PART ONE

The Basics

As we begin our discussion of TCP/IP networking with Windows 2000 we first look at the basics of TCP/IP. We look at all those funny numbers and talk about IP addresses, subnet masks, name resolution, and all kinds of strange things.

After we look at the basics of how TCP/IP works, we examine the TCP/IP protocol stack and look at the utilities and functionality that are contained in Windows 2000.

ONE

Introduction to TCP/IP

In this chapter we begin with an overview of TCP/IP. We first look at what TCP/IP is, how it works on a Windows 2000 network, and how we configure client machines to communicate using the protocol. Then we look at how IP addresses work and conclude with a discussion of how we plan the deployment of a TCP/IP network.

Network models assist us in visualizing the way that computers talk to one another. We will use them in our efforts to first understand the protocols that make up the TCP/IP suite and the data flow, and we will revisit them as we examine network optimization and as we look at the troubleshooting scenarios. The DARPA model, the one that TCP/IP was based upon, maps well to the OSI model. For a more in-depth look at networking models, refer to Appendix H.

What Is TCP/IP?

TCP/IP provides a method, or a means, for computers and other devices on a network to be able to communicate with one another. Each of these devices is called a host. A host can be a computer, a server, a printer connected to a network, or even a managed hub, switch, or router. TCP/IP is actually a suite of protocols. A protocol is a set of agreed-upon commands and functions that allow two computers to talk with one another and to transfer information. For instance, within the TCP/IP protocol suite, there is a method that lets one computer know when another computer wishes to communicate. This function, or command, is the SYN command that essentially says, Let's synchro-

nize our numbers so we know which numbers we are going to start using and we will not lose information as we send it to each other. The proper response to a SYN command is a SYN with an ACK or an acknowledgement that the request has been received and understood. There are lots of other commands that are part of the protocols that make up the TCP/IP protocol suite, but these two should give you an example of the kinds of things that protocols allow computers to do.

When we refer to TCP/IP as a protocol suite, we mean there are many protocols that are commonly distributed together. Each of these protocols provides a certain function and serves specific purposes. In fact, there are two protocols mentioned in the name TCP/IP. These two protocols are TCP, which is the Transmission Control Protocol, and IP, which is the Internet Protocol. Even though IP is called the Internet Protocol, it does not mean that use of the protocol is limited to the Internet. In fact, there is also the internet which when used with a small letter "i" refers to an internal network. When the word Internet is used with a capital "I," it is understood to refer to the worldwide Internet. These protocols are the transmission control protocol and the internet protocol. There are however, other protocols usually contained in a full implementation of the TCP/IP protocol suite. A full-implementation TCP/IP will normally contain the following protocols:

- Transmission Control Protocol (TCP).
- Internet Protocol (IP).
- User Datagram Protocol (UDP).
- Address Resolution Protocol (ARP).
- Internet Control Message Protocol (ICMP).
- Internet Group Management Protocol (IGMP).

The six protocols listed above will normally be resident in any implementation of the TCP/IP protocol suite. This implementation is also sometimes referred to as a protocol stack. As we will see, each of the protocols provides certain functionality and a specific purpose in allowing computers to communicate with each other. So, how do these machines find each other on the network? Well, that falls into the area of name resolution. Let's look at that now.

Name Resolution

There are several ways that names get resolved. Let's say I am supposed to meet a new client at her office in a large office building downtown. Once I find the building (and a place to park) I still need to find my client. One way to find the client is to walk into the building, and yell "Hey, Sally! Where are you?" If Sally is within hearing distance of the yell she will quickly let me

know where she is, and I find her office number and go in for our meeting. This method of name resolution may work fairly well if it is a small office building and there are not too many other consultants in the lobby yelling for their clients. If there are too many people yelling in the lobby, then people will not be able to hear the yell, and there will not be a response. In addition, people in offices above the first floor will not be able to hear the yell and will not respond with their office number. So we need to find an additional way to resolve client names to office addresses.

Enter the office directory. Most office buildings nowadays have evolved from the old-fashioned yelling of consultants in the lobby and have implemented a directory of names of the tenants in the office building. Using this method of name resolution, the consultant enters the lobby of the office building, goes up to the desk in the middle of the lobby, and asks for the office number of Sally. The person at the desk looks in the name directory and informs our consultant that Sally is in office 823. This method of name resolution offers several advantages over the yelling method. For instance, it keeps the noise down in the lobby, people in the lobby are able to hear one another when they talk, and therefore they do not have to repeat themselves. By using a central desk with a list of all the tenants, the consultant can quickly and efficiently find the desk, ask the question, and go meet his or her client. This is done without yelling and waiting for a response.

Evaluation of Name Resolution Methods

The first method of resolving the tenant name to the office number (the yelling method) has several advantages. These advantages are listed below:

- Yelling is easy to implement. You simply inform all the visitors to the office building that in order to find a tenant you must yell. Once all the consultants are configured to yell to find the clients, nothing else is required. It is quick and easy to implement. There are no additional facilities required.
- Yelling is fast. When the visitor to the building yells, the tenant will respond with his or her office address. It works fine. There are no lines for the central desk, you do not have to ask anyone else, the tenant responds, and that is it.
- Yelling works well for a small office building. It is really the best solution for a small office building with a single floor and not too many tenants.
- Yelling works well for an office building that is not too busy. If there were a constant parade of consultants and other visitors in and out of the office building, then there would be too many people yelling at the same time and messages would get lost. For example, if there were hundreds of visitors in the lobby all yelling for the tenants, and there are hundreds of tenants all yelling out their office

addresses, it would be a constant barrage of "What was that? I can't hear you. Can you repeat that please, I did not get it all." This only aggravates the problem.

OK, so there are several advantages to the yelling method of name resolution; what are some of the disadvantages? Well, the disadvantages are listed below.

- Yelling does not work in a multifloor office building. In fact, most office buildings have multiple floors, and therefore the yelling method is not practical.
- Yelling is noisy. All the yelling back and forth creates a lot of noise. In busy office buildings, there will be a lot of repeating of the information, because the visitors to the building will not be able to hear the answers from the tenants.
- While it may seem like a cheap solution, it is actually rather expensive to have consultants standing around in the lobby yelling. It slows things down, and with consultants time is money. Therefore it is an inefficient utilization of resources.
- Yelling leads to congestion in the lobby. As the visitors are standing around waiting for the address of the tenant, they are blocking exits to the street, they are blocking entrances to the elevators, and they cause additional visitors to have to stand around and wait as well.

Clearly, there needs to be a better way to resolve names to addresses. There is—it is the directory at the central desk. Now when the visitor to the building needs to find the tenant, he or she goes to the central desk and asks for Sally's office number. In this method, the visitor needs to know where to find the central desk, and know that he is supposed to ask at the central desk instead of yelling to find the tenant's office. Once the visitors are informed of the location of the central desk and know they are supposed to ask at the desk for the address of the tenant, everything else will be automatic. There are several advantages to this method. These are listed below.

- For one thing it is quieter. There is a single conversation between the central desk and the visitor. "Where can I find Sally's office?" "Sally is in office 823." It can be just about as fast as yelling.
- One huge advantage of the central information desk is the desk can know about offices on other floors. One desk can have information about the entire building. In addition to this, if you own several office buildings, you can have the receptionists share information between buildings; they can even merge their list of names, but we will talk about that later. There would be no way to find tenants on other floors using the yelling method. By having a central desk, we can provide information about the entire building.
- It is a scalable solution. As we get more traffic in our office building, we have more visitors, trying to find more tenants; we can simply add

additional information desks. We can even increase the horsepower of each information desk by adding additional receptionists to each desk. So we have the option of adding extra receptionists, or extra information desks. With the yelling method, there is a finite limit to the number of visitors and conversations we can have. Using the information desk method, we can scale very well. We are limited only by the amount of physical space in our building.

Addressing Concepts

Just as each tenant in the office building analogy has an address, so too do devices on a TCP/IP network. In fact, each device that wishes to communicate with another device must have an address. This includes, but is not limited to, computers, servers, printers, routers, bridges, switches, network-attached storage devices, telephones, networked copy machines, and networked scanners. In fact, on the Internet, there are hot tubs, coffee machines, soda machines, and other devices as well. Anything that wants to communicate with another device needs a TCP/IP address. In fact, there are coffee pots that have a network interface card, an IP address, and an embedded Web server. You open up your Web browser, and can tell when the coffee was made, how much is left in the pot, and the temperature. Some versions even have information about what kind of coffee it is (decaf, or some strange flavored bean). This is a real time-saver in the office; instead of wasting time walking to the coffee pot to get a cup of nonexistent coffee, you simply click a hyperlink on your desktop, and voila! (Of course you know what the interface is programmed in, don't you? It uses Java.)

So each device must have a unique address; in TCP/IP language, the device is often referred to as a host. The IP address is sometimes referred to as a host address, and the process of translating a host name to an IP address is sometimes referred to as hostname resolution. So just as we have the office address, we also need to know what floor the office is on. For example, Sally's office is on the 8th floor. We know that because we understand the numbering scheme. The first group of numbers refers to the floor, and the second group of numbers refers to the specific office on the specific floor. In order to visit Sally, we must go to the 8th floor and find office number 23.

Each floor in our office building can be thought of as a different network. So in order to know where to find Sally, we need to first know how to interpret her address—which part of the address is the floor, and which part of the address is the office number. For instance, Sally, with an address of 823, could be on floor 82 with an office number of 3. So in order to know how to separate the floor number from the office number, we need additional information.

In IP addressing, the additional information comes in the form of what is called a subnet mask. These are the numbers that often look like 255.255.255.0. We will look at subnet masks later.

The next thing we need to know once we have broken the office address into floors and doors is how to get to the office. This is probably via the elevator bank. Often there is a single bank of elevators that goes to all other offices in the building. This would be a default gateway in TCP/IP language. A default gateway is where a piece of information will go if it is leaving the network to go to another network. It is the default path it will take. In our example of the office building, we could have the potential of two default paths—the door going outside and the doors to the elevators. So how do we know which way to go? It depends on our destination and information we receive when we get ready to go there. For instance, we could provide our visitors with information to let them know how to get to Sally's office: Use the elevator bank on the north side of the building. It becomes a static route to her office. If Sally moves to another floor, then we will need to configure a new route to the office. In addition, all the other visitors to Sally's office will need this information as well. When she moves her office to the 24th floor and we need to use the elevator bank at the south side of the building, then we will need to tell all the visitors to Sally's office that there is a new route to her office. This is in fact the problem with static routes; they have to be manually configured. In addition, if we miss telling visitors how to get to the new office, they will get stuck riding the elevator, going to the 8th floor, looking around, not finding Sally, going back to the floor, going back and just staying in the elevator until they retire! Obviously that is a situation we need to track! We will talk about that later.

So, in order to set our default path in our office building, we need to decide, Where does all the traffic go? Is most of the traffic using the elevator bank on the north side of the building, is it on the south side of the building, or is it at the front door? So looking at traffic patterns is a key factor in selecting our default gateway.

In TCP/IP language, the default gateway is where a packet will go by default if the destination is not something on the local network. In our office building analogy, if a visitor to the building came in looking for address 112, he would be able to know that it was on the first floor, office 12. He would not need to go to the elevator bank, as he would already be on the floor. If upon coming out of office 12, he wanted to go see Sally, at 823, then he would go to the elevator bank if that were the default gateway.

So if my computer attempts to communicate with a server on the same network, then it will not need to use the default gateway. If however, it needs to go somewhere else, it will use the default gateway. So in this example, local traffic goes out on the wire and everything else goes out the default gateway. If, however, we want to use a different router to go somewhere else, then we may configure a static route to get to the other address. We will

go into this in a lot more detail in the chapter on routing. As we can already see, however, routing and IP addressing go hand in hand.

Understanding IP Addressing

TCP/IP addresses are essentially numbers; these numbers are expressed in groups of four. Each device that resides on a TCP/IP network must have a unique address. So how do we get these addresses, and how do we make sense of these numbers? Well, these numbers are often referred to as octets, because we can fit eight binary numbers into the space separated by the four periods. These numbers can range from 0 to 255. For instance, zero would be 00000000 in binary for the octet, and 255 would be 11111111. For readability, I will sometimes put a space in between each four binary numbers to make the octet easier to read in binary. Each octet is separated by a period. Therefore, with four octets, we have a 32-bit address. Rather than having to deal with the 32-bit number, it is easier to separate it into four eight-bit numbers. However, at the heart of the IP address are binary numbers. Binary numbers count in powers of two. As we can see in Table 1.1, each number doubles as

Table 1.1 *Binary numbers and their decimal equivalents.*

Binary	Decimal
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

the binary 1 moves over a spot. For instance, binary 1 is equal to 1 in decimal. But 10 in binary is equal to 2. Notice that when we add a 0 to the end of the binary number we double the value. So for instance, 100 in binary is equal to 4, and 1000 in binary is equal to the decimal value of 8. We can use this little trick to help us to memorize the binary and decimal equivalents. In order to work well with IP addressing, we will need to have a firm grasp of binary numbers. The chart in Table 1.1 lists binary numbers up to 15 decimal.

Out of these four octets, we need to find both the network id and the host id. The network id is also at times referred to as the network address, and the host id is also referred to as the host address. For instance, if we have two networks separated by a router and we have hosts connected to those two networks, then we will need to know how much of the four octets are considered as network id and how much are considered as host id. Each network id must be unique within the entire internetwork, and each host id must be unique within each network.

The address class provides one way to tell the network id. There are five address classes defined by the Internet community. These classes can accommodate small networks as well as very large networks. By looking at the IP address, it is possible to tell what class of address you are examining, and therefore, you would know the network id and the host id. Let us now look at the various classes of addresses.

Class A Addresses

Class A addresses are great for very big networks that have lots of hosts. A class A address uses 8 bits for network addresses and 24 bits for host ids. The first bit in the first octet of a class A address is always set to 0. Therefore, the largest number we could get out of 7 bits is 127. Let's look at this in Table 1.2 below.

Table 1.2 *Class A addresses have a 0 in the first position of the octet.*

Binary Number	Decimal Number
0111 1111	127

Let's take a look at how we get the 127 in Table 1.3 below.

Table 1.3 *Class A address exposed.*

0	1	1	1	1	1	1	1
128	64	32	16	8	4	2	1

Since there is a 0 in the 128 position, it is not added. However, we do have 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127. The first bit in a class A address will always

be 0. This gives class A addresses a potential range of 1 to 127 seen in Table 1.4 and Table 1.5. However, 127 cannot be used for a network address as it is reserved for diagnostic purposes illustrated in Table 1.6. For instance, 127.0.0.1 is the traditional loopback address used in TCP/IP diagnostics. We therefore have a total of 126 network addresses available using class A addressing. However, we have a huge amount of potential hosts as we have 24 bits left from our 32-bit number to use for hosts. The total number of hosts available is 2^24 – 2 which is equal to 16,777,214 hosts. The reason we have to subtract 2 is because the 0.0.0 is used to refer to the network address, and the 255.255.255 is used for the broadcast address in a class A network. If you look at them in binary, then it is all 00000000.00000000.00000000 for the 24 bits and all 11111111.11111111.11111111 for the 24 bits, respectively.

Table 1.4 *The range of network addresses in class A.*

Binary	Decimal
0000 0001	1
0111 1110	126

Note: 127 is not used in class A addresses. 127 would be 0111 1111.

Table 1.5 *A typical class A address.*

Network	Host	Host	Host
0000 1010	0000 0000	0000 0000	0000 1111
10	0	0	15

Note: This address would appear as 10.0.0.15. The 10. is the network, and the .0.0.15 is the host address on the 10. network.

Table 1.6 *The loopback address.*

Network	Host	Host	Host
0111 1111	0000 0000	0000 0000	0000 0001
127	0	0	1

Class B Addresses

The class B address is used for medium-sized networks. The first two octets are used for networks, and the last two octets are used for host addresses. This will give you 16,384 network addresses and 65,534 host addresses. Let's take a look at how these numbers come about. Since the first bit is always set at 1 and the second bit is always set at 0, that gives you 14 bits to use for network addresses. We can then take 2^14 (2 raised to the 14th power) and we

will get 16,384 network addresses. Remember, the reason we can take 2 and raise it to the various powers is because we are dealing with binary numbers, which are base 2. The only values allowed in binary numbers are 0 and 1. Therefore, it is a base 2 number system. With the class B address, the first bit is always set to 1 and the second bit is always set to 0.

In order to find the allowable range of hosts in a class B network, we take the remaining 16 bits. (IP addresses are 32-bit numbers, and we used 16 bits for the network address; therefore, it leaves 16 bits for the host addresses.) We can take 2^16, which will give us 65,536, and we have to subtract 2 from the number, which will give us 65,534 host addresses available on a class B network. Remember, the reason we subtract 2 from our number is because all 0's are used for the network address, and all 1's are used for the broadcast address. So we see in Table 1.7 a typical class B address with both the network address and the broadcast address exposed. The range of addresses is seen in Table 1.8, with a sample address illustrated in Table 1.9.

Table 1.7 *Typical class B addresses using the network and the broadcast addresses.*

Binary	Decimal	Description
1000 0000 . 0000 0000 . 0000 0000 . 0000 0000	128.0.0.0	Network address
1000 0000 . 0000 0000 . 1111 1111 . 1111 1111	128.0.255.255	Broadcast address for the 128.0 network
1000 0000 . 0000 0000 . 0000 0000 . 0000 0001	128.0.0.1	First valid host address on the 128.0 network.
1000 0000 . 0000 0000 . 1111 1111 . 1111 1110	128.0.255.254	Last valid host address on the 128.0 network.

Table 1.8 *The range of addresses for class B networks.*

Binary	Decimal
1000 0000 . 0000 0000	128.0
1011 1111 . 1111 1111	191.255

Note: The first network address in the class B address range is 128.0 and the last class B network address is 191.255.

Table 1.9 *The class B address exposed.*

Network	Network	Host	Host
1011 1111	0000 0000	0000 0000	0000 0011
191	0	0	3

Class C Addresses

The class C address is suitable only for small networks. The range of networks in the class C range is very large as it spans three octets. However, we are limited to one octet for host addresses. As we see in Table 1.10, the first three bits in a class C address are always 1 1 0. When we do this, we are left with 21 bits (or places for numbers either 1 or 0). If we go back to our binary (base 2) math, then we have the number of potential network addresses equal to 2^21 which gives us 2,097,152 network addresses.

Table 1.10 *The range of class C network addresses is rather large.*

Binary	Decimal
1100 0000 . 0000 0000 . 0000 0000	192.0.0
1101 1111 . 1111 1111 . 1111 1111	223.255.255

Now let's take at look at the range of class C hosts we can have on our class C network as seen in Table 1.11. We have only 8 bits left from our 32-bit number. Therefore 2^8 is equal to 256, but we must subtract 2 from it, which will give us a total of 254 hosts available on each of the 2,097,152 network addresses. A sample address is seen in Table 1.12.

Table 1.11 *Typical class C addresses using the network and the broadcast addresses.*

Binary	Decimal	Description
1100 0000 . 0000 0000 . 0000 0000 . 0000 0000	192.0.0.0	Network address.
1100 0000 . 0000 0000 . 0000 0000. 1111 1111	192.0.0.255	Broadcast address for the 192.0.0 network.
1100 0000 . 0000 0000 . 0000 0000 . 0000 0001	192.0.0.1	First valid host address on the 192.0.0 network.
1100 0000 . 0000 0000 . 0000 0000 . 1111 1110	192.0.0.254	Last valid host address on the 192.0.0 network.

Table 1.12 *The class C address exposed.*

Network	Network	Network	Host
1101 1111	0000 0000	0000 0000	0000 0011
223	0	0	3

Note: The network address here is in the class C address range, it is 223.0.0 and the host address is 3.

Class D Addresses

Class D addresses are reserved for IP multicasting. The first four bits are always set to 1110. The remaining bits are used as destination addresses that computers, which subscribe to a particular multicast event, will recognize. If you add up the 1110 0000, we have 128 + 64 + 32 + 0, which gives 224. Quite often multicast addresses will be in the 224 range. Windows 2000 DHCP service is capable of handing out multicast addresses to applications.

Class E Addresses

Class E addresses are experimental and are reserved for future use. The first four bits are always set to 1111, which is 128 + 64 + 32 + 16, which equals 240. If you see an IP address that starts in the 240 range, know that it is an experimental address.

IP Addressing Summary

OK, so by and large then, we have to be concerned with class A, B, and C addresses on our network. We need to be aware of class D addresses as they are used for multicasting (by the way, if you are doing network monitoring traces, you will see WINS using multicasting to identify itself to other WINS servers). So here is a chart (Table 1.13) that summarizes what we have talked about so far with IP addresses.

Table 1.13 *IP addressing summary.*

Class Id	Network Id Range	Host Id Range	Number Networks	Number Hosts
A	1.– 126.	0.0.1– 255.255.254	126	16,777,214
B	128.0 – 191.255	0.1– 255.254	16,384	65,534
C	192.0.0 – 223.255.255	1– 254	2,097,152	254

As we have seen, the network id is used to provide an identifier for computers, printers, and other devices to communicate with one another. It can be thought of as a physical network, and often the IP network will in fact mirror a physical network. If we go back to our example of the office building earlier in the chapter, we can think of each floor in the office building as a separate IP network. In fact, if we used class A addressing, we could use 1. for the first floor, 2. for the second floor, and as long as our building was not any taller than 126 floors, we would have it made. Of course, the more alert

reader will immediately recognize there are some serious concerns in our IP addressing scheme for the office building, and we will address (no pun intended; well OK, maybe it was intended) those issues in just a few minutes.

Now, if we are going to route IP traffic from one network to another, then the networks must in fact have different network addresses. For instance, we cannot route from a network with a network id of 1 to another network with a network id of 1. As we see in Figure 1.1, the traffic will not be able to route between the two floors of the office building. This, of course, does not completely keep us from communicating between the two floors; we can bridge the two networks, or we could simply plug all the devices into the same hub. We would have a single network that resided on two floors of the office building. So, as we see, IP networks do not always have to follow physical typology; however, often it does make sense to do so.

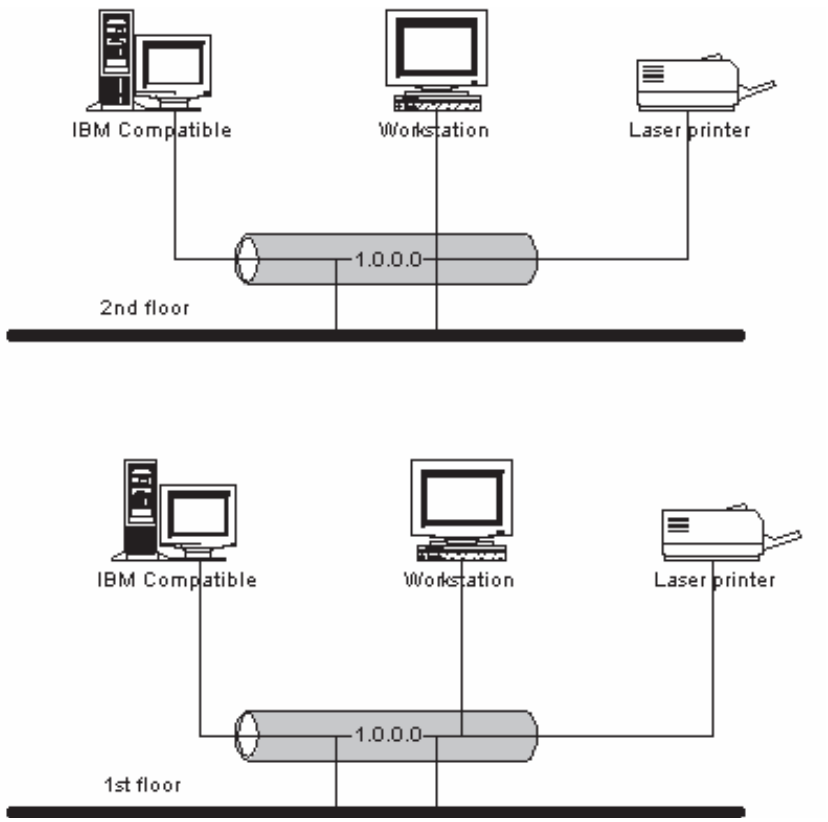

Figure 1.1 *If the network id's are the same, there is no way to route between the two floors in the office building.*

IP Addressing Guidelines

As we wind up this section on IP addressing, let's summarize some addressing points to remember. Following these points will keep your network running smoothly and help to avoid problems.

- Each network id must be unique.
- The network id cannot be 127. This is reserved for diagnostic functions.
- The network id cannot be set to all 0's. This is reserved for the local network and is not routed.
- The network id cannot be set to all 1's. This is reserved for network broadcast.
- Host id's must be unique within the subnetwork.
- Host id's cannot be set to all 0's. This is reserved for local network id.
- Host id's cannot be set to all 1's. This is reserved for network broadcast.

Subnets

Well, so far, we have been talking about network id's as a whole, or as a single entity. We want to now expand our horizons and look at a cool trick we can do with the subnet mask. If all we did was accept the default subnet mask, life would be rather boring, because as we have already seen, we can look at the first three bits of the first octet and figure out how much of the address is network and how much is host id.

If we go back to our friend the class A address, we have a scheme that will produce 126 network addresses each with over 16 million hosts. Now that is a lot of hosts, but it is not very many network id's at all. Because of the way the subnet mask works, we can actually move things around a bit and make some more room for host addresses. As we can see in Table 1.14, we have a typical class A address, with a default subnet mask. In this instance, the IP address is 10.0.0.3, and the subnet mask is 255.255.255.0. As we look at the address and the subnet mask, a pattern should begin to emerge—everywhere the network address is in the first line, it is covered up by 1's in the second line. Look again at Table 1.14. The network address for a class A address in this example is 10.0.0 and under each of the three octets, there is a whole bunch of 1's (8 to be exact). Under the host id portion, there is a whole bunch of 0's (again, 8 to be exact). This is a pattern: Network id's are separated by 1's in the subnet mask, and host id's are separated by 0's in the subnet mask. This was by design; it is not a bug, nor a hidden feature, and it is in fact the way that IP is able to separate the host id from the network id.

Table 1.14 *A look at class A addressing.*

Network	Host	Host	Host
0000 1010	0000 0000	0000 0000	0000 0011
1111 1111	0000 0000	0000 0000	0000 0000

IP uses a method that is actually rather common in binary math that is called "ANDING" (and the spell checker went wild on that one). In some respects, it is kind of the way that Jethro Bodeen does ciphering. It goes like this: 1 "AND" 1 equals 1. 1 "AND" 0 equals 0. 0 "AND" 0 equals 0. That's it. That is all there is to "ANDING." So in summary, 1 "AND" 1 equals 1; everything else, every other combination of numbers, equals 0.

Well, that is really cool. You have now read nearly 17 pages of this book, and you have learned to cipher like Jethro Bodeen. So what good is that? Ah, it is cool! Let us now look in Table 1.15 at our class A address we were working with earlier, and then "AND" the subnet mask and see what we get. (I decided to put *and* in capitals and quotation marks when I am talking about the process of "ANDING" for readability purposes).

Table 1.15 *A look at a class A address, "ANDED" with the subnet mask.*

Network	Host	Host	Host
0000 1010	0000 0000	0000 0000	0000 0011
1111 1111	0000 0000	0000 0000	0000 0000
0000 1010	0000 0000	0000 0000	0000 0000

After we "AND" the subnet mask with the IP address, we get 0000 1010, which is equal to 10 in decimal. With the next two octets, we have 0000 0000 and 0000 0000, which gives us 10 and that is the network id. Now let's take a look at another example. In Table 1.16, we are going to steal some of the bits commonly used for host addresses, and we are going to use some of them for network id's. This is called creating a custom subnet mask.

Table 1.16 *Addional network id's can be obtained by stealing them from hosts.*

Network	Network	Host	Host
0000 1010	0000 0000	0000 0000	0000 0011
1111 1111	1111 1111	0000 0000	0000 0000
0000 1010	0000 0000	0000 0000	0000 0000

OK, what have we done now? Well, we used to have 24 bits for host id's and we used to have 8 bits for network id. Now we have 16 bits for host

id, and 16 bits for network id. In this example, we are treating our class A address as if it were a class B.

So what is the value of this? Let's go back to our office building example. Let's suppose that we have the 10-range class A address for our use (in fact we do, but we will talk about that later). Now remember, if we want to be able to isolate broadcast traffic and to reduce collisions on our network, we are going to have to segment the network. We can do this rather easily by simply making each floor of the office building a different network (or subnetwork). Now, is it really effective to use an address scheme that has 16 million addresses for each floor of the office building? In addition, since we only have one network id, the 10 range, we need to be able to get a bunch of networks out of that address space. In order to do this, we will steal space from the host side and turn it into network space. Let's see how that works in our example now. On the top row of Table 1.17 we have the IP address, the second line is the subnet mask, and the third line is the network id revealed by using "ANDING."

Table 1.17 *By using a custom subnet mask we gain network ids.*

Network	Network	Host	Host
0000 1010	0000 0001	0000 0000	0000 0000
1111 1111	1111 1111	0000 0000	0000 0000
0000 1010	0000 0001	0000 0000	0000 0000

We now have the ability to make multiple networks inside the 10 range. For instance, in our example above, we have 10.1 as a network address, and we now are using only 16 bits for host id's as opposed to the original 24 bits that were set aside in the default subnet mask. With 16 bits we are "limited" to 65,534 host addresses. In our example then, 10.1.0.0 is reserved as the network id, and 10.1.255.255 is the broadcast address for the 10.1.0.0 network. Using this combination of bits, we are now able to subnet our building into 255 different networks, which will handle just about any office building in the world. In addition to having 255 networks, each of those networks can handle 65,534 hosts. Remember that each device on the network has to have at least one IP address. That is, each server, printer, router, managed hub, or switch, must have a unique IP address. Even on small networks with 50 users, I have seen them take up over 100 IP addresses. We will talk about that more in the planning section.

What if we work for a company that has more than 254 remote locations? We have to move the subnet mask line of 1's over into the next octet. In order to figure out how far over to bring the subnet mask, we need to know how many networks we need, and we need to know how many hosts we need. Notice, there is a balance between hosts and networks: more net-

works, fewer hosts. If you have more hosts, then you can have fewer networks. You can almost think of the line of 1's in the subnet mask as a slider control in a standard windows GUI (graphical user interface).

Let's suppose that 254 networks is not enough for us to use, and we in fact do not have 65,534 hosts. If we move the subnet mask over two more notches, then we will be using 18 bits on the network side and 14 bits on the host side. Table 1.18 illustrates this solution. Using this solution, we can have 16,382 hosts on each network. The way we get that number is 2^14, which equals 16,384, and then we subtract two addresses, one for the network id address of all 0's and one for the network broadcast address of all 1's. We also have a lot of network id's as well. Since we are stealing 10 bits from the host address space, we have 2^10, which is 1024, and then we subtract two addresses (one is all 0's and the other is all 1's). That gives us a total of 1022 network id's and 16,382 host id's using a custom subnet mask of 255.255.192.0 with our 10-range address. Now you might be tempted to say that actually we are using 18 bits for the network address, and that therefore we should have 2^18, which would give us 262,144 network addresses. But remember that the first octet is essentially already spoken for by the class A address space. All we are doing is making use of our class A host address in a little more efficient manner. So when calculating the number of networks you can make out of a single IP address, you need to take into account the class of the address, and how many bits you are stealing from the host side of things. This is illustrated in Table 1.18.

Table 1.18 *Custom subnet mask.*

Network	Network	Custom	Host
0000 1010	0000 0001	0100 0000	0000 0000
1111 1111	1111 1111	1100 0000	0000 0000
0000 1010	0000 0001	1100 0000	0000 0000

Now, we need to look at the allowable addresses we have in our current configuration with our 255.255.192 IP address with our 10.1.64.0 network. Well, it breaks out like this:

- The network address is 10.1.64.0.
- The broadcast address is 10.1.127.255.
- The subnet mask is 255.255.192.0.
- The range of host addresses allowed on the network is 10.1.64.1 – 10.1.127.254.

What range of addresses do we have available to us for network addresses with our 10 network and our 255.255.192 subnetmask? We count network addresses in the following manner:

- The first allowable network id is 10.0.64.0.
- The next one is 10.0.128.0.
- 10.0.192.0.
- 10.1.64.0.
- The last one is 10.255.128.0.

Each of the above is a valid network id, and each can handle a range of IP host addresses. It is important to look at the reason the third octet only had network addresses with 64, 128, or 192. The reason is the way the subnet mask uses the ones and the zeros. For instance, we can have an IP address like 10.5.255.0

Subnetting Tasks

If we decide we need to use a custom subnet mask, there are essentially three tasks we have to perform. These tasks are listed below.

1. Determine how many networks are required.
2. Determine how many host id's are required on each network.
3. Determine the custom subnet mask to use in order to meet the objectives from steps one and two.

Let's suppose we have a small business with two locations. At each of the locations there are 50 hosts on the network. This includes servers, printers, managed network devices, and routers. The two locations are connected with a T1 line as illustrated in Figure 1.2.

In this example, we know there are 50 hosts and 2 networks required. The next question to ask is, will this company expand to more than two locations? For this example, they will not. They have been in the two locations for 50 years and have no plans to move or expand their business to a third location. This is actually a very important question to ask, as it is a major pain to create a new address space for a company, and it is really a major pain when it is spread out to multiple locations. So make sure you carefully consider all the growth plans for the company prior to creating your addressing scheme.

We are going to use the private address space in the 192.168.1.0 range. This is a class C address that is reserved from public usage and therefore we will not run into any problems with the "real world" live Internet addresses. Let's follow the procedure listed below to find our custom subnet mask.

1. Find out how many networks are required. We need two networks.
2. Convert the number of networks into a binary format. In this instance two looks like 0000 0010 in binary.

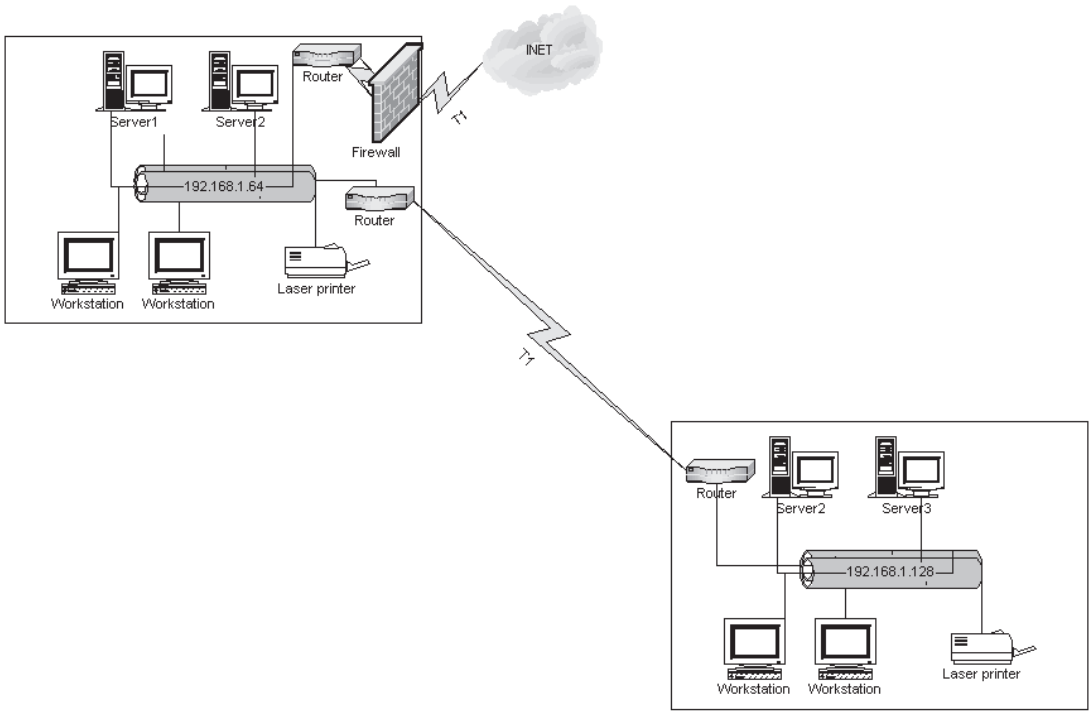

Figure 1.2 *Small business with two networks and a custom subnet mask.*

3. Find out how many bits it takes to convert the number of networks into binary. Notice that we really need only two bits to convert decimal base ten two into binary. We use 10, that is two bits, since each space occupies one bit.
4. Take the bits from the right side of the octet and move them to the left side of the octet. These are sometimes referred to as low-order bits and high-order bits, respectively. Our binary number now looks like 1100 0000. The reason we use 1100 0000 instead of 1000 0000 is that we are only trying to find out how many bits we need to create the custom subnet mask. We are not actually interested in the binary value of the number of subnets we need; we only want to know how many spaces it will occupy.
5. Convert the binary number from step 4 into a decimal format. The first 1 in 1100 0000 is equal to 128. The second 1 in 1100 0000 is equal to 64. Therefore 128 + 64 = 192.

Since we began with a network address of 192.168.1.0 and a default subnet mask of 255.255.255.0, we know we had one network and 254 hosts. We

know that we are moving into the fourth octet and stealing hosts away from the hosts position and therefore we will have fewer hosts available to us. The advantage, however, is we can use our single network, and create the two networks we need for our small business. Our new subnet mask is therefore going to be 255.255.255.192. This will give us our two networks we need.

It may be obvious to the alert reader that 192.168.1.0 is no longer our network address. So our next step is to find out what networks we can use with our custom subnet mask. Table 1.19 will illustrate how we can find our networks.

Table 1.19 *Using the bits occupied by a custom subnet mask to find valid network addresses.*

Binary	Decimal
0000 0000	0
0100 0000	64
1000 0000	128
1100 0000	192

Since we are only using two bits from the host address space, we can only come up with four combinations of the first two bits, as illustrated in Table 1.19. Now, what do we remember about all 0's in a network address space? That's right, all 0's means local, and all 1's is broadcast, so we cannot use those two combinations of numbers for a network. Therefore there are only two networks that we can make using the subnet mask of 255.255.255.192. The first network is 192.168.1.64. with a subnet mask of 255.255.255.192, and the second network address is 192.168.1.128 with a subnet mask of 255.255.255.192. Table 1.20 illustrates the two networks, available hosts, network id's, and broadcast addresses.

Table 1.20 *Available hosts using the 255.255.255.192 subnet mask.*

Network Id	Range of Hosts	Network Broadcast address
192.168.1.64	192.168.1.65 – 192.168.1.126	192.168.1.127
192.168.1.128	192.168.1.129 – 192.168.1.190	192.168.1.191

Let's see how all this shakes out when we "AND" these numbers together. This is illustrated in Table 1.21. In Table 1.22 we have the network address, the subnet mask, and the value that shows through the "ANDING" process. Remember, 1 and 1 is 1; every other combination of bits is 0. We see our custom subnet mask is a line of 1's that is 26 bits long. That leaves 6 bits available for hosts. We can also see that 192.168.1.0 is invalid because it re-

sults in a combination that is all 0's. We can also see that 192.168.1.192 is invalid because it results in a combination that is all 1's. By process of elimination, then, we only have two valid network addresses by using the 26-bit mask—that is, 192.168.1.64 and 192.168.1.128.

Mr. Ed Tip

Alternate subnet mask expressions

Use the slash method to express subnet masks.

For instance, 255.255.255.192 takes up 26 bits. The first network address can be expressed as 192.168.1.64/26. The second network address would be expressed as 192.168.1.128/64.

There is a short method we can use to find out how our numbering scheme will create network addresses. For instance, we know our subnet mask is 255.255.255.192. Let's see how we can use a shortcut method to determine our network id's.

1. Take the custom subnet mask and convert it to binary. You do not need to do all the 255's, only the last octet of the custom mask. In our instance, it is 192. It converts to binary as 1100 0000.
2. Now we look at the smallest binary bit used by the subnet mask 1100 0000. In this instance, the first bit is 128, and the smallest bit that is in use is 64 (the second high order position).
3. The lowest binary bit in use is the network id. In our example, it is 64. Therefore, the network id is 192.168.1.64.
4. Each additional valid network will be a multiple of the lowest binary bit in use. For our example they will be 192.168.1.64 and 192.168.1.128. As we see in Table 1.22 we cannot use 192.168.1.0 because it is all 0's; also we cannot use 192.168.1.192 as it is all 1's.

Suppose we had a different subnet mask that was 255.255.255.224. How would we find our valid network id's for that subnet mask? We would use the same steps outlined above.

1. Convert to binary, 1110 0000. That is, 128 + 64 + 32 = 224.
2. Take the smallest binary bit used by the subnet mask. In this example it is 1110 0000, so the smallest bit is 32.
3. Each additional network will be a multiple of 32. So we will have 32, 64, 96, 128, 160, and 192 for a total of 6 networks.

What if we had a subnet mask of 255.255.255.240? In this instance we are stealing four bits from the host address space in our class C address. Let us look at how we would figure this out.

1. Convert the number to binary, 1111 0000. We get this by adding 128 + 64 + 32 + 16. This is equal to 230.
2. We take the smallest binary number position that is occupied by the custom subnet mask; in this instance it is the 16 position.
3. Each network address will be a multiple of 16. So we will have 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, and 224.

Let's do one more example using the subnet mask of 255.255.255.248. In this instance, we are stealing five bits from the host address space. We do the calculation as listed below.

1. Convert 248 to binary, 1111 1000. That is, 128 + 64 + 32 + 16 + 8 = 248.
2. We take the smallest binary number, 8.
3. Each network address will be a multiple of 8.

Table 1.21 *Valid network and host addresses with a 255.255.255.248 subnet mask.*

Network Number	Network Id	Valid Host Range	Network Broadcast
1	192.168.1.8	192.168.1.9 – 192.168.1.14	192.168.1.15
2	192.168.1.16	192.168.1.17 – 192.168.1.22	192.168.1.23
3	192.168.1.24	192.168.1.25 – 192.168.1.30	192.168.1.31
4	192.168.1.32	192.168.1.33 – 192.168.1.38	192.168.1.39
5	192.168.1.40	192.168.1.41 – 192.168.1.46	192.168.1.47
6	192.168.1.48	192.168.1.49 – 192.168.1.54	192.168.1.55
7	192.168.1.56	192.168.1.57 – 192.168.1.62	192.168.1.63
8	192.168.1.64	192.168.1.65 – 192.168.1.70	192.168.1.71
9	192.168.1.72	192.168.1.73 – 192.168.1.78	192.168.1.79
10	192.168.1.80	192.168.1.81 – 192.168.1.86	192.168.1.87
11	192.168.1.88	192.168.1.89 – 192.168.1.94	192.168.1.95
12	192.168.1.96	192.168.1.97 – 192.168.1.102	192.168.1.103
13	192.168.1.104	192.168.1.105 – 192.168.1.110	192.168.1.111
14	192.168.1.112	192.168.1.113 – 192.168.1.118	192.168.1.119
15	192.168.1.120	192.168.1.121 – 192.168.1.126	192.168.1.127
16	192.168.1.128	192.168.1.129 – 192.168.1.134	192.168.1.135
17	192.168.1.136	192.168.1.137 – 192.168.1.142	192.168.1.143
18	192.168.1.144	192.168.1.145 – 192.168.1.150	192.168.1.151
19	192.168.1.152	192.168.1.153 – 192.168.1.158	192.168.1.159

continued

Table 1.21 *Continued.*

Network Number	Network Id	Valid Host Range	Network Broadcast
20	192.168.1.160	192.168.1.161 – 192.168.1.166	192.168.1.167
21	192.168.1.168	192.168.1.169 – 192.168.1.174	192.168.1.175
22	192.168.1.176	192.168.1.177 – 192.168.1.182	192.168.1.183
23	192.168.1.184	192.168.1.185 – 192.168.1.190	192.168.1.191
24	192.168.1.192	192.168.1.193 – 192.168.1.198	192.168.1.199
25	192.168.1.200	192.168.1.201 – 192.168.1.206	192.168.1.207
26	192.168.1.208	192.168.1.209 – 192.168.1.214	192.168.1.215
27	192.168.1.216	192.168.1.217 – 192.168.1.222	192.168.1.223
28	192.168.1.224	192.168.1.225 – 192.168.1.230	192.168.1.231
29	192.168.1.232	192.168.1.233 – 192.168.1.238	192.168.1.239
30	192.168.1.240	192.168.1.241 – 192.168.1.246	192.168.1.247

Table 1.22 *"ANDING" the network with the custom subnet mask.*

Network	Network	Network	Custom
1100 0000	1010 1000	0000 0001	0000 0000
1111 1111	1111 1111	1111 1111	1100 0000
1100 0000	1010 1000	0000 0001	0000 0000
192	168	1	0
1100 0000	1010 1000	0000 0001	0100 0000
1111 1111	1111 1111	1111 1111	1100 0000
1100 0000	1010 1000	0000 0001	0100 0000
192	168	1	64
1100 0000	1010 1000	0000 0001	1000 0000
1111 1111	1111 1111	1111 1111	1100 0000
1100 0000	1010 1000	0000 0001	1000 0000
192	168	1	128
1100 0000	1010 1000	0000 0001	1100 0000
1111 1111	1111 1111	1111 1111	1100 0000
1100 0000	1010 1000	0000 0001	1100 0000
192	168	1	192

Once we have determined our subnet id, we need to identify all the valid network id's and hosts that will be allowed on our network. If we go back to our example using the 192.168.1.32 with the subnet mask of 255.255.255.224, we already determined that we have 6 valid networks. They will look like Table 1.23. It is important to be able to draw this out, as it helps you to quickly see where the valid networks and host address combinations are located.

Table 1.23 *Valid networks and hosts for 192.168.1.x/27.*

Subnet Id	Range of Hosts	Subnet Broadcast
192.168.1.32	192.168.1.33 – 192.168.1.62	192.168.1.63
192.168.1.64	192.168.1.65 – 192.168.1.94	192.168.1.95
192.168.1.96	192.168.1.97 – 192.168.1.126	192.168.1.127
192.168.1.128	192.168.1.129 – 192.168.1.158	192.168.1.159
192.168.1.160	192.168.1.161 – 192.168.1.190	192.168.1.191
192.168.1.192	192.168.1.193 – 192.168.1.222	192.168.1.223

Mr. Ed Tip

Just need the number of networks from a subnet mask?

Then use this trick for class C addresses.

Convert the submask to binary. Ex. 224 = 1110 0000
Count the number of bits used. Ex. 1110 0000 uses 3 high bits.
Move the high-order bits to low-order. Ex. 1110 0000 becomes 0000 0111.
Convert number from binary to decimal. Ex. 0000 0111 becomes 7.
Subtract 1 from the number. Ex. 7–1 gives you 6 valid subnets.

As we see in Table 1.23 using the subnet mask of 255.255.255.224, within each valid network id there is a range of addresses that can be used for host id's. The number of hosts allowed on each network is determined by how many bits remain after you have created the custom subnet. The range of addresses for each network will be between each network id. As we see in Table 1.23, since each network starts with a multiple of 32, then we are able to use the space in between the network id's for host addresses. For example, the first network id is 192.168.1.32. The second network id is 192.168.1.64. Therefore we are able to use the space in between. However, we cannot use 192.168.1.32 for a host as that is the "all zero" network id. Also we cannot use 192.168.1.63, as that is the "all one" broadcast id. So our valid range of host addresses is 192.168.1.33 to 192.168.1.62.

If we need to figure out how many host addresses we can have on each subnet, we can take the subnet mask and convert it to binary. Then we look at the space occupied by the zeros in the address, convert them to decimal, add them up, and subtract 1 from the number. Let's take a closer look at this procedure using our subnet mask from above—255.255.255.224.

1. First convert the subnet mask to binary. 255.255.255.224 becomes 1111 1111 . 1111 1111 . 1111 1111 . 1110 0000. This leaves five bits available for host id's.
2. Take the number of bits available for host addresses and flip them to ones. For example, since 255.255.255.224 leaves us with 5 bits, we have 1 1111.
3. Convert the "flipped" host bits into decimal. For example, 1 1111 becomes 16 + 8 + 4 +2 +1 = 31.
4. Now we subtract 1 from the answer achieved in step 3. This becomes 31 – 1 = 30. In this example we have 30 hosts per subnet available to us. This number excludes one address for the network id, and it excludes one address for the all-1's broadcast address. If we need more than 30 hosts per subnet, then we will not be able to use the subnet mask of 255.255.255.224.

Let's do another example using the subnet mask of 255.255.255.248. This is illustrated in Table 1.21.

1. Convert the subnet mask to binary. 255.255.255.248 becomes 1111 1111 . 1111 1111 . 1111 1111 . 1111 1000.
2. Take the number of bits left for host addresses and flip them to ones. In our example, 255.255.255.248 leaves 3 bits, so we have 111.
3. Convert the "flipped" host bits into decimal. For example, 111 becomes 4 + 2 + 1 = 7.
4. Now we subtract 1 from the answer we obtained in step 3. This becomes 7 – 1 = 6. Using the 255.255.255.248 subnet mask we will have 6 hosts per subnet.

This time let's do a class A address with a custom subnet mask of 255.255.248.0.

1. First convert the subnet mask to binary. 255.255.248.0 becomes 1111 1111 . 1111 1111 . 1111 1000 . 0000 0000.
2. Take the number of bits left for host addresses and flip them to ones. So 255.255.248.0 leaves us 11 bits for host addresses. So we have 111

1111 1111 which is equal to 2047 in decimal: 1024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 2047.

3. Now we subtract 1 from the answer we got in step 3. This becomes 2047 – 1 = 2046. Therefore, using the subnet mask of 255.255.248.0 we can support 2046 hosts per subnetwork.

We do not have to add all these numbers up in our heads. We can use the Windows 2000 calculator and switch to scientific view as seen in Figure 1.3. In order to get to the scientific view we select "scientific" from the View menu. Click on the Bin radio button to enable you to type in the binary number.

Once you have typed in the binary number, you can switch to decimal numbers by pressing the Dec radio button as seen in Figure 1.4. If we go back to our previous example, you would subtract 1 from the number you obtained from the decimal view of the calculator to obtain the number of hosts on the network.

There is an easier way to find out the number of hosts available on a subnet. We take the number of bits available for hosts, raise two to that power, and then subtract 2 from the total. Let's look at our previous example to do this shortcut.

1. In our previous example, we were using the subnet mask 255.255.248.0. If we convert this to binary, we see 1111 1111 . 1111 1111 . 1111 1000 . 0000 0000. We have 11 bits available for host addresses.

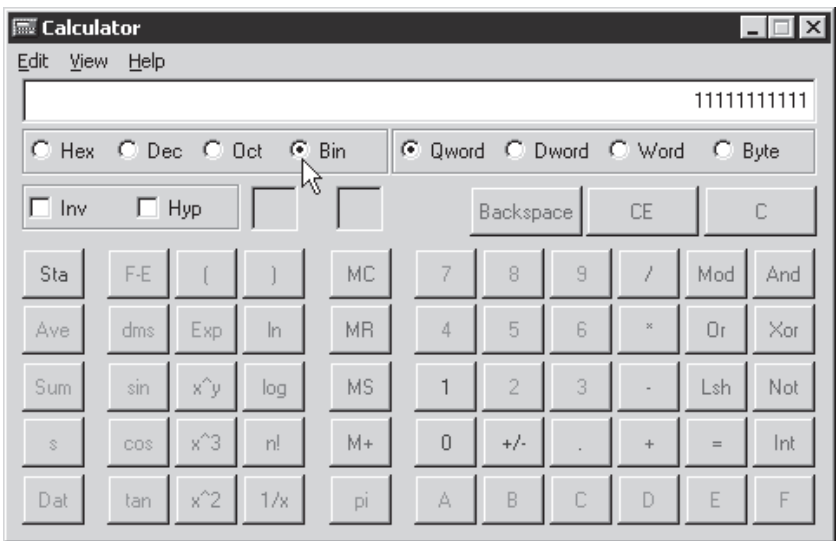

Figure 1.3 *The scientific view of the Windows 2000 calculator enables you to quickly perform binary calculations.*

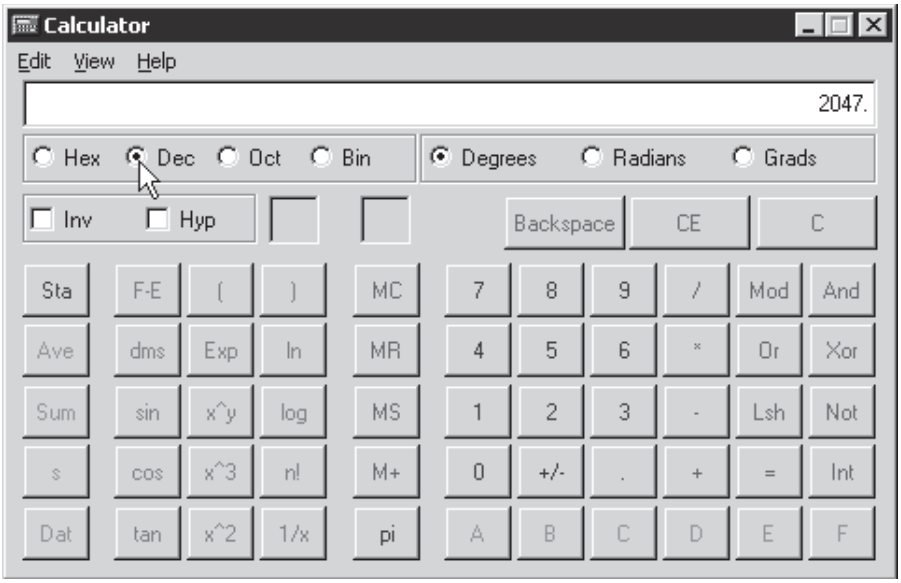

Figure 1.4 *Pressing the Dec button after you have entered a binary number converts it to a decimal value.*

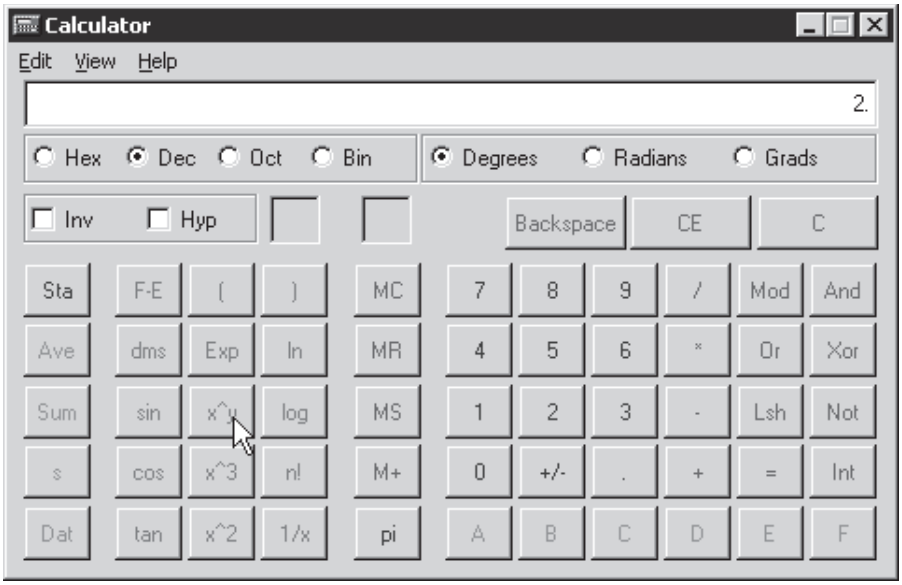

Figure 1.5 *Use the x^y button in the scientific view of the Windows 2000 calculator to perform 2 to the y power conversions.*

2. Raise 2 to the power of the available host bits. So we have 2^11 which will give us 2048.
3. Subtract 2 from the address obtained in step 2. So we have 2048 – 2 = 2046. This is exactly the same number we got in the previous example.

We can use the Windows 2000 calculator in the scientific view to perform this calculation for us. As seen in Figure 1.5, we use the decimal numbering; type in 2. Next we press the x^y button as seen in Figure 1.5 and then we simply press the = button. When the answer appears, we subtract 2 from it and obtain the number of hosts per subnetwork. Tables 1.24–1.26 show classes, subnet masks and host information.

Table 1.24 *Class A subnet table.*

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts per Subnet
2	255.192.0.0	2	4,194,302
3	255.224.0.0	4	2,097,150
4	255.240.0.0	6	1,048,574
5	255.248.0.0	14	524,286
6	255.252.0.0	30	262,142
7	255.254.0.0	62	131,070
8	255.255.0.0	126	65,534
9	255.255.128.0	254	32,766
10	255.255.192.0	510	16,382
11	255.255.224.0	1,022	8,190
12	255.255.240.0	2,046	4,094
13	255.255.248.0	4,094	2,046
14	255.255.252.0	16,382	1,022
15	255.255.254.0	32,766	510
16	255.255.255.0	65,534	254
17	255.255.255.128	131,070	126
18	255.255.255.192	262,142	62
19	255.255.255.224	524,286	30
20	255.255.255.240	1,048,574	14
21	255.255.255.248	2,097,150	6
22	255.255.255.252	4,194,302	2

Table 1.25 *Class B subnet table.*

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts per Subnet
2	255.255.192.0	2	16,382
3	255.255.224.0	6	8,190
4	255.255.240.0	14	4,094
5	255.255.248.0	30	2,046
6	255.255.252.0	62	1,022
7	255.255.254.0	126	510
8	255.255.255.0	254	254
9	255.255.255.128	510	126
10	255.255.255.192.0	1,022	62
11	255.255.255.224.0	2,046	30
12	255.255.255.240.0	4,094	14
13	255.255.255.248.0	8,190	6
14	255.255.255.252.0	16,382	2

Table 1.26 *Class C subnet table.*

Number of Bits	Subnet Mask	Number of Subnets	Number of Hosts per Subnet
2	255.255.255.192	2	126
3	255.255.225.224	6	62
4	255.255.255.240	14	30
5	255.255.255.248	30	14
6	255.255.255.252	62	6

Variable-Length Subnet Masks

Up to this point, we have been talking about subnetting as if you could only use one subnet mask for the entire address space. While in most cases you may do that and things will work out fine, there may be instances in which you want to use one subnet mask on one network and another subnet mask on another portion of your network. This can allow for a better utilization of your IP address space. For instance, if you have a very large network in one

location that has several very small networks in remote locations, it would be a waste to use a subnet mask of 255.248.0.0 for your class A network, as this will give you 30 networks, each with 524,286 hosts. This is simply a waste for a remote office with four computers, a printer, and a router.

One problem with using a variable-length subnet mask (VLSM) is that the routing internet protocol (RIP) in its first iteration (version 1) does not allow you to have multiple subnet masks for a single network. The reason for this is that RIP 1 does not know what the subnet mask is from the information that is passed among RIP 1 routers. Without this subnet mask information, RIP is forced to guess about the subnet mask to use in order to send packets to routes that it learns from other routers. If an RIP 1 router is configured with a custom subnet mask on one of the interfaces, it will use that same subnet mask for other routes it learns from that interface. For instance, if we have 10.0.1.0 with a subnet mask of 255.255.255.0 on one interface, and the RIP 1 router learns of a route to 10.0.3.0, it will use the 255.255.255.0 subnet mask for that network. If, however, it learns of a route to 15.0.0.0 it will follow classful subnet masking and use 255.0.0.0. Because of the way that RIP 1 routers handle the subnet masking, VLSM does not work. RIP 2 routers, however, do support VLSM.

So what is the big deal in using variable-length subnet masks? It allows a network to be much more efficient in the use of the IP address space. As we mentioned earlier, if you have a combination of both very large and very small networks, then you are a prime candidate for VLSM. It makes much more sense to use a more restrictive subnet mask for the smaller locations, and a less restrictive subnet mask for the larger network.

Suppose we are using 10.0.0.0 as our address space. If we use a mask of 255.255.252 we have 16,382 networks each with 1022 hosts. This does not work very well if we have a site that needs only four addresses. Until we use VLSM we simply do not have any good options. This problem is acute when using real-world addresses, especially for Internet Service Providers (ISP's) who often need to provide a couple of addresses for a small business email server, and then provide hundreds of addresses for a larger client. We simply cannot afford to waste real-world IP addresses. If we have the limitation of using one subnet mask for our network number, then we will invariably waste IP addresses. We need to come up with a way to more effectively allocate our addressing.

What Is ARP?

ARP is not related to retired persons; rather it stands for the address resolution protocol. Address resolution? I thought we just got finished talking about address resolution. No, that was name resolution. Once we have resolved the

name to an IP address, we next need to resolve that IP address to a MAC address. So what the heck is a MAC address? Well, MAC stands for Media Access Control. The term comes from the OSI model, which is discussed in great detail in Appendix H (you did read Appendix H, didn't you?) Anyway, it is the layer just above the physical layer of the OSI model. The physical layer, as it may sound like, is the hardware, the Ethernet cards, hubs, wiring, and stuff like that—things you can actually touch and feel. The MAC layer sits just above the physical layer. The MAC address, then, is a unique identifier of the interface on the Ethernet network. It is also sometimes referred to as a burned-in address, Ethernet address, and so forth and so on. This is actually the address that the packets will go to when on the network. Once at the MAC address, it will come off the wire, work its way up the OSI model at the destination machine, and eventually find its way to the application that is trying to communicate across the network.

So, ARP is the protocol that is used to find a MAC address and match it up with an IP address. TCP/IP will always need to use an ARP to resolve the MAC address when it is used on a shared-access, broadcast-based network such as Ethernet or Token Ring. The ARP protocol will essentially issue a broadcast to find the MAC address associated with an IP address. If the address is on a remote network, the ARP broadcast does not cross the router, so the response back is the MAC address of the router that will be used to get to the remote host.

ARP uses a broadcast—that is, the packet that is sent out does not have a specific destination; it is simply sent out for all to hear. The machine that hears the ARP broadcast will respond with its IP address. ARP is a very efficient protocol. We will look at ARP in more detail later on.

What Is ICMP?

The Internet Control Message Protocol let's us know when it is unable to reach a particular destination in order to communicate with it. ICMP provides troubleshooting and error reporting information for packets that cannot be delivered. Because ICMP messages do error reporting, other devices often act upon the information it provides. There are several types of messages that ICMP is capable of sending; these are listed in Table 1.27.

What IS IGMP?

Internet group management protocol (IGMP) manages membership in multicast groups. A multicast group is a collection of computers that have decided to listen for traffic that is heading to a particular IP address. This collection of computers is called an IP multicast group, or a host group. The multicast traffic is sent to a single MAC address, but due to their membership in the host

Table 1.27 *There are several messages carried by the ICMP protocol.*

Type of Message	Description
0	Echo reply. A response to a PING request. Used for troubleshooting.
3	Destination unreachable. Sent by routers and other devices to let the sender know the information cannot be delivered.
4	Source quench. Sent by routers and other devices to inform the sender that packets are being dropped due to congestion. The sender will lower the sending rate in response to this message.
5	Redirect. Sent by routers and other devices to inform the sender of a better route to a destination address.
8	Echo request. A PING request. Used for troubleshooting. An echo request is sent when you type "ping *hostname"* at a command prompt.
11	Time exceeded. Sent when the time-to-live reaches 0 during transit. This can be seen with tracert from a command prompt.

group, many machines will receive the traffic. When a computer joins a multicast group, it will then pull off all traffic destined for the multicast address.

Multicasting gives us the ability to send a single message to multiple computers. This is better than broadcasting, which goes to all computers on the network and which all network adapters must process. Only computers that have joined the multicast group receive multicasting. All other machines on the network can safely ignore the multicast traffic. The basic building block for enabling multicasting to take place is the IGMP protocol, which provides for the management of the groups.

Some of the characteristics of a multicast group are listed below.

- The membership in the multicast group is dynamic. That is, computers can join and leave the group at any time.
- Size does not matter. Due to the nature of the multicast protocol, membership in a multicast group is not limited. Any number of computers can pull the traffic off the wire.
- When a host joins the multicast group, it will send an IGMP report when the first application begins the multicast group. Only one report is sent. When additional hosts join the group, they will not have to send an IGMP report.

- When a computer (host) leaves the group, it does not send an IGMP report. Even when the last computer leaves the group, it still does not send a report.
- A multicast router will send an IGMP query at specified intervals to see if the IGMP group is still active. If it does not receive a reply to this query, then it will know the multicast group is dead. In this instance, it will no longer forward traffic to the group.
- If we are operating on a single subnet, and therefore we do not have routers, then there will only be the IGMP report that is issued when the first host joins the group.
- Membership in the multicast group is not limited to a single subnet. Indeed, multicasting is commonly utilized on the Internet for live broadcast events. Therefore, the IGMP is a routable protocol and can span multiple networks. Of course, the routers have to be smart enough to route IGMP if you need to provide multicast support. In addition, the routers will need to provide the ability for computers to register for membership in the group.
- It is possible for a computer to send traffic to a multicast group without being a member of the group.

If a computer is going to subscribe to a particular multicast group, there must be an application that is aware of the IP multicast address. For example, it could be a Netshow application that is going to "tune into" a particular Internet broadcast. Once the application knows the IP address of the multicast, then it will tell the network adapter on the machine to pull off traffic going to the particular address. For instance, if the network typology is Ethernet, it will pull off multicast traffic destined for a particular MAC address that corresponds to the IP address of the multicast group.

Chapter Review

In this chapter, we have covered a lot of territory. We began by looking at the TCP/IP protocol suite. We looked at much of the functionality provided by the protocol suite, examined client configuration issues, and finally concluded by looking at some of the considerations for deploying a TCP/IP network.

In the Next Chapter

We will take a more in-depth look at the TCP/IP protocol stack. We see how the protocol suite maps to the various layers of the OSI model, and we will examine some of the protocols commonly implemented in a standard TCP/IP protocol stack.

Next we will turn to the transmission control protocol where we will look at the methods used by it to ensure reliable connection-oriented communication. We will look at flow control methods and sequencing numbers. Following our discussion of TCP, we will look at his evil twin, the IP protocol.

In our discussion of IP we examine the way IP encapsulates data in decent detail, as well as the IP header and the like. We look at some of the common commands used with IP and find hints we will use in our troubleshooting efforts later in the book.