter.

Windows 2000 TCP Selective Acknowledgment

Windows 2000 TCP supports a performance feature known as *Selective Acknowledgment (SACK)*, which is described in RFC 2018, "TCP Selective Acknowledgement Options." In TCP, acknowledgments are cumulative. Cumulative acknowledgments have the advantage that lost acknowledgments do not force retransmission, because even if one acknowledgment is lost in transmission, subsequent acknowledgments acknowledge all data that has been received so far. However, cumulative acknowledgments are not efficient when there is a gap representing lost data and data after the gap has been received. Data received after the gap occurs cannot be acknowledged unless the missing data in the gap is received. The sender does not receive information about the successful transmission of data after the gap because the receiver can acknowledge data only before the gap occurs.

Without SACK, a receiver could only acknowledge the last sequence number of an octet stream that had been received, or the left edge of the receive window. When SACK is enabled, the receiver can selectively acknowledge other blocks of received data individually. If there is a gap in the data received, therefore, the receiver can acknowledge the data segments that were received after the gap. The sender then does not have to retransmit data already received successfully. The selective acknowledgment is sent in a special SACK option in the TCP header.

The SACK Permitted option (see [Figure 3.12](#)) is a 2-byte option sent on a SYN TCP segment to indicate that the sending TCP can perform selective acknowledgments. The Length field in the SACK Permitted option refers to the entire length of that option (in this case, 2).

Figure 3.12

TCP SACK Permitted option.

The actual segment to be retransmitted is sent by a TCP receiver as a special SACK option. The SACK option (see [Figure 3.13](#)) is variable in length and can send information on blocks of data received successfully as pairs of numbers that represent the left edge and right edge of the block of data.

Figure 3.13

TCP SACK option.

In Windows 2000 TCP, the SACK Permitted option is enabled by default. If TCP segments are dropped, the receiver can inform the sender of exactly which data blocks have been received and where are the gaps in the data. The sender can then retransmit only the missing gaps rather than the entire segment. In Windows 2000, the SACK is controlled by the `SackOpts` Registry parameter.

Windows 2000 TCP Timestamps

Windows 2000 supports the TCP Timestamps option, described in RFC 1323, "TCP Extensions for High Performance." TCP uses the Positive Acknowledgment Retransmission (PAR) scheme to overcome problems with lost TCP messages. When the TCP sends a message segment, it sets up a retransmission timer whose value is set to an estimate for how long it takes for the acknowledgment to come back. If an acknowledgment for the message is received within the retransmission timer, TCP sends the next message segment; otherwise, TCP assumes that data is lost and retransmits the message segment. The TCP Timestamp option is used by TCP to accurately measure Round Trip Time (RTT) and set the retransmission timer to this value. The TCP Timestamp option is shown in

Figure 3.14.

Figure 3.14

TCP Timestamp option.

The Timestamps option contains two 4-byte timestamp fields. The first timestamp field (TSval) is set to the current timestamp clock of the sending TCP. The second timestamp field, the Timestamp Echo Reply field (Tsecr), echoes the timestamp value that was sent by the remote TCP in the TSval field of a Timestamps option. The Tsecr field is valid only if the ACK flag is set in the TCP header; otherwise, it is set to zero. A single subtract operation (current time – Tsecr) gives the sender TCP an accurate RTT measurement for every ACK segment. The calculated time is called the Round-Trip Time Measurement (RTTM) mechanism.

A TCP might send the Timestamps option (TSopt) in an initial SYN segment and might send a TSopt in other segments only if it received a TSopt in the initial SYN segment for the connection.

Windows 2000 TCP Path Maximum Transmission Unit (PMTU) Discovery

Windows 2000 TCP implements the PMTU discovery, as described in RFC 1191, "Path MTU Discovery." When two hosts establish a TCP connection, they exchange their TCP MSS values in the SYN TCP segments using the MSS option that is appended to the TCP header. The TCP connection then uses the smallest of the two MSS values for subsequent transmissions. In earlier TCP implementations, a host set its MSS value to the MTU at the Data Link layer minus 40 bytes for the IP and TCP headers. With support for additional TCP options, the combined typical TCP/IP header can be 52 bytes or more. Figure 3.15 shows the relationship between MSS and MTU.

Enabling Timestamps

In Windows 2000, timestamps are disabled by default and can be enabled by setting the `Tcp1323Opts` Registry parameter.

Figure 3.15

Relationship between MSS and MTU.

A TCP sender can set the Don't Fragment (DF) flag in the IP header if it is concerned that the destination has problems with assembling IP fragments. If a router along the path has an interface with an MTU size that is too small to forward the IP datagram, the router attempts to fragment the datagram, unless the DF flag is set. If the DF flag is set in the IP header, the router (that is compliant with RFC 1191) sends an ICMP Destination Unreachable message to the sender. This message contains the MTU for the next hop placed in the low-order 16 bits of the ICMP header field.

When the sender TCP receives the ICMP Destination Unreachable message, it adjusts the MSS for the connection to the MTU (sent in the ICMP message) minus the TCP and IP header size. When the sender TCP resends the data, it is sent as a smaller segment and at least passes through the router's interface that complained that it could not forward the IP datagram because of the small MTU size of the interface.

If a router does not support RFC 1191, it might drop IP datagrams that cannot be fragmented or not correctly report the next-hop MTU in the ICMP Destination Unreachable message. Windows 2000 uses two Registry parameters to detect these types of routers and disable or enable PMTU. The `EnablePMTUBHDetect` Registry parameter adjusts the PMTU discovery algorithm to detect noncompliant routers. The `EnablePMTUDiscovery` parameter can be used to enable or disable the PMTU discovery mechanism.

The Windows 2000 `ping` command has a `-f` (don't fragment) option that can be used to discover PMTU:

```
ping -f -n numberOfPings -l size destinationIPAddress
```

You can adjust the size parameter until the MTU is found. The size parameter describes the data payload and not the protocol overhead of IP and ICMP headers. The ICMP header size is normally 8 bytes, and the IP header size is normally 20 bytes. For Ethernet that has an MTU of 1500, the maximum ping data size is $1500 - 8 - 20 = 1472$ bytes, as verified here:

```
>ping -f -n 1 -l 1472 shield.siyon.com

Pinging shield [63.224.13.114] with 1472 bytes of data:

Reply from 63.224.13.114: bytes=1472 time<10ms TTL=128
Reply from 63.224.13.114: bytes=1472 time<10ms TTL=128
Reply from 63.224.13.114: bytes=1472 time<10ms TTL=128
Reply from 63.224.13.114: bytes=1472 time<10ms TTL=128
Reply from 63.224.13.114: bytes=1472 time<10ms TTL=128

Ping statistics for 63.224.13.114:
    Packets: Sent = 5, Received = 5, Lost = 0 (0% loss),
    Approximate round trip times in milliseconds:s
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

The second `ping` is sent with a data size of 1473. Note that the ping utility reports that the packet could not be sent:

```
>ping -f -n 1 -l 1473 shield.siyon.com

Pinging shield [63.224.13.114] with 1473 bytes of data:

Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
```

```
Packet needs to be fragmented but DF set.
```

```
Ping statistics for 63.224.13.114:
```

```
Packets: Sent = 5, Received = 0, Lost = 5 (100% loss),
```

```
Approximate round trip times in milliseconds:
```

```
Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

Windows 2000 TCP Dead Gateway (Router) Detection

A Windows 2000 computer can be set with more than one default router. The default router entries are described using the following entries in the routing table:

Destination	Subnet Mask	Next Hop
:		
:		
0.0.0.0	0.0.0.0	defaultRoute1
0.0.0.0	0.0.0.0	defaultRoute2
:		
0.0.0.0	0.0.0.0	defaultRouteN

What happens if `defaultRoute1` is down? Should the Windows 2000 computer try to send data through this default router every time? If Windows 2000 tries to use the dead router, long delays would occur as TCP timed out in attempting to send data through the dead default router. Windows 2000 TCP uses dead router detection to detect a failure of the default router and bypass it. RFC 816, "Fault Isolation and Discovery" describes the triggered reselection method to detect default routers.

Windows 2000 TCP detects that a router is not responding after a number of unsuccessful attempts (equal to one-half the Registry value `TcpMaxDataRetransmissions`) to route through the default router. The TCP module then changes the Route Cache Entry (RCE) for that connection to use the next default router. If 25 percent of the TCP connections have been switched to the next default router in this manner, TCP tells IP to change the computer's default router to the one that is working.

If the new default router "dies," the dead default router detection mechanism is used again. If the last default router has been tried, the search returns to the beginning of the list of default routers, in the hope that other default routers are now up. If the Windows 2000 computer is restarted, the first default router on the list is used.

Windows 2000 TCP Retransmission Behavior

TCP starts a retransmission timer when each TCP segment is sent and then waits for an acknowledgment to come back before the retransmission timer expires. This technique is the Positive Acknowledgment Retransmission (PAR) scheme.

In Windows 2000 every new TCP connection request uses an initial retransmission timer value that is set to three seconds. This value can be changed using the `TcpInitialRtt` Registry parameter. If a TCP connection cannot be established, it is retried `TcpMaxDataRetransmissions` times (the default is two times).

For established TCP connections, the number of retransmissions is controlled by the

`TcpMaxDataRetransmissions` Registry parameter (set to 5 by default). The retransmission timeout is adjusted dynamically as per RFC 793, "TCP DARPA Internet Program Protocol Specification" to adjust for the delay characteristics of the connection. A Smoothed Round Trip Time (SRTT) calculation is used as described in the following sidebar.

In some situations TCP retransmits before the retransmission timer expires. This situation commonly occurs when TCP implements a *fast retransmit* feature, described in RFC 2581, "TCP Congestion Control" and explained in the following sidebar.

Calculating Smoothed Round Trip Time (SRTT)

Measure the elapsed time between sending a data octet with a particular sequence number and receiving an acknowledgment that covers that sequence number. This measured elapsed time is the Round Trip Time (RTT). Based on the RTT, compute a Smoothed Round Trip Time (SRTT) as

$$\text{SRTT} = ([\text{alp}] * \text{SRTT}) + ((1-[\text{alp}]) * \text{RTT})$$

[alp] is a weighting factor whose value is between 0 and 1 ($0 \leq [\text{alp}] < 1$). The SRTT is an estimated timeout. The SRTT computation, therefore, takes into account its old SRTT value and the new RTT value and takes a weighted average of these two. When [alp] = 0.5, SRTT is just the simple average of old SRTT and the RTT values:

$$\text{SRTT} = (\text{SRTT} + \text{RTT})/2 \text{ when } [\text{alp}] = 0.5$$

When [alp] is less than 0.5, a greater weight is given to the actual Round Trip Time (RTT). When [alp] is greater than 0.5, a greater weight is given to the previous estimated Round Trip Time (SRTT).

When [alp] = 0 or almost 1, you have the two extremes. When [alp] = 0, SRTT is always set to RTT; and when [alp] is almost 1, a fixed SRTT is used with little deference to the actual RTT that is implied. Needless to say, the two extremes are seldom used in actual practice but are useful for understanding the nature of the weighted average.

The SRTT is then used to compute the retransmission timeout (RTO) using the following:

$$\text{RTO} = \min[\text{UBOUND}, \max[\text{LBOUND}, ([\text{beta}] * \text{SRTT})]]$$

UBOUND is an upper bound on the timeout (for example, one minute), and LBOUND is a lower bound on the timeout (for example, one second). These values ensure that the RTO will fall between LBOUND and UBOUND. [beta] is a delay variance factor to change the sensitivity to the delay. If [beta] is set to 1, RTO will be set to SRTT unless the SRTT value is below LBOUND or above UBOUND. This value ([beta] = 1) makes TCP sensitive to packet loss and does not cause TCP to wait for a long time before retransmitting. However, small delays might cause unnecessary retransmissions. To make TCP less susceptible to small delays, a larger [beta] value might be used. RFC 793 suggests using a value from 1.3 to 2.0.

Windows 2000 resends a segment if it receives three ACKs for the same sequence number and that sequence number is less than the expected one. This number can be changed using the `TcpMaxDupAcks` Registry parameter.

Windows 2000 TCP Slow Start, Congestion Avoidance Algorithm

Windows 2000 TCP implements the slow-start and congestion-avoidance techniques, discussed in RFC 1122. When congestion occurs in a network, TCP can respond by adjusting the timeout and retransmitting when acknowledgments are not received within the timeout interval. However, retransmissions can add to the problems of an already congested network by increasing the amount of data already being processed by the network. If retransmissions increase to a point that the retransmitted data being added to the network approaches the storage capacity of the network, a condition called *congestion collapse* is imminent. When a network is experiencing congestion, a method is needed that stop adding data to the network.

In 1988 Van Jacobson published the paper "Congestion Avoidance and Control in Proceedings ACM SIGCOMM'88." This paper outlines a technique for responding to congestion by reducing the amount of data sent by the sender. Jacobson's slow start algorithm tries to avoid congestion by automatically adjusting the TCP window size and is implemented in all modern TCP implementations. In fact, Jacobson's algorithm along with Karn's algorithm is required to be implemented as per RFC 1122.

In Jacobson's slow start algorithm, TCP keeps track of a second limit, called the *congestion window size*. The actual window size that is used is the smaller of the congestion window size and the TCP receiver window size:

$$\text{Actual window size} = \min \{ \text{congestion window size, TCP window size} \}$$

TCP's Fast Retransmit

A TCP receiver should send an immediate duplicate ACK when an out-of-order segment arrives. An out-of-order segment is one that has a sequence number greater than the next expected one. If a sender receives duplicate acknowledgments, they can be caused by dropped segments, reordering of data segments by the network, or replication of ACK or data segments by the network. A TCP receiver should send an immediate ACK when the incoming segment fills in all or part of a gap in the sequence space.

The TCP sender uses the fast retransmit algorithm to detect and repair the loss detected by incoming duplicate ACKs. The fast retransmit algorithm uses the arrival of three duplicate ACKs as an indication that a segment has been lost. After receiving three duplicate ACKs, TCP retransmits the "missing segment," without waiting for the retransmission timer to expire.

After the fast retransmit algorithm sends the missing segment, the fast recovery algorithm is used, rather than Van Jacobson's slow start algorithm, to determine the transmission of new data until a non-duplicate ACK arrives. Normally, Van Jacobson's slow start algorithm is used when data segments are lost because of network congestion.

The reason for not performing a slow start is that receiving the duplicate ACKs while indicating that a segment has been lost also indicates that several segments were successfully transmitted; therefore, the network is not congested enough to justify the use of the slow start algorithm.

Under normal network conditions, when no congestion occurs, the congestion window size is the same as the TCP window size, which means that the actual window size is the same as the TCP window size. When congestion is detected, such as when TCP segments are lost, Jacobson's algorithm reduces the congestion window size by half. At every successive loss of a segment, the congestion window is again reduced by half. This process results in a rapid exponential decrease in window size. If the loss continues, the TCP window size is reduced to one segment, and only one datagram is transmitted at a time. At the same time, Karn's algorithm is used to double the timeout value before retransmitting. This dramatic reduction is called *exponential backoff*, or *multiplicative decrease*.

Jacobson's algorithm uses a slow start algorithm to recover from congestion. If an acknowledgment is received for a segment, the algorithm increases the congestion window size by one segment. Because the initial window size for a severely congested network is one segment, when an acknowledgment is received, the congestion window size becomes two segments. TCP can then send two segments. If the acknowledgments for them come back, the congestion window size becomes four segments. This technique enables TCP to send four segments; when the acknowledgments for them come back, the congestion window size becomes eight segments. This process continues until the congestion window size is equal to the receiver window size. Because of the exponential increase in congestion window size, it takes only $\log_2(N)$ successful transmissions before TCP can send N segments.

If the congestion window size increases too rapidly, it can add to network traffic before the network has completely recovered from congestion. If congestion is experienced again, the congestion window size shrinks rapidly. This alternating increase and decrease in congestion window size causes the congestion window size to oscillate rapidly. Jacobson's algorithm prevents too rapid an increase in congestion window size by using the *congestion avoidance* technique: When the congestion window size becomes one-half the original TCP window size, the congestion window size increases by one segment only if all segments sent so far have been acknowledged.

Windows 2000 TCP Avoidance of SWS

Windows 2000 implements the avoidance of the *Silly Window Syndrome* (SWS). Silly Window Syndrome is a pathological condition that can occur in TCP implementations. This condition was so named because under certain conditions the TCP transmission took place one octet at a time! A TCP segment consists of only one octet, therefore, and each octet was individually acknowledged! The SWS can occur when any of the following conditions is true:

- The application at the receiver reads data one octet at a time from a buffer that is full, and the receiver TCP sends a window size of 1 in an acknowledgment to the sender.
- The application at the sender generates data one octet at a time, and the sender TCP sends this data immediately.

The overhead of sending the data is enormous because of these reasons:

- A large number of small packets are generated.
 - Each packet is processed separately by each of the Data Link, IP, and TCP layers at the sender and the receiver. Intervening routers must process the packets at the Data Link and IP layers.
 - A large computational overhead is involved in each of the protocol layers because each packet must be individually routed, checksummed, and encapsulated or decapsulated, and sequence and acknowledgment numbers must be computed, handled by send/receive buffers, and so on.
 - The ratio of the protocol headers to actual payload (data delivered) is high. That is, protocol efficiency is low. As a result, data throughput is low.
 - As each octet segment is individually acknowledged, the delays for each octet accumulate to a significant value, and the data throughput is low. For 1,000 octets sent in 1,000 packets over a link with a 50ms delay, the total delay is $1000 \times 50\text{ms} = 5$ seconds. Assuming that the amount of time needed to transmit the packet to the link is small compared to the delay of the link, the overall throughput is only $1,000 \text{ octets} \times 8 \text{ bits per octet} / 5 \text{ seconds} = 1600\text{bps}$.
-

Occurrence of SWS

The SWS conditions can occur if an application is designed or written to read and write one octet at a time. Programmers who are aware of this possibility use Winsock functions to read and write data several octets at a time.

To avoid the SWS on the receiver side, TCP implementations use these heuristics:

- If the receive window size is smaller than a given size, the receiver delays advertising the window size until the receiver window size is at least equal to the following:
$$\min (0.5 \times \text{Receiver buffer size, Maximum Segment Size})$$
- Acknowledgments containing the receiver window size as computed in the preceding example are delayed. This delay reduces network traffic by decreasing the number of acknowledgments. Recall that TCP acknowledgments are cumulative, so a later acknowledgment can acknowledge all data received so far. The risk here is that the delaying of acknowledgments can cause the sender to time-out and retransmit segments. To prevent this situation, ACKs are delayed to a maximum of 500 milliseconds, after which they must be sent. Many implementations delay acknowledgments by no more than 200 milliseconds. Additionally, the receiver should acknowledge every other data segment to ensure that the sender has a sufficient number of round-trip time estimates to implement its adaptive timeout algorithm.

A variation on delaying the acknowledgment is not to delay the acknowledgment but rather to send the acknowledgment immediately and not advertise an increase in window size until the prespecified limit has been reached. This approach is not recommended by TCP standards, however.

To avoid SWS on the sender side, TCP implementations use the Nagle avoidance algorithm. The basic ideas behind Nagle's SWS avoidance algorithm are the following:

- The sender sends the first data segment. After the segment is sent, the sender does not immediately send short size segments even if the sender application sets the PSH flag.
- Data is sent only if the following conditions are true:

There is sufficient data to send to fill a maximum size segment.

An acknowledgment arrives while waiting for data to send.

A preset timeout occurs.

SWS Avoidance

Windows 2000 TCP/IP implements SWS avoidance by sending data only if a sufficient window size is advertised by the receiver to send a full TCP segment. Windows 2000 also implements SWS avoidance on the receiving end by not advancing the receive window in increments of less than a TCP segment.

Nagle Algorithm for SWS in Windows 2000

Windows 2000 disables the Nagle algorithm for NBT connections and SMB-based connections because they send numerous small-size commands.

If an application generates data rapidly, it rapidly fills up the sender TCP buffer to the maximum segment size. TCP sends the maximum segment size without delay. Also, when an acknowledgment is received, the segment data in the buffer is sent without delay, even if it is smaller than the maximum segment size.

If the application generates data slowly, segments are sent when successive acknowledgments are received. Meanwhile, the sender TCP buffer accumulates data.

The Nagle avoidance algorithm can be disabled for real-time applications that require a minimum delay and send large bursts of data at a time.

The Nagle avoidance algorithm for TCP connections can be disabled by using Windows Sockets to set the TCP_NODELAY socket option. Disabling the Nagle algorithm can increase network utilization while improving the performance of applications in some instances. If the network is already congested, the increased network utilization can affect the performance of all network-based applications.

© Copyright Pearson Education. All rights reserved.