Chapter 3
Implementing TCP/IP

In This Chapter
▶ Defining TCP/IP
▶ Microsoft's implementation of TCP/IP
▶ A detailed analysis of TCP
▶ A detailed analysis of IP
▶ Windows Sockets: a definition and the application
▶ Three parts of Internet addressing: the IP address, subnet mask, and default gateway
▶ Windows 2000 Server basic routing

Have you ever wondered what the payoff was from years of US military expenditures? Was it $200 wrenches and other doodads publicly highlighted by former Senator William Proxmire and his “Golden Fleece” awards? No, two of the great payoffs from the huge military buildup that have spanned generations are Transmission Control Protocol/Internet Protocol (TCP/IP) and the Internet itself. Not only has TCP/IP become a de facto standard for internetworking, it is also the default protocol for Windows 2000 Server.

As I prepared this chapter, I promised myself that I wouldn’t drone on about the history of the Internet, Request for Comments (RFCs), and other historical hooey that has been covered in far too many books. In fact, I make two assumptions: First, that you are not a newbie — you know the definition of TCP/IP and have other thicker and more technical resources dedicated specifically to TCP/IP. And second, that perhaps like me, you have trouble sitting still when the going gets boring; if the presentation of TCP/IP (which can be very dry) isn’t exciting, you will drift away and miss the finer points about TCP/IP that are important to catch. Call it attention deficit disorder, but you have my assurance that I’ll cut to the chase and tell you what you need to know about using TCP/IP with Windows 2000 Server. This said, I’d like to take just a few pages to set the foundation for our TCP/IP discussion.
About TCP/IP

Although TCP/IP’s popularity can in part be traced to darn good publicity, it is also an efficient routable protocol that is robust enough to perform well on large corporate networks. Network engineers now favor TCP/IP because it is scalable from the smallest node (a single workstation running TCP/IP for dial-up Internet access) to a local area network (LAN) and even a worldwide enterprise wide area network (WAN). Developers know that TCP/IP has an important role in their lives as they develop client/server WinSock-compliant applications at the upper layers of the OSI model.

Remember the golden rule for TCP/IP: It’s a good fit, and you can say one size truly fits all when discussing protocols.

So important is TCP/IP that a grassroots movement has arisen within the MCSE certification community, claiming TCP/IP should be included as a core exam for the MCSE track. And this claim is with good reason. Perhaps no greater paradigm shift has occurred in network computing than the early- to mid-90s shift to the Internet and use of the TCP/IP protocol. Not only did the move to the Internet catch many (including Microsoft) off guard, but the rapid acceptance of the TCP/IP protocol left more than one from the Novell camp (and the IPX world) briefly concerned about their job prospects.

The standard-bearer

Transmission Control Protocol/Internet Protocol (TCP/IP) has emerged as the standard protocol for not only networked personal computers but also standalone computers that access the Internet. Close to celebrating its 30th birthday, TCP/IP was developed in 1969, at the height of the antiwar protests against the Department of the Defense. It was part of the experiment that created ARPANET, which became the Internet as we know it.

The key point to remember about TCP/IP is that it is routable, scalable, connects unlike systems via FTP and Telnet, and is designed for use with WANs (or internets). Equally important today, don’t forget that TCP/IP is designed for use with the Internet.

Additionally, TCP/IP is an “enabling,” or foundation technology that not only supports Internet connectivity but also Point to Point Protocol (PPP), Point to Point Tunneling Protocol (PPTP) and Windows Sockets. PPP and PPTP are discussed later in this book.

Uppercase “Internet” refers to the global Internet that millions use and enjoy. Lowercase “internet” refers to a private WAN that is connected via routers.

Developers can rejoice because Microsoft’s implementation of TCP/IP supports the Windows Sockets interface, a Windows-based implementation of the Berkeley Sockets interface for network programming (a widely used standard). Developers and users alike can also rejoice because Microsoft’s TCP/IP protocol suite in Windows 2000 Server is older and wiser. Back in the
old NT days, did you ever have to reinstall TCP/IP because it had somehow become mysteriously corrupted? Did the Dynamic Host Configuration Protocol (DHCP) ever fail you? You bet! This iteration of the TCP/IP protocol stack, which was revised for Windows 2000 Server, is greatly welcomed; developers (in general) and users will find that Microsoft's TCP/IP is a robust, scalable cross-platform client/server framework, which grants them success.

By committee: Requests for Comments

A popular saying in professional basketball in the late 1990s is “We win by committee.” That is, everyone contributes to the effort. Well, you can say everyone has contributed to the effort to develop the TCP/IP protocol suite, and more importantly, everyone has contributed to the effort to maintain it. The contributions are in the form of Requests for Comments (RFCs), and the RFC documents go a long way toward helping both developers and network engineers understand the TCP/IP protocol suite and the Internet itself. And unlike those who are media shy, those of us who spend significant time in the TCP/IP community are very interested in "comments" related to implementing TCP/IP.

The standards setting process is managed by the Internet Activities Board (IAB). This is a committee that is responsible not only for setting Internet standards but also controlling the publication of RFCs. Two groups are governed by the IAB: the Internet Engineering Task Force (IETF) and the Internet Research Task Force (IRTF). Whereas the IRTF coordinates all TCP/IP research projects and the like, the IETF focuses on Internet problems and solutions. RFCs are officially published by the IETF with input from the parent organization (IAB), the IRTF, and contributors such as you and me.

In fact, anyone in the networking and development communities can contribute to the TCP/IP standard-making process. Just submit a document as an RFC to the IETF. Crazy? You bet. But it's true. The RFC that you submit might just cut the mustard and become published after extensive editorial review, testing, and consensus among the powers that be.

Many companies in the software and technology fields contribute significant resources to have their implementations of protocols and other networking and developer features adapted as standards by industry boards such as the IETF. Although it is possible to contribute to the TCP/IP RFC process, in reality, this world consists mainly of corporate-level software engineers from companies such as HP, IBM, and Microsoft. Standards implemented via the consensus method in place for TCP/IP RFCs have survived a very political process (and not necessarily of the Justice Department-variety).

Likewise, a manufacturer may elect to implement certain RFCs and ignore others. Table 3-1 shows the full set of RFCs that make up Microsoft's implementation of the TCP/IP protocol suite.
<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>768</td>
<td>User Datagram Protocol (UDP)</td>
</tr>
<tr>
<td>783</td>
<td>Trivial File Transfer Protocol (TFTP)</td>
</tr>
<tr>
<td>791</td>
<td>Internet Protocol (IP)</td>
</tr>
<tr>
<td>792</td>
<td>Internet Control Message Protocol (ICMP)</td>
</tr>
<tr>
<td>793</td>
<td>Transmission Control Protocol (TCP)</td>
</tr>
<tr>
<td>816</td>
<td>Fault Isolation and Recovery*</td>
</tr>
<tr>
<td>826</td>
<td>Address Resolution Protocol (ARP)</td>
</tr>
<tr>
<td>854</td>
<td>Telnet Protocol (TELNET)</td>
</tr>
<tr>
<td>862</td>
<td>Echo Protocol (ECHO)</td>
</tr>
<tr>
<td>863</td>
<td>Discard Protocol (DISCARD)</td>
</tr>
<tr>
<td>864</td>
<td>Character Generator Protocol (CHARGEN)</td>
</tr>
<tr>
<td>865</td>
<td>Quote of the Day Protocol (QUOTE)</td>
</tr>
<tr>
<td>867</td>
<td>Daytime Protocol (DAYTIME)</td>
</tr>
<tr>
<td>894</td>
<td>IP over Ethernet</td>
</tr>
<tr>
<td>919, 922</td>
<td>IP Broadcast Datagrams (broadcasting with subnets)</td>
</tr>
<tr>
<td>950</td>
<td>Internet Standard Subnetting Procedure*</td>
</tr>
<tr>
<td>959</td>
<td>File Transfer Protocol (FTP)</td>
</tr>
<tr>
<td>1001, 1002</td>
<td>NetBIOS Service Protocols</td>
</tr>
<tr>
<td>1034, 1035</td>
<td>Domain Name System (DNS)</td>
</tr>
<tr>
<td>1042</td>
<td>IP over Token Ring</td>
</tr>
<tr>
<td>1055</td>
<td>Transmission of IP over Serial Lines (IP-SLIP)</td>
</tr>
<tr>
<td>1112</td>
<td>Internet Group Management Protocol (IGMP)</td>
</tr>
<tr>
<td>1122, 1123</td>
<td>Host Requirements (communications and applications)</td>
</tr>
<tr>
<td>1134</td>
<td>Point to Point Protocol (PPP)</td>
</tr>
<tr>
<td>1144</td>
<td>Compressing TCP/IP Headers for Low-Speed Serial Links</td>
</tr>
<tr>
<td>1157</td>
<td>Simple Network Management Protocol (SNMP)</td>
</tr>
<tr>
<td>1179</td>
<td>Line Printer Daemon Protocol</td>
</tr>
<tr>
<td>1188</td>
<td>IP over FDDI</td>
</tr>
<tr>
<td>1191</td>
<td>Path MTU Discovery</td>
</tr>
</tbody>
</table>
### RFC Title

<table>
<thead>
<tr>
<th>RFC</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1201</td>
<td>IP over ARCNET</td>
</tr>
<tr>
<td>1231</td>
<td>IEEE 802.5 Token Ring MIB (MIB-II)</td>
</tr>
<tr>
<td>1332</td>
<td>PPP Internet Protocol Control Protocol (IPCP)</td>
</tr>
<tr>
<td>1333</td>
<td>PPP Authentication Protocol</td>
</tr>
<tr>
<td>1518</td>
<td>An Architecture for IP Address Allocation with CDIR*</td>
</tr>
<tr>
<td>1519</td>
<td>Classless Inter-Domain Routing (CDIR)*</td>
</tr>
<tr>
<td>1533</td>
<td>DHCP Options and BOOTP Vendor Extensions</td>
</tr>
<tr>
<td>1534</td>
<td>Interoperations between DHCP and BOOTP</td>
</tr>
<tr>
<td>1541</td>
<td>Dynamic Host Configuration Protocol (DHCP)</td>
</tr>
<tr>
<td>1542</td>
<td>Clarifications and Extensions for the Bootstrap Protocol</td>
</tr>
<tr>
<td>1547</td>
<td>Requirements for Point to Point Protocol (PPP)</td>
</tr>
<tr>
<td>1548</td>
<td>Point to Point Protocol (PPP)</td>
</tr>
<tr>
<td>1549</td>
<td>PPP in High-Level Data Link Control (HDLC) Framing</td>
</tr>
<tr>
<td>1552</td>
<td>PPP Internetwork Packet Exchange Control Protocol (IPXCP)</td>
</tr>
<tr>
<td>1553</td>
<td>IPX Header Compression</td>
</tr>
<tr>
<td>1570</td>
<td>Link Control Protocol (LCP) Extensions</td>
</tr>
</tbody>
</table>

The RFCs listed with an asterisk (*) in Table 3-1 are new additions to Windows 2000 Server versus its predecessor, Windows NT Server 4.0.

Windows 2000 Server also incorporates the functionality of the following draft RFCs that are unnumbered at this time:

- NetBIOS Frame Control Protocol (NBFCP)
- PPP over ISDN
- PPP over X.25
- Compression Control Protocol

If you want to track RFC activity in the TCP/IP and Internet communities, point your browser to review the IAB-published quarterly memo titled “IAB Official Protocol Standard.” Here you will receive the current RFC status for each “protocol” in the TCP/IP protocol suite. You may also download the RFCs from this site.

The IAB home page may be found at [www.iab.org](http://www.iab.org). (Figure 3-1) To be honest, those who are newer to Windows 2000 Server may not be as interested in RFC and visiting the IAB home page as those who have worked in the industry for longer periods of time. Why? My observation is that the longer you’ve been working with networks, the more interested you are in expanding your horizons beyond just the Windows 2000 Server NOS. If this is true, then RFCs are a great place to start that expansion.
To obtain RFCs via FTP, go to the following FTP sites:

- NIS.NSF.NET
- NISC.JVNC.NET
- VENERA.ISI.EDU
- WUARCHIVE.WUSTL.EDU
- SRC.DOC.IC.AC.UK
- DS.internic.net
- NIC.DDN.MIL

**It’s a suite, not just a protocol**

It’s essential to understand that TCP/IP is a protocol suite that spans several layers of the OSI model. It is incorrect to think of TCP/IP as a protocol in a singular sense; it’s not just a networking layer protocol. In fact, TCP/IP doesn’t even map well to the OSI model because it’s based on an alternate networking model called the DOD model (that’s DOD for Department of Defense). It is also known as the Internet Protocol Suite. Where the OSI model has seven layers, the DOD model has four layers that map to the seven layers of the OSI model, as shown in Figure 3-2.
It is interesting to note the following:

- The DOD application layer maps to the upper three layers of the OSI model (application, presentation, and session). This is where you find TCP/IP applications such as Telnet. It is also the home of Windows Sockets and NetBIOS.

- The transport layer is simple. The DOD model and the OSI model map directly to each other. Here is where TCP and UDP reside. Tip: Remember that TCP is connection-oriented and guaranteed; UDP is connectionless and doesn’t guarantee delivery.

- The DOD “Internet” layer maps directly to the OSI network layer. Here you find IP, ARP, ICMP, and so on.

- The network interface is at the bottom of the DOD model. This layer maps to the OSI’s data link and physical layers.
The DOD model provides a comparative framework for understanding TCP/IP in contexts other than the OSI model. That is clear from the points just covered. But, as an MCT instructor, I still enjoy hearing the predictable question, “How does it apply to me?” from my MCSE students. Granted, certification students quickly lose interest in a subject if they can’t see its relevance. Regarding the DOD discussion, you can see that TCP/IP packets captured on a Windows 2000 Server network reference the DOD model (see the Ethernet Type line in Figure 3-3).

Basically, it is important to understand the underlying models that can be used to describe the TCP/IP protocol suite and how it is implemented. When you work with peers from the Windows 2000 Server community, you will find that they assume you know this stuff. If you don’t at least understand the basics, you’ll quickly be lost in the proverbial ether.

Figure 3-3: A reference to the DOD model in a packet capture
Comparing TCP/IP to operating systems

Unfortunately, one thing neither the OSI or DOD model will do for you is conceptually map TCP/IP to the underlying operating system on your computer. This is because both the OSI and DOD models are oriented to explain communications models via a layering approach. A “ring” model describes the world of operating systems, starting basically with the Kernel (see Figure 3-4).

With the operating system ring model, note that the Kernel is in Ring 0 and applications run in Ring 3. Do you know what the difference between running code in Ring 0 and in Ring 3 is? Hint: Ring 3 has protected memory space for applications and Ring 0 is very fast!

A Look at the Protocols

As with a large American farm family, every member or protocol in the TCP/IP protocol suite has a job. These family members contribute to the greater whole — the communication between hosts on a network. And as with a large family, each protocol lives on a certain floor in the house. This was shown previously when the TCP/IP protocol suite was mapped to both the OSI model and the DOD Model. At each layer of the DOD model (see Figure 3-5), data is treated differently, which is similar to the OSI model.
Transmission Control Protocol

The most common protocol used today at the transportation level, Transmission Control Protocol (TCP) provides a reliable, connection-oriented delivery service on top of IP. TCP not only guarantees the delivery of packets, but goes so far as to ensure the proper order of packets. A checksum feature assures that the packet header and data are accurate. If these are not accurate or a packet is otherwise faulty or lost, TCP assumes responsibility for retransmitting the packet. Because TCP is reliable, it is a good upper-level member of the TCP/IP protocol suite to use for client/server applications and for important or critical services such as messaging. In short, TCP is the transport layer protocol of choice for session-oriented transmissions. In Figure 3-6, frames 3–5 show the three-way handshake that is the signature of ACK-based communications involving TCP. Packet 4 is both a response and an acknowledgment. Packet 5 is an acknowledgment that acknowledges packet 4. So many packets, so little bandwidth!

Figure 3-5: DOD layering and communications
As with life, using TCP has tradeoffs. A primary tradeoff from implementing TCP is that the acknowledgment process increases network traffic. Think of it as active listening in a group-counseling scenario. Each participant tells his or her story and someone in the group is responsible for repeating the information shared by the participants back to the group. If you’ve been in a group-counseling scenario, you know what I mean. Well, this form of acknowledgment certainly slows down the overall speed of communications but, arguably, it does increase the quality of the communication (the repetition exercise is a form of acknowledgment). Similarly, with the network, these acknowledgments (ACKs) effectively slow down the network but, of course, increase reliability.

User Datagram Protocol

Meet TCP’s evil brother, who is unreliable to say the least. User Datagram Protocol (UDP), which operates at the transport layer in the DOD model as TCP does, offers connectionless-oriented service without delivery or sequencing guarantees.
UDP has long been a favorite method for transporting multimedia information such as video and sound. Think about it. If you are using CUSEEME on the Internet and an occasional frame is dropped, it’s really not a big deal (it would have the effect of appearing to lower your video frame capture rate). The video might look a little bumpy but otherwise it would be fine.

However, don’t despair about UDP. The application using UDP may handle the error correction and reliability issues. Thus you might achieve the best of both worlds: higher network data transfer speeds because ACKs are not required, at least as far as UDP is concerned, and high-level applications enforcing acknowledgments. An optional UDP checksum value can also validate header and data integrity.

For this checksum value setting, see the UDP Checksum field under the UDP header when viewing UDP-related packets in Network Monitor.

The default value for the UDP Checksum is 0 or not active (see Figure 3-7), as shown via the Display Filter of Network Monitor. Of course, if the checksum capability isn’t implemented, it is of little use.

Figure 3-7: The default value for the UDP checksum
In Figure 3-8, UDP is being used and the checksum value is 0xCA58. This checksum value will vary from packet to packet — it is packet unique. A checksum may be your best ally when you attempt to debug a network communication issue surrounding UDP.

![Image](Figure 3-8: The UDP checksum value)

**Internet Protocol**

Essentially Internet Protocol (IP) provides packet delivery for all of the other protocols within the TCP/IP protocol suite. Computer data under the auspices of IP communications is treated to a best-effort, connectionless delivery system. There is no guarantee that IP packets will arrive at their destination or in order. The checksum feature, shown in Figure 3-9 as value 0x525B, relates only to the integrity of the IP header.

So what, you ask, is the connection between IP and TCP? Well, TCP provides the guarantee; IP offers us the world in terms of addressing. IP provides the routing information necessary so that data may be passed from one host to another. IP also assists in the fragmentation and reassembly of packets. This is accomplished by imposing rules to which routers adhere.
Address Resolution Protocol

Although it is not directly tied to the transport of data and although it is not seen by users and applications, Address Resolution Protocol (ARP) can be considered a maintenance protocol that is important in its support of the TCP/IP protocol suite. Basically, the sending host must somehow map the destination IP address to its respective MAC address (physical hardware address). This is because network communications, at the physical level, look like the diagram in Figure 3-10.

The physical address is acquired by IP via the broadcast of a special inquiry packet (ARP request packet). Because it is a broadcast packet, it is evaluated by all ARP-enabled system on the local IP subnet. The node that "owns" the IP address being queried replies and reports its MAC address to the original sender in an ARP reply packet. This sequence is shown in Figure 3-11.

Frame 543 shows the initiation of the ARP polling process. Notice the Target’s Hardware Address is empty but the Target’s Protocol Address is displayed as 209.20.232.1. Remember that you are trying to resolve the target’s hardware address to its IP address. This is the first step in that process.
Figure 3-10: A packet traveling between network adapters

Figure 3-11: The ARP request packet
Frame 544 (see Figure 3-12) presents the ARP reply packet. Here the target of frame 543 becomes the sender and reports its MAC address as 00902778A9CC, thus resolving the IP address of 209.20.232.11. This information is returned to the computer that was the sender in frame 543 and is the target in frame 544.

Don’t forget that each system maintains an ARP table that caches the type of resolution information just detailed. You may use the ARP utility to view the ARP tables. The correct syntax for this is:

C:>arp -a

Internet Control Message Protocol

Internet Control Message Protocol (ICMP) is a bread-and-butter “maintenance” protocol. Two nodes on a network may share status and error information via ICMP. You must look no further than the ping utility to see ICMP in action. Ping uses the ICMP-based echo request and echo reply packets to detect if a specific IP-based network node is functioning. Thus, while ping is really important in managing your network or routers, it’s safe to say that ICMP is even more so.
The ICMP packet (packet 7) shown in Figure 3-13 initiates the echo command from IP address 131.107.2.214 to 131.107.2.210.

The ICMP packet (packet 8) shown in Figure 3-14 shows the echo reply from IP address 131.107.2.210 to 131.107.2.214.

ICMP is required in each and every IP network implementation.

Internet Group Management Protocol

Yet another maintenance protocol, Internet Group Management Protocol (IGMP) is similar to ICMP in that it provides a method that dictates how devices can share and report status information on an IP network. Interestingly, IGMP uses multicasting (a form of broadcasting) to communicate with all devices contained in the membership of a multicast group. This group is displayed in the Group Address field shown in Figure 3-15. Notice the Group Address option displays the NetBIOS name of all machines detected on the subnet.
Figure 3-14: The ICMP packet

Figure 3-15: IGMP protocol properties
Your router must be configured to allow multicasting to permit IGMP-based packets to pass through.

**Simple Network Management Protocol**

Simple Network Management Protocol (SNMP) is a management protocol that facilitates the exchange of information from SNMP devices on the network. Routers, hubs, and bridges can all be SNMP devices, depending on the device you use.

Be careful about paying the SNMP tax unless you really need this functionality. The SNMP tax is the incremental cash outlay you will incur to purchase a device that is SNMP-enabled rather than one that is not. A perfect example is a DSU. Here the SNMP tax can amount to several hundred dollars or more depending on the DSU model. Note that Windows 2000 Server has limited SNMP management capabilities that are available via Management Information File (MIF) interpretation in System Management Server (SMS), a Microsoft BackOffice component. You will typically use a specific SNMP management device, such as a high-end HP Open View–based solution.


Earlier I said that even Microsoft missed the early signs of the TCP/IP revolution. This is true. Those of us who rushed out to purchase and install Windows NT Advanced Server 3.1 will recall that the default protocol was NetBEUI. This changed with the Windows NT Server 3.5x releases; first IPX and then TCP/IP became the default protocol for Windows NT Server. As many say about Microsoft, they got it right on the third try.

Another sign of Microsoft's late arrival to the TCP/IP party was that it had zero presence in the early days of the commerce use of the Internet. Early arrivals to the Internet were using such things as FTP, Gopher, and Mosaic browsers. Microsoft didn't have a contribution until Pearl Harbor Day 1995 when Bill Gates issued an internal memo proclaiming Microsoft the new Internet company. And they haven't missed a beat since then. In fact, as you shall see, the TCP/IP protocol suite included with Windows 2000 Server includes several features that improve both its performance and its reliability.

Specifically, the TCP/IP protocol stack included with Windows 2000 Server has the following features:

- **Core TCP/IP.** Support for core TCP/IP protocols, including TCP, IP, User Datagram Protocol (UDP), Address Resolution Protocol (ARP), and Internet Control Message Protocol (ICMP). This TCP/IP protocol suite implementation dictates how computers communicate and how networks interconnect.
Network programming support. Support for network programming interfaces such as Windows Sockets, remote procedure calls (RPC), NetBIOS, and network dynamic data exchange (Net DDE).

TCP/IP connectivity utilities. Basic connectivity utilities found in Microsoft’s TCP/IP protocol suite that enable Windows 2000 users to interact with different systems such as UNIX.

LPR. “Server” side of TCP/IP printing for UNIX clients. Prints a file to a host running the Lpdsvc service. Note that “lpd” is line printer daemon in the UNIX community.

FINGER. Can obtain system information from a remote computer that supports the TCP/IP Finger server.

FTP. Enables the bidirectional transfer of files between a Windows 2000 computer and another host running FTP server software.

RCP. Remote Copy Protocol. Copies files between a Windows 2000 computer and a host server running the RCP service (which is a UNIX utility).

REXC. Remote Execution. Runs a process on a remote computer.

RSH. Remote Shell. Runs commands on a server running the RSH service (which is a UNIX utility).

Telnet. Provides terminal emulation to a TCP/IP host running Telnet server software.

Telnet is your best friend in the TCP/IP world when working with WANs. There is perhaps no better test than to Telnet into a distant server and check e-mail. Telnet is a better test than a simple ping because Telnet runs more as an application. For example, just because you can ping a distant location doesn’t mean that you can Telnet into the site. As you will discover, Telnet is incredibly valuable for managing your routers and DSUs.

Tftp. Provides bidirectional file transfers between a Windows 2000 computer and a TCP/IP host running TFTP server software. Communicates via messaging using UDP instead of TCP.

Diagnostic utilities. TCP/IP diagnostic utilities to detect, resolve, and prevent TCP/IP networking problems.

ARP. Resolves hardware or MAC addresses to IP addresses.

Hostname. Returns the computer host name for authentication by the RCP, RSH, and REXEC utilities.

IPConfig. Similar to winipcfg in Windows 95/98 but, quite frankly, not as robust. Part of the problem is that IPConfig is still character-based and its reports are somewhat limited. By contrast, winipcfg is GUI-based and more pleasant to work with.

Lpq. Obtains the status of a print queue on a host running LPD service.
- **Nbtstat.** Returns a list of NetBIOS computer names that have been resolved to IP addresses.
- **Netstat.** Returns TCP/IP protocol session information.
- **Ping.** Tests connectivity and verifies configurations.
- **Route.** Modifies or returns the local routing table.

Route can be used to set up a static route. A static route is extremely valuable in certain conditions. If you work with multiple subnets and multiple routers, it's likely you will learn this command.

- **Tracert.** Displays the path taken by a packet to its destination host. Here again, you will use this utility extensively in your TCP/IP travels. Tracert is frequently employed when resolving possible WAN failures or router configuration errors.

- **Internet support.** Support for Internet, internet, and intranet-based computers. This support includes:
  - **Internet Information Server.** Used for Web publishing and administration
  - **Dynamic Host Configuration Protocol (DHCP).** Used for automatically configuring TCP/IP network nodes
  - **Windows Internet Naming Service (WINS).** Used for dynamically registering and querying NetBIOS computer names
  - **Domain Name System (DNS).** Service used for registering and querying DNS domain names
  - **TCP/IP printing.** Used for accessing UNIX-defined printers or network printers directly connected via a network adapter card (such as HP's JetDirect card, which now uses TCP/IP printing; as an aside, HP's JetDirect card used the DLC protocol almost exclusively back in the "old days.")

A common mistake Windows 2000 Server newbies make is to overload the services required to effectively manage and optimize their network. TCP/IP printing services are often loaded without any reason. I suspect this occurs during setup because they look like the kind of services one should "just load." Know your services and don't overload your Windows 2000 Server unnecessarily.

- **Simple Network Management Protocol (SNMP) agent.** Remote management of your Windows 2000 computer is possible by loading this service and using a management tool such as HP Open View. SNMP support is also included for DHCP and WINS servers with Microsoft's TCP/IP protocol stack.

- **Simple Protocols.** Simple protocols to respond to simple requests — Microsoft's TCP/IP protocol suite enables Windows 2000 to respond to computers that request and support the following: Character Generator, Daytime, Discard, Echo, and Quote of the Day.
• **Path MTU Discovery.** Provides the capability to determine the datagram size for all routers between Windows 2000-based computers and other computers on the WAN.

• **Internet Group Management Protocol (IGMP).** Microsoft TCP/IP supports IGMP. It is typically used by workgroup software products at the upper layers of the OSI model.

Many of these TCP/IP utilities and commands are discussed at length in Chapter 5.

### The TCP/IP Settings in Windows 2000 Server

Where are the Windows 2000 TCP/IP settings stored? In the Windows 2000 Registry, of course! Typically modified via GUI-based applications such as the Local Area Connection Properties property sheet — General tab sheet, basic TCP/IP configuration parameters are fairly straightforward. However, what about modifying parameters such as Time to Live (TTL)? The TTL value is either 128 seconds or 128 hops, depending on which comes first (see Figure 3-16). It is effectively the number of routers that a packet may pass through before being discarded.

![Figure 3-16: The TTL value](image-url)
Many TCP/IP parameters may only be modified via the Registry. Quite frankly, most TCP/IP parameters are configured under the Parameters key found via HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip. The Parameters subkey (see Figure 3-17) houses most of the important TCP/IP configuration parameters that you should be concerned with.

Figure 3-17: The Parameters subkey

Here are important existing values within the Parameters subkey:

- **DataBasePath.** Used by the Windows Sockets interface, it specifies the path to such standard Internet database files as HOSTS, LMHOSTS, networks, and protocols.

- **Domain.** The domain entry you make in the TCP/IP Protocol Properties dialog box on the DNS tab sheet (see Figure 3-18). This value is used by the Windows Socket interface. The Domain is the name of the Internet domain that your computer belongs to. As you know, a domain is nothing more than a group name that has computers associated with it.
EnableSecurityFilters. If the entry value is 1, Windows 2000 Server filters all incoming UDP datagrams, raw IP datagrams, and TCP SYNs (connection requests). Accepted values may be defined via these keys: UdpAllowedPorts, TCPAllowedPorts, and RawIpAllowedProtocols. Interestingly, incoming packets are filtered with respect to the local computers. Packets destined for other computers are not filtered.

ForwardBroadcasts. Entry that specifies if broadcasts should be forwarded between two or more adapters. The enabled state has visible broadcasts forwarded.

Hostname. The name entered in the Host Name field in the TCP/IP Protocol Properties dialog box on the DNS tab sheet. It is the DNS host name of the system. Whenever a Windows Sockets application issues the hostname command, this name is returned.

IPEnableRouter. A value of 1 indicates that the system can route IP packets between the networks it is connected to. Packets are not routed if the value is 0.

NameServer. Lists the DNS servers that will resolve names when queried by Windows Sockets. This value overrides the value in the DhcpNameServer value field (a value supplied to DHCP clients via a scope).

SearchList. Lists domain name suffixes to try when an unsuffixed name cannot be resolved by using DNS. This information is used by the Windows Sockets interface.

Adding Registry Values. Other values that you may add to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters are:
• **ArpCacheLife.** Specifies how long entries will remain in the ARP cache table. An entry remains in the ARP table until it expires or until the table entry is reused.

• **DefaultTTL.** The default Time to Live (TTL) value found in the header of outgoing IP packets. TTL is the number of seconds that an IP packet can live on a network without reaching its destination.

• **KeepAliveInterval.** Calculates the interval between keep-alive retransmissions until a response is received. Basically calculates the wait until the next keep-alive transmission. After the number of transmissions specified in TcpMaxDataRetransmissions are unanswered, the connection aborts.

• **KeepAliveTime.** Specifies the interval that TCP sends a keep-alive packet to verify a connect is still intact.

• **TcpMaxConnectRetransmissions.** The number of times TCP retransmits a connect request before aborting its attempts. Interestingly, this value is doubled with each attempt from its default of three seconds.

• **TcpMaxDataRetransmissions.** The number of times a data segment will be retransmitted by TCP before aborting.

• **TcpNumConnections.** The maximum number of open TCP connections that can occur simultaneously.

• **TcpTimedWaitDelay.** Determines how long a connection stays in a wait state, known as TIME_WAIT, before being closed.

• **TcpUseRFC1122UrgentPointer.** Defines how TCP defines urgent. "1" is based on RFC 1122. "0" or no entry in the Registry uses the “mode” from Berkeley-derived BSD systems.

---

### A Day in the Life of a TCP/IP Packet

Remember the famous photo essay book *A Day in the Life of America*, which showed a snapshot of life in America? It was interesting and introspective. I thought you might enjoy getting into the details of a day in the life of a packet in a similar manner. As stated in RFC 791, which relates to the IP portion of the packet:

> The implementation of a protocol must be robust. Each implementation must expect to interoperate with others created by different individuals. While the goal of this specification is to be explicit about the protocol, there is the possibility of differing interpretations. In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret.

When you get down to packet analysis, you are often left thinking that it’s amazing this stuff works as well as it does. That said, many have asked over the years — just how do you read a TCP/IP packet? Well, this is how you do it!
First, examine the IP portion of the TCP/IP network packet because it comes first in the packet (see Figure 3-19).

As you will discover, by breaking down the IP portion into its discrete components, the analysis is actually easier than it looks. Contrast that to looking at the packet as a whole. That view can be a little overwhelming.

The following list starts with the version setting.

- **Version**: 4 bits. The Version field indicates the format of the internet header. As you can see in Figure 3-19, this is Version 4.

- **Internet Header Length (IHL)**: 4 bits. Internet Header Length is the length of the internet header in 32-bit words and it points to the beginning of the data. The minimum value for a correct header is 5.

- **Type of Service**: 8 bits. The Type of Service provides an indication of the abstract parameters of the quality of service desired. These parameters are used to guide the selection of the actual service parameters when transmitting a datagram through a particular network. Several networks offer service precedence, which treats high-precedence traffic as more important than other traffic (generally by accepting only traffic above a
certain precedence at times of high load). An example of this might be some of the newer networking technologies, such as ATM, that enable you to set priorities for different traffic. With ATM, the major choice is a three-way tradeoff among low delay, high reliability, and high throughput.

The information in Table 3-2 is shown as subcategories underneath the IP:Service Type = 0 (0X0) field.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 0-2:</td>
<td>Precedence</td>
</tr>
<tr>
<td>Bit 3:</td>
<td>0 = normal delay; 1 = low delay</td>
</tr>
<tr>
<td>Bits 4:</td>
<td>0 = normal throughput; 1 = high throughput</td>
</tr>
<tr>
<td>Bits 5:</td>
<td>0 = normal reliability; 1 = high reliability</td>
</tr>
<tr>
<td>Bits 6-7:</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>

The use of the delay, throughput, and reliability indications may increase the cost (in some sense) of the service. In many networks, better performance for one of these parameters is coupled with worse performance for another.

**Precedence.** The type of service is used to specify the treatment of the datagram during its transmission through the internet system. Example mappings of the Internet type of service to the actual service provided on networks such as AUTODIN II, ARPANET, SATNET, and PRNET are given in “Service Mappings.” The sample packet in Figure 3-19 shows “Routine.” Table 3-3 provides the list of precedence options.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>Network Control</td>
</tr>
<tr>
<td>110</td>
<td>Internetwork Control</td>
</tr>
<tr>
<td>101</td>
<td>CRITIC/ECP</td>
</tr>
<tr>
<td>100</td>
<td>Flash Override</td>
</tr>
<tr>
<td>011</td>
<td>Flash</td>
</tr>
<tr>
<td>010</td>
<td>Immediate</td>
</tr>
<tr>
<td>001</td>
<td>Priority</td>
</tr>
<tr>
<td>000</td>
<td>Routine</td>
</tr>
</tbody>
</table>
The Network Control precedence designation is intended only for use within a network. The actual use and control of that designation is up to each network. The Internetwork Control designation is intended only for use by gateway control originators. If the actual use of these precedence designations is of concern to a particular network, it is the responsibility of that network to control the access to, and use of, those precedence designations.

**Total Length: 16 bits.** Total Length is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. Such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 octets, whether they arrive whole or in fragments. It is recommended that hosts only send datagrams larger than 576 octets if they have assurance that the destination is prepared to accept the larger datagrams.

The number 576 has been selected so that a reasonable-sized data block can be transmitted in addition to the required header information. For example, this size permits a data block of 512 octets plus a 64-octet header to fit in a datagram. The maximal internet header is 60 octets, and a typical internet header is 20 octets; this provides a margin for headers of higher-level protocols.

**Identification: 16 bits.** This is an identifying value assigned by the sender to aid in assembling the fragments of a datagram.

**Flags: 3 bits.** In our same packet, the settings are “Last fragment in datagram” and “Cannot fragment datagram.” Here are the possible flag settings:

- Bit 0-5: reserved; must be zero
- Bit 6: (DF) 0 = May Fragment; 1 = Don’t Fragment
- Bit 7: (MF) 0 = Last Fragment; 1 = More Fragments

**Fragment Offset: 13 bits.** This field indicates where in the datagram this fragment belongs. The fragment offset is measured in units of eight octets (64 bits). The first fragment has offset zero.

**Time to Live: 8 bits.** This field indicates the maximum time the datagram is permitted to remain in the internet system. If this field contains the value zero, then the datagram must be destroyed. This field is modified in internet header processing. The time is measured in units of seconds, but given that every module that processes a datagram must decrease the TTL by at least one (even if it processes the datagram in less than a second), the TTL must be thought of only as an upper bound on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded, and to bound the maximum datagram lifetime.

**Protocol: 8 bits.** This field indicates the next-level protocol used in the data portion of the internet datagram.
I

- **Header Checksum: 16 bits.** This is a checksum on the header only. Because some header fields change (as does Time to Live), the header checksum is recomputed and verified at each point that the internet header is processed.

  The checksum algorithm is: The checksum field is the 16-bit ones complement of the ones complement sum of all 16-bit words in the header. In English, it's a calculated value similar to a cyclical redundancy check to ensure that the data packet has “integrity.”

- **Source Address: 32 bits.** This is the source address.

- **Destination Address: 32 bits.** This is the destination address.

- **Options: variable.** The options may appear or may not appear in datagrams. As you can see, none appear in our sample data packet. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation.

- **Padding: variable.** The internet header padding is used to ensure that the internet header ends on a 32-bit boundary. The padding is zero.

And with that, give yourself a round of applause as you’ve reached the end of the detailed IP packet discussion. Now, onward to TCP.

**TCP**

It's time for the details about TCP. TCP is that part of the TCP/IP protocol suite that resides in the middle OSI layers. Its primary responsibilities include assuring reliable delivery by maintaining a connection-oriented session between sender and receiver (see Figure 3-20).

This list, which breaks down the TCP portion of a packet, starts with the source setting.

- **Source Port: 16 bits.** The source port number is the port value of the sender.

- **Destination Port: 16 bits.** The destination port number is the port value of the receiver.

- **Sequence Number: 32 bits.** This is the sequence number of the first data octet in this segment (except when SYN is present). If SYN is present, the sequence number is the initial sequence number (ISN) and the first data octet is ISN + 1.

- **Acknowledgment Number: 32 bits.** If the ACK control bit is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established, this is always sent.
Figure 3-20: The TCP portion of our sample data frame

- **Data Offset:** 4 bits. This is the number of 32-bit words in the TCP header. This indicates where the data begins. The TCP header (even one including options) is an integral number 32 bits long.

- **Reserved:** 6 bits. Reserved for future use, this must be zero. This is exactly the type of field that a software developer, seeking an edge for its technology, might use to fundamentally modify TCP for its own purposes.

- **Flags: 6 bits (from left to right).** This is officially known as Control Bits in RFC 791. This list starts with the Urgent pointer field and ends with a No more data from sender setting:
  - URG: Urgent Pointer field significant
  - ACK: Acknowledgment field significant
  - PSH: Push function
  - RST: Reset the connection
  - SYN: Synchronize sequence numbers
  - FIN: No more data from sender
Our sample data packet in Figure 3-20 has these settings:

- No urgent data
- Acknowledgement field significant
- No Push function
- No Reset
- No Synchronize sequence numbers
- No Fin

**Window: 16 bits.** This is the number of data octets beginning with the one indicated in the acknowledgment field that the sender of this segment is willing to accept.

**Checksum: 16 bits.** The checksum field is the 16-bit ones complement of the ones complement sum of all 16-bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96-bit pseudoheader conceptually prefixed to the TCP header. This pseudoheader contains the Source Address, the Destination Address, the Protocol, and the TCP length. This gives the TCP protection against misrouted segments. This information is carried in the Internet Protocol and is transferred across the TCP/Network interface in arguments or results of calls by the TCP on the IP.

**Urgent Pointer: 16 bits.** This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field is only interpreted in segments with the URG (urgent) control bit set.

**Options: variable.** Options may occupy space at the end of the TCP header and are a multiple of eight bits in length. All options are included in the checksum. An option may begin on any octet boundary. There are two cases for the format of an option:

- **Case 1. A single octet of option-kind**
- **Case 2. An octet of option-kind, an octet of option-length, and the actual option-data octets**

The option-length counts the two octets of option-kind and option-length as well as the option-data octets. Note that the list of options may be shorter than the data offset field might imply. The content of the header beyond End of Option must be header padding (that is, zero).

When adhering to strict RFC, TCP must implement all options. When not adhering to the RFC, software developers may elect not to implement all TCP options. Currently defined options are included in Table 3-4 (the Kind column is indicated in octal).
Table 3-4  TCP Options

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>End of Option List</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>No-Operation</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Maximum Segment Size</td>
</tr>
</tbody>
</table>

Here are the specific option definitions for Table 3-4 to assist you in understanding each possible selection:

- Kind = 0 “End of Option List.” This option code indicates the end of the option list. This might not coincide with the end of the TCP header according to the Data Offset field. This is used at the end of all options — not the end of each option — and need only be used if the end of the options will not otherwise coincide with the end of the TCP header.

- Kind = 1 “No-Operation.” This option code may be used between options; for example, it may be used to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.

- Kind = 2 Length = 4 “Maximum Segment Size.” Maximum Segment Size Option Data: 16 bits. If this option is present, then it communicates the maximum receive segment size at the TCP that sends this segment. This field must only be sent in the initial connection request (that is, in segments with the SYN control bit set). If this option is not used, any segment size is permitted.

- **Padding: variable.** The TCP header padding is used to ensure that the TCP header ends and data begins on a 32-bit boundary. The padding is composed of zeros.

Give yourself another round of applause. You’ve completed the detailed TCP discussion. Whew!

**Internetworking with TCP/ IP**

TCP/IP is especially well suited to take advantage of the Windows 2000 Server paradigm — that is, that Windows 2000 Server is the digital nervous system for an enterprise. Because of its scalability, TCP/IP is an easily a solution for a peer-to-peer network or a global-wide area network for a Fortune 500 corporation.
Out of the box, Microsoft has released its TCP/IP stack for:

- Windows 2000 family (Server, Professional, et. al.)
- Windows NT Server family (3x, 4x)
- Windows NT Workstation family (3x, 4x)
- Microsoft TCP/IP-32 for Windows for Workgroups
- Microsoft LAN Manager

Breeder networks

One thing you can count on today in the networking profession is subnet proliferation once you deploy a networking solution in your organization. Inevitably, you will find your company opening a new branch office that must be connected to the central site. In many cases, you will eat or be eaten by a rival. Whatever the circumstance may be, you’re bound to see your network group (or breed) once deployed. Plan for this in advance, even when it seems like the remotest of possibilities.

Heterogeneous networks

Perhaps the coolest thing about TCP/IP is its capability to communicate with foreign hosts in a way that no other protocol I’m aware of can. These foreign hosts include:

- Host systems on the Internet
- Unix-based host systems
- Macintoshes running TCP/IP protocol stacks such as MacTCP

Note that an Apple Macintosh can communicate and interact with a Windows 2000 server if Services for Macintosh are installed on an NTFS partition. In this scenario, the Macintosh and Windows 2000 server would be running AppleTalk as the common transport protocol.

- IBM-based mainframes
- Open VMS systems
- Printers with network adapters cards such as the HP JetDirect card

Although TCP/IP is obviously the preferred and recommended transport protocol, don’t forget that other transport protocols are supported that allow Windows 2000 Server to be a good citizen on diverse and heterogeneous networks:

- IPX/SPX
- NetBEUI
- AppleTalk
- Third-party protocol stacks such as DECnet are supported
Windows Sockets

Just as Microsoft has selected TCP/IP as its default protocol to ensure wide support and acceptance in the networking community, the Windows Sockets applications programming interface (API) exists to provide a standard framework for developing applications. No doubt this API-based standard framework is provided to enable third-party developers to write network-based applications that will run well on Microsoft-based networks using Microsoft's TCP/IP protocol stack. And thank god for that. Could you imagine the state of affairs on Windows 2000 Server-based networks if such an API weren't provided for developers to hook into underlying services? It wouldn't be pleasant, to say the least.

So what exactly is WinSock, as the Windows Socket API is called? If you were on the proverbial 90-second elevator ride and had to explain it to a layperson, well... you couldn't. But this said, Microsoft refers to it as “...an interface and a communications infrastructure... not a protocol.” Technically, it is an open specification derived from Berkeley Software Distribution (BSD) Unix 4.3 sockets API for TCP/IP. Starting with the original WinSock 1.1 specification, which was created in early 1993, WinSock has now matured to Version 2.0 and is now known as WinSock2. WinSock2 was originally introduced in late 1995 and shipped with Windows NT Server 4.0. It is the version that ships with Windows 2000 Server. WinSock2 is completely backward compatible with applications developed under previous WinSock specifications.

Beyond providing robust backward compatibility, WinSock2 introduced:

- Better performance capabilities. Underlying architectural changes to WinSock2 have taken the original WinSock subsystem and divided it into two layers. One layer is the DLLs providing the Windows Sockets APIs. The second layer is the service providers residing below the DLLs — these interact with upper-layer services via the Service Provider Interface (SPI).

- Name resolution. Basically, developers can write applications that access directory services such as Active Directory, DNS, NDS from Novell, and X.500.

- Concurrent access to multiple network transports. By writing to the SPI, developers can write once and use twice. That is, an application can use both TCP/IP and IPX/SPX under WinSock2.

Third-party TCP/IP software support

Obviously, TCP/IP as an entity doesn't live in a static vacuum. In fact, just the opposite is the case. The design goals from the founding fathers of TCP/IP were to create an open and extensible protocol suite for use by the masses. That key tenet has not only resulted in the widespread acceptance of TCP/IP but has also spurred tremendous advances in TCP/IP tools, applications, and utilities. Developers, knowing that no one company explicitly owns TCP/IP, are emboldened to devote their best and brightest development resources on projects that extend the features, functionality, and manageability of TCP/IP.
In this section, I profile a few third-party solutions that extend TCP/IP beyond its native capabilities in Windows 2000 Server.

It's a fact: Even in Windows 2000 Server, Microsoft's TCP/IP protocol suite is incomplete and lacks certain connectivity utilities and server services (daemons) that are supplied by third-party software vendors, such as NetManage with its Chameleon TCP/IP protocol stack product line. One such service that is lacking is support for NFS or Network File System. In order to communicate with a Sun workstation and a computer running Windows 2000 (either Server or Professional), you must have NFS—Network File System—connectivity between the two dissimilar systems (otherwise you're doomed to ping or Telnet connectivity).

NetManage's Chameleon ViewNow NFS Client (see Figure 3-21) provides NFS-based connectivity between Unix systems and Windows 2000. NetManage is well known in particular for developing applications that are very robust and for fast NFS clients.

![NetManage ViewNow NFS Client](image)

Figure 3-21: NetManage ViewNow NFS Client

The Seattle-based company, WRQ, supplements Microsoft's TCP/IP protocol stack with its connectivity solutions for connecting disparate systems. In particular, WRQ Reflection products are basically terminal emulation applications for HP, Digital, Unix, IBM, and X systems hosts and Windows-based clients. Reflection NFS Connection (see Figure 3-22) works with Windows 2000 Network Neighborhood and Windows Explorer.
Simple routing

By definition, an internetwork needs some form of OSI Layer 3 routing capabilities to connect its subnets. Without routing, each subnet lives in a vacuum and does not communicate fully with the other subnets. However, this discussion is in the context of TCP/IP and internetworking and thus is about the “out-of-the-box” Window 2000 Server routing capabilities. I would highly recommend that you educate yourself on hardware-based routing solutions for vendors such as Cisco.

When you hear term “multihomed,” do you think of the well to do with homes both in the city and the country? The term actually refers to Microsoft TCP/IP’s support for IP routing. You can use these built-in IP routing capabilities in systems with multiple network adapters attached to distinct and separate networks.

Implementing the multiple network adapter card-based routing capabilities of Windows 2000 Server is easy. The configuration is made via the Routing and Remote Access Microsoft Management Console (MMC) (see Figure 3-23).
There are risks associated with native routing in Windows 2000 Server. For example, if your Windows 2000 server is connected directly to the Internet, then enabling this capability is akin to leaving the barn door open. Don't forget that IP routing is a two-way street; be aware that you are potentially inviting unwelcome guests to enjoy your network. This is even a problem in misconfigured implementations of Microsoft Proxy Server. Affectionately called the “poor man’s firewall,” Proxy Server uses a dual network adapter card to create the physical separation between your network and the Internet. However, if you accidentally and incorrectly implement Windows 2000 Server’s routing capabilities, you have effectively defeated and overridden the firewall built by Proxy Server. Take this problem seriously; I’ve seen this mistake and unfortunately, the Proxy Server documentation assumes that you know this problem can occur.

Don’t be fooled that Windows 2000 Server-based IP routing is the same as configuring and using a traditional hardware-based router such as a Cisco 4500. It is not. The IP routing capabilities found in Windows 2000 Server merely forward packets between physically separate subnets. No intelligent decisions are made with respect to IP filtering or cost-based path analysis.

Implementing TCP/IP

Now that you’ve seen TCP/IP, allow me to offer a second view. You have at least three reasons to be attentive to all of the basic TCP/IP issues and definitions presented in this chapter. First, there is the technical side. If you don’t understand the basic components used to implement TCP/IP, you may very well create a network that doesn’t work. An example of this is if you have the wrong subnet mask class associated with a specific range of IP addresses. If you do this, some of your older routers, being the unforgiving devices that they are, won’t work properly. Second, if you lack the TCP/IP fundamentals, you are more likely to make a tactical error. Examples of these errors are endless and include humiliating yourself in front of other network professionals who know TCP/IP. Think of it as native English speakers finding humor in a non-native
English speaker’s attempt to hold a fluent conversation. Third, you may make life harder on yourself. By not understanding the fundamentals of TCP/IP, you risk having to take the TCP/IP certification exam several times instead of only once. You can avoid all of the situations I’ve just described if you simply take the time, and use this chapter, to master the TCP/IP fundamentals.

Internet Addressing

The packet needs to get from here to there. This is the role of TCP/IP addressing. Remember that there are three components to a TCP/IP address: IP address, subnet mask, and default gateway. I encourage you to draw analogies from other “worlds” to appreciate the importance of these components. If this were the U.S. mail system, you might think of the street address as the IP address, the city as the subnet mask, and the zip code as the router. Take a moment to challenge yourself to think “outside” of the TCP/IP box and draw analogies to other parts of your life to better define the IP address, subnet mask, and default gateway (hint: try thinking out how telephone numbers are organized with area code, prefix, and telephone number).

IP addresses

The IP address is a 32-bit value that is used to correctly identify a source and destination address in packet-based network communications. An IP address typically reads something like 204.107.2.100. The address can be broken down into four positions of eight bits each and can also be represented in binary form. These positions are called octet positions and are each eight bits in length.

Make sure you commit to memory Table 3-5, which shows these octet positions. You will inevitably find yourself in a planning meeting or on a telephone call where others drop into what could be called TCP/IP shorthand. You'll hear terms such as “dot one hundred in the fourth octet position” thrown around freely, and you will be expected to keep up.

<table>
<thead>
<tr>
<th>Value</th>
<th>Octet Position</th>
<th>Standard Alpha Variable Notation*</th>
</tr>
</thead>
<tbody>
<tr>
<td>204</td>
<td>First Octet</td>
<td>W</td>
</tr>
<tr>
<td>107</td>
<td>Second Octet</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>Third Octet</td>
<td>Y</td>
</tr>
<tr>
<td>100</td>
<td>Fourth Octet</td>
<td>Z</td>
</tr>
</tbody>
</table>

*This is how IP models typical refer to the octet position when referenced via alpha characters. This is conceptually similar to the study of algebra where discrete items are referred to as either A, B, C, or X, Y, Z.
To represent an address in binary format, let’s revisit binary counting. Binary uses values of one and zero. In other words, it’s the classic on-off mentality of digital computing that we’ve all grown up with and love. Table 3-6, a simple binary conversion table, helps you translate a dotted decimal value to its binary form.

**Note:** we are reading left to right, as with this sentence. So for the binary representation 11001100, which is the number 204, the first bit position of the octet is a one (“1”). While it may seem incredibly obvious to say that we read from left to right, it is necessary to communicate this point correctly because some areas in the TCP/IP and Internet worlds actually read right to left. For example, Internet domain names are technically read from right to left. Thus, the address is first evaluated as a commercial or “com” domain.

<table>
<thead>
<tr>
<th>Octet Position</th>
<th>Decimal Notation</th>
<th>Binary Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Bit</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>2nd Bit</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>3rd Bit</td>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>4th Bit</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>5th Bit</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>6th Bit</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>7th Bit</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>8th Bit</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Thus, the IP address of 204.107.2.100 would translate to the binary representation shown in Table 3-7.

Stated another way, 204.107.2.100 translates to zeros and ones, organized by octets in a column format, as shown in Table 3-8. This table helps you understand how a specific IP address appears when organized by octet.

Don’t forget that an IP address should be thought of as a unique address on a TCP/IP network. Stated again, every node on a TCP/IP network must have a unique IP address. If two nodes on your TCP/IP network have the exact same IP addresses, you will experience unreliable network performance and receive error messages that identical IP addresses exist.
Table 3-7 Translation of IP Address to Binary Values

<table>
<thead>
<tr>
<th>Octet Bit Position</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
<th>5th</th>
<th>6th</th>
<th>7th</th>
<th>8th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octet value/dotted decimal notation</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>204 (first octet)</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>107 (second octet)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2 (third octet)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>100 (fourth octet)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 3-24 shows the error message that would be displayed by the Windows 2000 Server that was first to participate on the TCP/IP network before the second host appeared with a duplicate address. This first host will continue to operate without incident. In short, the first computer has obtained the specific IP address in question and doesn't have to give it up just because a second host wants it.

Table 3-8 Alternative View: Binary to IP Address Conversion

<table>
<thead>
<tr>
<th>First Octet</th>
<th>Second Octet</th>
<th>Third Octet</th>
<th>Fourth Octet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0 0</td>
<td>1 1 0 1 0 1 1</td>
<td>0 0 0 0 0 1 0</td>
<td>0 1 1 0 0 1 0 0</td>
</tr>
<tr>
<td>204</td>
<td>107</td>
<td>2</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 3-24: A Windows 2000 Server IP address conflict error message

The message shown in Figure 3-25 indicates that the computer operator has attempted to authenticate on the same network using an IP address that already exists. The resulting error message means that all networking functionality has been disabled because of the duplicate address conflict. In short, the second host simply can't have the same IP address that is being used by the first host.
Getting around a duplicate IP address scenario is actually quite easy. If you have a second transport protocol loaded and bound to your network adapter, you will be automatically authenticated as a network citizen using the second valid protocol. Of course, I'm assuming the Windows 2000 Server you are trying to attach to is also running this second protocol. Remember that in a nonrouted scenario, the NetBEUI protocol is a fine selection. In a routed scenario, however, you will need to consider using NWLink (IPX/SPX), which is routable.

The discussion so far relates to IP addressing issues on a LAN or a WAN that is not participating on the Internet via a full-time connection. When connected to the Internet, the rules may change, depending on your situation, but the basic rule still applies: any node on the Internet must have a unique address.

Remember, you must be especially careful that those computers directly connected to the Internet and not routed through a proxy server to use an authentic Internet IP address use the correct address — one that has been assigned by the Internet Network Information Center (InterNIC). Given the architecture of TCP/IP and how it is implemented on the Internet, it is essential there are unique IP addresses for nodes with direct Internet connections.

To receive a valid Internet address and connect to the Internet, contact: Network Solutions, InterNIC Registration Services, 505 Huntmar Park Drive, Herndon, VA 22070 (www.internic.net).

The IP green movement

Let's face it, TCP/IP addressing is not as easy as it looks; with a finite supply of IP addresses on the Internet, it is essential that you consider an alternate IP address strategy that enables your internal network nodes to have unique IP addresses while taking full advantage of the Internet. Think of this as the green strategy — you're doing your part to save IP addresses, as many good-hearted souls are trying to save the world's trees from harvest.

Although you can contact Network Solutions to get your unique Internet address (a.k.a. netid), I can offer a better idea.

There is one way to use both phony and real Internet IP Addresses on the same network. This situation requires the use of Microsoft Proxy Server, wherein the internal LAN may have "phony" IP addresses ranging from 131.107.2.100 to .200 and yet the organization may only have one real Internet
IP address, such as 204.222.104.2, for its “real” Internet connection. This is a perfectly legitimate way to connect to the Internet from the desktop and has the added benefit of helping save real Internet IP addresses (an endangered species). Proxy Server, with its routing and filtering capabilities, in effect converts the phony internal IP addresses to route to the Internet via the valid Internet IP address.

**Mysteriously misappropriated Internet domain names**

While writing this book, I twice observed situations in which domain names were “switched” to different Internet IP addresses (something that was unknown to all network parties that needed to be in the know), so I thought it was worth mentioning. Perhaps the mere telling of these stories will prevent you from suffering a similar situation. In each case, the clients noticed that their Internet e-mail service seemed to be failing. Both were running Windows NT Server as their network operating system and Microsoft Exchange 5.x as their e-mail application. Note: this same example applies to networks running Windows 2000 Server.

One day, client 1 (a small software developer) noticed that while the internal Exchange e-mail continued to operate as expected, the externally routed e-mail to the Internet, both internal and external, was failing. Of course, a round of network troubleshooting commenced: routers were tested, telephone company services were looped back, and Exchange services were analyzed. Many billable hours later this discovery was made: the marketing director had apparently taken advantage of, and been taken advantage of by, a discount Web-hosting firm that offered not only to develop your Web site but to host it as well. There was only one catch to this great deal. The Web hosting firm would take control of your domain name and the IP address assigned to your organization and repoint the domain name to the firm’s server. Well, this caused fits with the Exchange-based Internet e-mail service. Once corrected — that is, once the domain name and IP address were pointed back to the client’s original location — the Exchange-based Internet e-mail worked just fine. What is the moral here? Communication. This is a classic case where the marketing department did not communicate with the networking engineers. It sounds like the classic MBA lecture where marketing and engineering don’t speak to each other by default; that is, many MBA case studies and far too many MBA lectures emphasize organizational conflict where departments are not on speaking terms. Perhaps you’ve experienced such political divisions in your own organization.

The second time I encountered a domain name and IP address reassignment was another situation where a management team member, the chief financial officer (CFO), contracted with a bargain Web-hosting firm to develop and host the firm’s Web site. Here again, the sun rose one day and the Exchange-based Internet e-mail failed, although not completely — the performance was somewhat random at first; it would work for a few hours and then fail. After several hours of troubleshooting, including threatening to FDISK (use the
format disk command) the Exchange Server machine and start from ground zero, the discovery was made that the Web-hosting firm had contacted InterNIC and pointed the client’s domain name and corresponding IP address away from the client’s original location. Once realized, the domain name and IP address were restored to their proper locations, and Exchange worked just fine.

So beware of low-cost Web-hosting solutions. More important, beware of other stakeholders in your organization accepting such offers on your behalf.

Just as there is a trend toward low-cost Web-hosting firms reassigning domain names and IP addresses inappropriately, be advised that there is a disturbing trend on the Internet that involves unethical bulk e-mailing organizations. Apparently these organizations have InterNIC reassign legitimate domain names to their illegitimate sites. Consequently, a flood of annoying bulk e-mails are launched and the replies are gathered for several hours or days until the domain name assignment is discovered and cured.

Subnet masks

The subnet mask is used to distinguish the network portion of an address from the host or node portion of an address. It provides information about how an address should be read.

The subnet mask is extremely valuable for organizing IP address assignments on the Internet into classes. Ingeniously, we have Class A, Class B, and Class C. Important reasons exist for using address classes to manage all of the TCP/IP madness on the Internet. Although, as networking professionals, we are only concerned with Classes A, B, and C (shown in Table 3-9), there are two other classes. Class D is used for multicasting to a special group of nodes in a land where we mere mortals don’t participate. Class E is reserved for future use and growth (thank goodness — we’re going to need this class considering the incredible growth of Internet nodes).

Table 3-9 Internet Address Classes Using the 204.107.2.100 Sample IP Address

<table>
<thead>
<tr>
<th>Class</th>
<th>Network Address</th>
<th>Host Address</th>
<th>Bit Position That Separates Network and Host Address*</th>
<th>Standard Subnet Mask Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>204</td>
<td>107.2.100</td>
<td>8th</td>
<td>255.0.0.0</td>
</tr>
<tr>
<td>B</td>
<td>204.107</td>
<td>2.100</td>
<td>16th</td>
<td>255.255.0.0</td>
</tr>
<tr>
<td>C</td>
<td>204.107.2.</td>
<td>100</td>
<td>24th</td>
<td>255.255.255.0</td>
</tr>
</tbody>
</table>

*Remember that an IP address is expressed as a 32-bit value. The division points between octets occur at the 8th, 16th, and 24th bits.
The Microsoft Official Curriculum (MOC) for several Microsoft Certified Systems Engineer (MCSE) courses provides some damaging misinformation for the student who is new to TCP/IP implementations. Let’s play, “What’s wrong with this picture?” This is the actual configuration in the MOC instructor’s manual I used for several of my MCSE courses, including these titles: Implementing and Supporting Windows 95, Supporting Windows NT Server 4.0 – Core Technologies, Supporting Windows NT Server 4.0 – Enterprise Technologies, Supporting Microsoft Internet Information Server, and Supporting Proxy Server:

IP Addresses: 131.107.2.100-.200
Subnet Mask: 255.255.255.0

So I ask you — what’s wrong with this picture? Challenge yourself to find what is fundamentally flawed here. If you can do so immediately, my compliments to you. If you can’t, read on; the following text might enlighten you further as to the fundamentals of TCP/IP and enable you to see what is wrong with this IP address and subnet mask combination.

In the example, the IP has an invalid beginning and ending private Internet network address range given the standard subnet mask value that is used. While this is not a problem on a single subnetted internal private network, it can wreak havoc when you work with “real” routers, such as the hardware-based Cisco offerings; they may or may not expect your addressing to conform exactly to the known valid private Internet network address ranges.

Although it is essential that you are able to identify the potential problems with this configuration, it’s more important that you don’t apply this misinformation to future TCP/IP implementations. Oh yes, it’s also important that you don’t incorrectly answer MCSE examination questions based on this misunderstanding. That, perhaps, is the worst consequence of this goof, given it was delivered in the context of an MCSE certification course!

Table 3-10 shows valid private Internet network address ranges, enabling you to arrive at the correct answer to the previous riddle. Remember that we are talking about the network portion of the IP address. As you shall see in a few moments, the subnet mask helps you distinguish which portion of the IP address is defined as the network address and which portion of the address is defined as the host address.

<table>
<thead>
<tr>
<th>Class</th>
<th>Start of Address Range in First Octet</th>
<th>End of Address Range in First Octet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>001</td>
<td>126</td>
</tr>
<tr>
<td>Class B</td>
<td>128</td>
<td>191</td>
</tr>
<tr>
<td>Class C</td>
<td>192</td>
<td>223</td>
</tr>
</tbody>
</table>
First octet values above 223 (starting with 224) are reserved and may not be used. Also don’t forget the following:

- All network addresses must be unique. This statement holds true whether you are on the Internet or your own internetwork, and it is a good rule to adhere to.
- A network address cannot begin with 127 (the reserved address for internal loopback testing).
- You may not have all zeros in the first octet of a network address. If all bits are set to zero, that indicates a local address and prevents routing.
- You may not have 255 in the first octet of a network address. If all bits are set to one (which equals 255) the eight bits set to the value of one act as a broadcast.

### Host addresses

Regarding the host portion of an IP address, you will find the following points vital in mastering the fundamentals of TCP/IP:

- Host addresses must be unique on a given network. This is a truism. Think about it. Each node on a network must have a unique address, and thus the host portion of the IP address must be unique. Table 3-11 shows valid host portion addresses within an IP address.

In the technical community, the word “host” is interchangeable with the term “node.” That is, the two terms essentially have the same meaning, assuming correct usage.

- All “1” bits may not be used as the host address. If this is the case, the address is interpreted to be a broadcast address, not a host address.
- Likewise, all “0” bits may not be used as the host address. Doing so would communicate: this network only.

<table>
<thead>
<tr>
<th>Class</th>
<th>Start</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class A</td>
<td>w.0.0.1</td>
<td>w.255.255.255</td>
</tr>
<tr>
<td>Class B</td>
<td>w.x.0.1</td>
<td>w.x.255.255</td>
</tr>
<tr>
<td>Class C</td>
<td>w.x.y.1</td>
<td>w.x.y.254</td>
</tr>
</tbody>
</table>
Caste system
Let’s discuss Classes A, B, and C. Perhaps you’ll agree that TCP/IP has such clear class distinctions that it might better be thought of as a caste society. Very important technical reasons exist for having classes. This separation enables you to better organize your network to achieve a best fit given the size of your network. If you are connected to the Internet, the class designation is even more important and is perhaps the most important way in which order is ultimately maintained on the wild and free wide-open world of the Internet.

Class A
Class A assignments are very rare and are for only the largest networks. Theoretically, only 127 Class A networks can exist on the Internet because the network portion of the 32-bit address is on seven of the first eight bits in the first octet. If all eight bits were used in this octet, the number of networks possible would be 128. The unused bit in the octet is referred to as the high-order bit and is reserved. But let’s not stop there. Because the number 127 in the first octet is a reserved number, there can really only be 126 Class A networks.

The number 127 is reserved in the first octet position to facilitate such features as internal loopback testing. To perform an internal loopback test in TCP/IP using the ping command, type `Ping 127.0.0.1`.

The number of nodes possible on a Class A network is approximately 17 million (actually, it’s 16,777,214 nodes). An example of a Class A network is the original Internet, which at first was called ARPANET and was managed by the Department of Defense.

Class B
Class B assignments are for medium-sized networks. This network address is 16 bits long and consumes the first two octet positions. Because two high-order bit positions are set to 1 and 0 respectively, there can only be 16,383 networks. But each network may have 65,534 nodes.

Class C
Class C allows for 2,097,151 networks and up to 254 nodes per network. Not surprisingly, this class is best suited for smaller networks. In fact, most of us work with Class C networks in the Internet. This class of license is by far the most popular address class. Three high-order bits have Class C addresses (set to 1-1-0 respectively).

Why only 254 nodes? Because 255 is a reserved number with TCP/IP addressing, typically employed for network- and internetwork-level broadcasting. Such a reservation is necessary for TCP/IP network management purposes.

Table 3-12 summarizes these network classes and nodes.
Default gateways

Default gateways are important, and provide a place for IP hosts to seek help when trying to communicate with distant IP hosts on another IP network. In essence, the default gateway address points or instructs the host to use another node to handle its routing needs. The goal of default gateways is to make IP routing efficient. In fact, individual hosts (say workstations) use default gateways extensively to avoid being burdened with address resolution.

On a larger network with hardware-based routers, this entry field will typically be the IP address of your router. On smaller networks using the routing capabilities of Windows 2000 Server, this would be the address of the server. In either case, the default gateway maintains current and detailed knowledge of network IDs of other networks on an internetwork (that is, a network of networks).

If you are new to TCP/IP and don’t know a lot about the network you are working with, just put the IP address of the Windows 2000 server in the default gateway field until you are able to better understand what the appropriate value for your network should be. (Of course this sage advice pertains to a smaller network; on a larger network, you should hire a Microsoft Certified Professional networking consultant to assist with your TCP/IP implementation.)

As shown in Figure 3-26, the default gateway is defined on the Internet Protocol (TCP/IP) Properties property sheet in Windows 2000 Server (under Network and Dial-up Connections in Control Panel, select Local Area Connection Properties).

On the Networking Essentials certification exam that you must pass to earn your MCSE designation, remember that the default gateway value is an optional entry in a TCP/IP scenario. The IP address and subnet mask are required.

This said, what are some situations where you might not enter a default gateway? One is with a small LAN where all nodes reside on one subnet. Before reading my second observation on not using a default gateway, take a moment to think about situations in your networking experience where you didn’t use a default gateway entry.

### Table 3-12 Summary of Class A, B, and C Networks and Nodes

<table>
<thead>
<tr>
<th>Class</th>
<th>Actual Number of Networks Possible</th>
<th>Actual Number of Nodes Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>126</td>
<td>16,777,214</td>
</tr>
<tr>
<td>B</td>
<td>16,384</td>
<td>65534</td>
</tr>
<tr>
<td>C</td>
<td>2,097,151</td>
<td>254</td>
</tr>
</tbody>
</table>

Chapter 3: Implementing TCP/IP
A second situation where you would be sans default gateway is with a network that, although it contains different segments, doesn’t use a router to route packets. Perhaps such a network uses a switch. Remember that switches reside at Layer 2 of the OSI model (the Data-Link Layer) and don’t require the kind of addressing information that packets do at OSI model Layer 3 (the Network Layer).

What are the mechanics of a default gateway and how is such a beast used on your Windows 2000 Server network? Suppose you have a network with a node called Host 100 and a node called Host 200. Note that Host 100 is located on Network 1 and Host 200 is located on Network 2. Assume that Host 100 addresses and sends a packet to Host 200. After Host 100 checks its local routing tables and is unable to resolve the path to Host 200, it forwards the packet to the default gateway.

A Windows 2000 computer builds the local routing table automatically, making one entry per route. This is obviously important for routing and resolution purposes. The local routing table can be displayed via the ROUTE PRINT command:

C:> route print
Active Routes:
The following sections describe the columns that are displayed above when we use the route print command. Such descriptions are necessary, as the output above, viewed alone, is both boring and somewhat difficult to interpret.

### Network address

The first column is where the server searches for the network address. Possibilities include 0.0.0.0, a value that refers to “this network” (the current network that originated the packet). At the other end of the scale, you have 255.255.255.255, a value that includes all networks; this means all the hosts on all the networks that can receive limited broadcast traffic. Typically, the 255.255.255.255 address is implemented when the default gateway proves unresponsive; it initiates a broadcast to all hosts on all networks.

### Netmask

The netmask is the second column. The entry found here represents the subnet mask that is associated with the network address.

A netmask of 255.255.255.255, which is all “ones” when displayed in binary format, tells us that this is the host entry; the network address and the IP address are exactly the same and so refer to one and the same machine (the local host).

### Gateway address

The gateway address is the default IP address used by the local host to resolve network address (the address of the IP router used to forward IP packets to other IP networks).

### Interface

The interface is the IP address of the local network adapter card on the machine. This network adapter card address will be used when forwarding IP packets to the network.

### Metric

This is the number of hops required to travel to the IP destination. A hop count is conceptually similar to the hop count discussion you might have encountered when mastering Messaging Transfer Agents (MTA) in Microsoft Exchange or Microsoft Mail.

<table>
<thead>
<tr>
<th>Network Address</th>
<th>Netmask</th>
<th>Gateway Address</th>
<th>Interface</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>131.076.120</td>
<td>131.107.6.120</td>
<td>1</td>
</tr>
<tr>
<td>127.0.0.0</td>
<td>255.0.0.0</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>1</td>
</tr>
<tr>
<td>131.107.6.0</td>
<td>255.255.255.0</td>
<td>131.107.6.120</td>
<td>131.107.6.120</td>
<td>1</td>
</tr>
<tr>
<td>131.107.6.120</td>
<td>255.255.255.255</td>
<td>127.0.0.1</td>
<td>127.0.0.1</td>
<td>1</td>
</tr>
<tr>
<td>131.107.255.255</td>
<td>255.255.255.255</td>
<td>131.107.6.120</td>
<td>131.107.6.120</td>
<td>1</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>224.0.0.0</td>
<td>131.107.6.120</td>
<td>131.107.6.120</td>
<td>1</td>
</tr>
<tr>
<td>255.255.255.255</td>
<td>255.255.255.255</td>
<td>131.107.6.120</td>
<td>131.107.6.120</td>
<td>1</td>
</tr>
</tbody>
</table>
You can interpret the local routing table according to the entries in the Network Address column of Table 3-13. Note the information displayed in this example reflects a Windows 2000 Server that has its own IP address as the default gateway. As discussed earlier, it is common to put the Windows 2000 Server's IP address in the default gateway column on LANs that have no subnets or routers.

<table>
<thead>
<tr>
<th>Network Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0.0.0</td>
<td>Default Gateway address line</td>
</tr>
<tr>
<td>127.0.0.0</td>
<td>Loopback address</td>
</tr>
<tr>
<td>131.107.6.0</td>
<td>Local network</td>
</tr>
<tr>
<td>131.107.6.120</td>
<td>Host IP address</td>
</tr>
<tr>
<td>131.107.255.255</td>
<td>Network-level broadcast</td>
</tr>
<tr>
<td>224.0.0.0</td>
<td>Multicast address</td>
</tr>
<tr>
<td>255.255.255.255</td>
<td>Internetwork-level broadcast</td>
</tr>
</tbody>
</table>

Table 3-13 Detailed “Route” Information

See the Gateway Address column (third column in the preceding list) for the correct default gateway address entry. In this case, that value is 131.107.6.120. Addresses are added to the local routing table via the ROUTE ADD command. This command once initiated IP-based WAN communications between two cities for a client when nothing else would. For example, imagine you have a situation in which you cannot successfully ping across a router. Now consider the following command:

C:>route add 204.107.3.0 mask 255.255.255.0 204.107.2.199 –p

So what’s going on here? Challenge yourself to answer that simple question by considering this router command in the light of Figure 3-27. Hopefully you, like me, enjoy an occasional analytical exercise to solve. If you can quickly see the main point (hint: make a persistent route), I again applaud your understanding of TCP/IP fundamentals and encourage you to strongly consider taking the Microsoft TCP/IP certification exam if you haven’t already done so.

The preceding route add command explicitly states that to get to the 204.107.3.0 subnet with a mask of 255.255.255.0, use gateway 204.107.2.199 and make this a static of persistent route by writing it to the Registry (-r).

This persistent route is written to `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControl\Set\Services\Tcpip\Parameters\PersistentRoutes`. Shown in Figure 3-28, this is significant because it enables the persistent static route to survive the expiration of the current computing session. That
is, when the Windows 2000 Server is rebooted, the persistent route is recreated on the basis of the Registry entries displayed in Figure 3-28. By analogy, think of a congressional bill that survives two different sessions of the U.S. Congress with the same number.

Figure 3-27: A network with routers
Multiple default gateways
Many people are confused about the use of multiple gateways in Windows 2000 Server. Multiple gateways are entered on the Advanced IP Addressing property sheet. Figure 3-29 shows where multiple gateway addresses are added. You may also elevate one gateway address over another by changing the metric value. The metric indicates the cost of using the default gateway. The lower the cost, the more desirable the path.

Note that a common misconception is that you add multiple gateway values to connect disparate networks. For instance, a corporate private WAN connected via Cisco routers could be connected to the Internet at some point via an ISDN router (and an entirely different subnet).
Figure 3-29: Preparing to add multiple gateway addresses

In Figure 3-30, notice the private WAN for this company is connected via Cisco routers. The Internet connection, via a 7×24 ISDN connection, is an entirely different network.

It would be a mistake to make the entry shown in the Gateways field in Figure 3-31 to account for both the Cisco and ISDN routers at the Corporate location. In the figure, notice that the Cisco router and ISDN router addresses have been entered as gateways on this Windows 2000 Server machine (located at Corporate in Figure 3-30). This is an incorrect way to try to connect two disparate networks that have multiple routers.
Figure 3-30: A private wide area network

The real purpose of the multiple gateway entries is not to account for the multiple routers at the corporate site in the preceding example, but to provide a backup route for WAN traffic routing should the “default” gateway fail or go down. Stated another way, only the default gateway is used. If it is down (by this, I mean it is physically unavailable and not just unable to resolve an address), the second gateway entry is employed to get the packet from Point A to Point B.
Here is a correct use of multiple gateway address. In this scenario, there is a backup WAN link between two sites via RAS. Should the regular WAN link, via the Cisco routers, fail, then we are rerouted and reconnected via the WAN connections. Note that this second gateway entry value would point to the dial-up adapter. As shown in Figure 3-32, the second gateway address relates to our second or backup WAN connection.

![Gateway Settings](image)

**Figure 3-31:** Router addresses on a Windows 2000 server

## Understanding IP Routing

We technical types often remember our college days with both glee over the meaty technical courses we eagerly excelled in, and depression over the required liberal arts courses we suffered through. One such unmemorable course for many may well have been Communications 101. In this course, you learned about something called an S/R model. The basic S/R model, showing communications based on a sender and a receiver, appears in Figure 3-33.
Likewise, the Communications 101 instructor probably told you that someday you would appreciate the lecture being delivered. Guess what?! The instructor was right. Not only is the S/R model the foundation of network communications, it is also the underlying theory behind routing. Think about it. Routing is nothing more then connecting two parties that want to communicate but need a helping hand because of some barrier,
such as separate subnets. Thus routing is used to get the data packets — in this case, IP datagrams — from sender to receiver. In an IP routing scenario, you are basically concerned with two addresses, the destination address and the origination (or sender’s) address.

Truth be told, routing is considered by many the primary function of IP. With IP routing, we’re stationed at Layer three (the NetworkLayer) of the OSI model. Basically, the IP-related services evaluate each packet by examining the destination host address. The address is compared against the local routing table and evaluated to determine if any forwarding is necessary. If the destination address can be found on the local subnet, the packet is not forwarded. If the opposite is true — that is, the destination address is located on a distant and (for our purposes) physically separate network, then IP services undertake the appropriate forwarding steps. The steps include having the first router (which has evaluated the packet destined for a distant subnet) forward the packet to the other router (which is associated with that distant subnet).

Many of you know (but a surprisingly huge number of Windows 2000 Server professionals do not) that routers only forward information from router to router.

Officially, according to Microsoft, IP routers “use two or more network connection interfaces to attach and interconnect between each physically separate IP network for which they are enabled to forward packets between.” And don’t forget we use IP routing (and routing in general) to filter as much as we do to forward. In fact, IP routers, by default, do not forward broadcast packets beyond their native subnets. With respect to hardware-based routing solutions, it’s best to think of these beasties as positive checkoff devices. You must elect to have certain packet types flow through the system. So out of the box, the average router is probably configured to do more filtering than forwarding.

Conversely, a negative checkoff is analogous to joining a mail-order CD-of-the-month club. That is, until you explicitly tell the company to stop sending you recordings, you receive monthly music in perpetuity. It’s a negative checkoff because you are forced to explicitly stop a certain activity, and this is more akin to lower OSI-level devices such as switches. Typically, a switch forwards everything until told differently (that is, if you have a “smart” switch that can be instructed to perform limited filtering).

Now let’s focus on common characteristics of IP routers. First, routers are multihomed hosts, meaning they have at least two network interfaces. The interfaces can be a Windows 2000 Server with two network adapter cards or a hardware-based router with an internal LAN network interface port and an external network adapter port. These interfaces provide the connection between the two separate networks, as shown in Figure 3-34.
Another function IP routers perform is forwarding packets for TCP/IP hosts. At a minimum, the IP router must be able to forward IP-based packets to other networks. And of course more often than not, it performs some level of filtering.

Finally, routers are both hardware based and out-of-the-box implementations, provided by such vendors as Cisco, Bay, and Hewlett-Packard. Software-based routing solutions included complex implementations of Windows 2000 Server’s Routing and Remote Access.

## Routing Tables

Routers depend on routing tables, which store addressing information about IP networks and hosts. Routing tables are called upon via a three-step approach (with an ultimate goal of getting the packet where it belongs):

1. The computer sends a packet. The computer then inserts its own source IP address and destination address into the IP header (see Figure 3-35).
2. The computer examines the destination address and compares it against the local routing table. Depending on what the computer finds in its local routing table, three possible course of action occur (see Figure 3-36).

3. This step determines how the packet is routed. There are three possible choices:
a. The packet can be forwarded up to a higher protocol layer in the OSI model if the destination address is the same as the machine performing the packet evaluation. This packet has already arrived at its destination (see Figure 3-37).

Figure 3-37: Step 3a: The packet can be forwarded up to a higher protocol layer.

b. The packet can be forwarded to another computer (or in the case of RAS, an attached dial-up adapter). See Figure 3-38.

Figure 3-38: Step 3b: The packet can be forwarded to another computer.

c. If appropriate, the packet can be killed then and there (see Figure 3-39). This is referred to as discarding.
In following these three steps, a packet on a routed network ultimately gets to where it belongs. This is in large part why many of us, as network professionals, are here: to make sure the packet gets from point A to point B. Do you remember the old saying about making the trains run on time?

A Word about Research

So we end this chapter on TCP/IP implementations. In the spirit of my next tip (research), I’ve placed a word about it at the end. That’s because far too many of us either conduct our research too late in a project or implementation to save us, or, as is more likely the case, ultimately chalk up an implementation to research.

How often have you pursued a possible solution only to discover later that more research would have prevented mistakes? A fledgling ISP shared an experience that came about from inadequate research. After it was too late, this ISP discovered what has been already well-documented in the trade press and other published resources: Windows 2000 Server provides adequate or better solutions for those seeking an application server, file server, and Internet server all in one; NetWare is a stronger file server than either an application or an Internet server; and, UNIX is about as friendly as a junkyard dog. The lesson this ISP learned was: don’t make your server into something that it is not. Only after investing many dollars did this ISP learn that Microsoft’s networking solutions including Windows 2000 Server met its needs better than the UNIX system it had already purchased and deployed.

A final thought regarding TCP/IP: A simple system is the most trouble free. Start with a simple TCP/IP system and work your way up to the more complex one. If you are new to TCP/IP, it’s critical that you start simple and progressively work your way into more complex TCP/IP implementations. If you break this rule, you will likely get in over your head.
Summary

The intense TCP/IP definitions in this chapter round out TCP/IP fundamentals that will assist you in working with TCP/IP and in reading the next several chapters devoted to TCP/IP. From here on out, you are ready to grow as both a Windows 2000 Server professional and TCP/IP practitioner. Godspeed to you as you start implementing TCP/IP-based networks using Windows 2000 Server.

This chapter covered the following:

- TCP/IP definitions
- Microsoft's implementation of TCP/IP
- A detailed analysis of TCP
- A detailed analysis of IP
- Windows Sockets — the definition and the application
- The three parts to Internet addressing: the IP address, the subnet mask, and the default gateway
- Windows 2000 Server basic routing