CHAPTER 5
IP

The most common Layer 3, or network layer, protocols in use on LANs are Internet Protocol (IP), IPX, and AppleTalk. IP, sometimes called TCP/IP, is an open standard protocol that is defined and developed by an organization called the Internet Engineering Task Force (IETF). The standards that define IP are distributed in the form of Request for Comment (RFC) documents that are freely available from many sites on the Internet. IP, IPX, and AppleTalk are all routable protocols and thus effective for large-scale networking.

Nonroutable protocols such as NetBEUI, SNA, and the older LAT protocol pose serious scalability problems to a LAN because they require that all segments sharing resources be bridged together. Breaking up broadcast domains (network regions interconnected by repeaters or bridges) by using routable protocols leads to much more efficient networks.

There are other routable protocols in use on LANs, such as the Banyan Vines VIP protocol. Banyan Worldwide officially changed its name to ePresence Solutions in 2000 and dropped support for all Banyan products in 2001 to become a service-centered, rather than a product-centered, company. Thus, this protocol is effectively obsolete and should be avoided in any LAN architecture.

Sometimes you'll encounter a handful of other routable protocols, such as DECNET and OSI. DECNET was used primarily by equipment made by Digital Equipment Corporation. When that company broke up, most organizations that had used DECNET began to migrate away from it. It can still be found in some networks, however. OSI, on the other hand, is a general purpose routable protocol that was once championed as the next great thing in networking. But it never quite managed to secure a foothold in the networking marketplace.

Over the last several years, IP has been replacing these other protocols as the favorite option of many network designers. This growth of IP has been fueled by a number of factors, particularly the public Internet, which uses IP exclusively. Accompanying

this growth has been a steady development of new features for IP. Features such as DHCP, VRRP/HSRP, multicast, and Quality of Service capabilities have effectively eliminated the technological advantages of some of these other protocols. Today, the only reason to consider other protocols is for compatibility with legacy applications.

I discuss IPX design issues in Chapter 7. IPX has some particularly interesting properties that affect how it is used in a LAN, and I still frequently encounter it in large LANs. AppleTalk, on the other hand, is a topic that would require an entire book of its own to do it justice. Its breadth puts it beyond the scope of this book.

IP is the protocol that the Internet uses, appropriately enough. Most comments I make in this section are specific to IPv4. A newer version, called IPv6 (or sometimes IPNG for "next generation"), is not yet in wide use. It seems likely that one day IPv6 will supplant the current IPv4 as the dominant version. I discuss IPv6 in more detail in Chapter 10.

IP-Addressing Basics

IP network addresses consist of 4 octets (8-bit bytes). The standard notation for this address is to express each octet as a decimal number from 0 to 255, separated by dots (dotted-decimal notation), for example, `10.212.15.101`. Because groups of these IP addresses identify network segments, it must be possible to express ranges of addresses in a simple summary notation. Using a *netmask* expresses these ranges. The netmask is another 4-octet number that is also often expressed as decimal numbers separated by dots. However, it is actually easiest to understand the meaning of the netmask in its binary representation.

Each 1 bit in the netmask indicates that the corresponding bit in the IP address is part of the network address. Each 0 bit in the netmask similarly identifies a host part of the address. As shown in Figure 5-1, if the address `10.212.15.101` has a mask of `255.255.255.0`, then the network portion of the address is `10.212.15.0` and the host portion is just the last `101`.

Address	10	212	15	101
Mask	255	255	255	0
Address	0A	D4	0F	65
Mask	FF	FF	FF	00
Address	00001010	11010100	00001111	01100100
Mask	11111111	11111111	11111111	00000000

Figure 5-1. Netmask example showing addresses and masks in decimal, hexadecimal, and binary

The netmask can also create larger or smaller networks. If the mask is `255.0.0.0`, then you can put a large number of hosts on a small number of networks. Similarly, a

mask of 255.255.255.252 allows a very small number of hosts and potentially more networks. The smaller networks that can be created this way are called subnets. Depending on the mask, though, not all IP addresses are usable.

Consider the common example where the netmask is 255.255.255.0, and assume that the network address is 10.212.15.0. As a result, the first usable host address in this range is 10.212.15.1 and the last one is 10.212.15.254.

The general rule is that you cannot use addresses that have either all ones or all zeros in the binary expression of the host parts of their addresses because these addresses are reserved for local broadcast purposes. Consider a subnet with a mask of 255.255.255.252 whose address is 10.212.15.100. The first available address to use on this subnet is 10.212.15.101. The last one is 10.212.15.102. Thus, this network segment can only have two devices on it.

Table 5-1 shows the number of host addresses available for several commonly used netmask options.

Table 5-1. Commonly used subnet masks

Netmask	Host bits available	Number of hosts	Applications
255.255.255.255	0	1	Host mask
255.255.255.252	2	2	Point-to-point links
255.255.255.248	3	6	Small special-purpose segments
255.255.255.240	4	14	Small special-purpose segments
255.255.255.224	5	30	Medium-sized segments
255.255.255.192	6	62	Rarely used
255.255.255.128	7	126	Rarely used
255.255.255.0	8	254	General-purpose segments

Notice that the first entry in this table, the one with netmask 255.255.255.255, has a binary representation that is all ones. In effect, the entire address is a network address. Clearly, this leaves no room for host addresses, but that doesn't mean that you can't configure a host on this network; you just can't differentiate between the hosts that are within a particular network using this address. As long as only one host is in the network, there is no conflict.

How can there be a network with only one host? What will that host send its packets to in order to get to other networks? Paradoxically, this netmask can be quite useful; it is typically used as a loopback address, for example. This is a purely internal address within a host that can be used for special purposes such as management. It is also common to use a loopback address for tunnel terminations, since this loopback interface is always guaranteed to be active, even if the device itself is on a backup circuit. Loopback addresses are also often used in conjunction with dial backup applications.

The 192 and 128 masks are rarely used for subtle compatibility reasons. This has to do with the closely related concepts of multiple subnet broadcasting and address classes. These concepts are now considered optional parts of the IP Core standard. I discuss these issues later in this chapter.

At one time it was fashionable to use more complicated subnet masks. Instead of just using masks that had all ones up to a certain bit, usually in the last octet, some large networks used masks such as `255.255.255.56`, for which the bit pattern in the last octet is `00111000`. The idea of these masks was to provide a way to open up a smaller address range. For example, the engineer could start with a mask of `255.255.255.248` (`11111000`). She might initially assign a particular Ethernet segment the subnet `192.168.1.16`. Then, as that segment grows beyond the 6 available addresses, she could give it an additional block by just changing the subnet mask to `255.255.255.120` (`01111000`). The available range of addresses now includes `192.168.1.17-23` and `192.168.1.144-150`. The range from 17 to 23 is the addresses that have the 0 bit in the first position. The address 144-150 has a 1 in this bit position. Table 5-2 shows why this works.

Table 5-2. Subnetting "counting from the left"

First three octets	Last octet	Binary last octet	Comment
`255.255.255.`	120	`0-1111-000`	Mask
`192.168.1.`	16	`0-0010-000`	All zeros
`192.168.1.`	17	`0-0010-001`	First address available
`192.168.1.`	23	`0-0010-111`	Last address, first half
`192.168.1.`	144	`1-0010-000`	First address, last half
`192.168.1.`	150	`1-0010-110`	Last address available
`192.168.1.`	151	`1-0010-111`	All ones

This procedure of subnetting is called "counting from the left." While it is effective, it is not commonly used anymore for several reasons. First, how the range from 17–23 is connected to the range from 144–150 confuses most casual observers. This confusion will invariably make troubleshooting much more difficult than it needs to be. Second, if you really want to use this scheme, then you have to set the second range aside just in case you need it later. If you think you might need more addresses, though, why not just assign larger subnets in the first place? The third reason to avoid using this sort of scheme is that specifying a subnet mask by just counting the one-bits has become commonplace. So the mask `255.255.255.240` would be the 28-bit mask. It is common to specify the subnet `192.168.1.16` with this mask as `192.168.1.16/28`. But `255.255.255.120` also has 28 bits of ones, so there is a risk of confusing these two networks.

Finally, this type of subnetting scheme clearly breaks one of network design's Core principles. It is inherently complicated and difficult to understand. Simplicity is always the best policy in network design.

Some networks may still use a counting-from-the-left subnetting scheme. This scheme is used because, once started, it would be difficult to get away from it without readdressing large numbers of end devices. However, I believe that this technique is not good, and I recommend migrating away from it if possible.

IP-Address Classes

IP defines four network classes called A, B, C, and D. Class A networks provide the largest number of addresses. Before subnetting is done, a Class A network has a mask of `255.0.0.0`. Unsubnetted, it supports up to 16,777,214 host addresses. Of course, it would be extremely unusual to use a Class A address without subnetting. Similarly, Class B networks have a mask of `255.255.0.0` before subnetting. Continuing the pattern, Class C networks use `255.255.255.0` and Class D networks consist of only one address, `255.255.255.255`.

Strictly speaking, Class only refers to the network mask before any subnetting is done. Sometimes people use the language loosely and call a subnet that has a mask of `255.255.255.0` a Class C subnet. That is not really what "class" means, however. There is actually a simple rule involving the first few bits of any IP address that determines what the class of a network is. If the first bit is a 0, which is to say that the first octet of the address has a value from 1 to 127, then it is a Class A address. If the first bit is 1 and the second bit is 0, then it is a Class B address. Class B addresses run from 128 to 191; Class C addresses have 1s in the first 2 bits and a 0 in the third bit, which includes everything from 192 to 223; Class D networks begin with 3 bits of 1s and a 0 in the fourth bit; a final group of Class E addresses includes everything else. Table 5-3 illustrates this.

Table 5-3. Classes of IP addresses

Class	Range of network addresses	Mask	Maximum number of host addresses per network	Number of networks
A	0.0.0.0–127.0.0.0	`255.0.0.0`	16,777,214	128
B	128.0.0.0–191.255.0.0	`255.255.0.0`	65,534	16,384
C	192.0.0.0–223.255.255.0	`255.255.255.0`	254	2,097,152
D	224.0.0.1–239.255.255.255	`255.255.255.255`	1	248,720,625
E	240.0.0.1–255.255.255.255	`255.255.255.255`	1	248,720,625

Note that some of these address ranges are reserved and will never be available for normal network addressing. For example, the networks `0.0.0.0` and `127.0.0.0` are reserved. The network `0.0.0.0` is used as a generic broadcast address and every

host has a local loopback address of `127.0.0.1` by which it knows itself. In many routing protocols, the global default address is designated as `0.0.0.0` with a netmask of `0.0.0.0`.

These entire ranges are set aside and not used for anything. The other important block of reserved addresses is the 224-239 range. Everything that starts with 224 through 239 is reserved for multicast addresses. An address starting with 255 in its first octet will probably not be assigned because of potential confusion with broadcast address conventions. Similarly, the entire range of Class E addresses is effectively unusable.

Originally, the classification scheme stopped with the last two classes taken together as Class D. The newer Class E range was developed to separate a distinct group of single-host addresses from the emerging multicast requirements.

Class is now considered an outdated concept. I have discussed it here because the word is still used. The various Internet authorities have stopped allocating IP addresses according to class, and all routing through the Internet uses Classless Inter-Domain Routing (CIDR). The currently preferred method for expressing the size of a network is to use the number of bits in the mask. For example, you would refer to the Class A address `10.0.0.0` as `10.0.0.0/8`. If you grouped together the first two unregistered Class B networks, you would call the resulting range of addresses `172.16.0.0/15`.

In CIDR, the IP address can be divided into host and network portions at any bit, but there are still some important addresses in this scheme. The address `0.0.0.0` is sometimes used as a source address. Any host can use it, particularly when the host doesn't know its own address (for example, during the first steps of a DHCP query). Similarly, it is possible to use an address in which the network part consists of all zeros and the host part is properly specified. Again, this address can only be used as a source address.

The address `255.255.255.255`, which is all ones in binary, is used as a destination address in a local broadcast. In effect, it indicates all hosts on the local network. This address is also used frequently when a device doesn't know anything about its local network number, as in the first steps of a DHCP query.

Related to this issue is the address in which the host portion is all ones in binary and the network portion is a real subnet address. Again, this address can only be used as a destination address. In this case, the packet is intended for all hosts on a particular subnet. The CIDR specification also allows one to specify a broadcast destination address in which the main network address is a real address and the subnet address and the host portion of the address are all ones. This specification is intended as a broadcast to all hosts in all subnets. However, I have never seen this specification used in practice, and it seems less than useful, if not unwise.

As in the Class system, any address with the decimal number 127 in the first octet is reserved as a local loopback address. Even in CIDR, all Class D addresses are reserved for multicast purposes.

ARP and ICMP

Address Resolution Protocol (ARP) and Internet Control Message Protocol (ICMP) are both key low-level parts of the IP protocol that one encounters every day. ARP is how end devices on the same network segment learn the Layer 2 MAC addresses for one another. ICMP is used for a wide variety of different network control and management functions.

ARP

For a device to take part in a Layer 3 protocol such as IP, it has to be able to send and receive these packets through a Layer 2 medium. Suppose a device wants to communicate with another device on the same Ethernet segment. It knows the IP address for this destination device, but, in general, it doesn't know the Ethernet MAC address. The 802.3 protocol says that if this is going to be a point-to-point conversation, then the Ethernet frame must have valid source and destination addresses.

The conversation can't begin until these two devices discover one another's MAC addresses. And, of course, this problem isn't specific to Ethernet. The same problem exists on every Layer 2 network, whether it is Token Ring or ATM. You must have a valid Layer 2 destination address before you can send even the first packet.

This is the problem that ARP solves. For simplicity, I will restrict this discussion to Layer 2 network technology that supports broadcasting. ARP still exists for non-broadcast media such as ATM networks, but it becomes significantly more complicated in these cases.

The solution is remarkably simple. Every device on the network segment receives broadcasts. All one has to do is send out a broadcast packet called an ARP Request and look for the required destination IP address. If one of the devices receiving this packet is the owner of this IP address, it sends back an ARP Reply.

The body of the ARP Request packet contains both the sender and receiver IP and MAC addresses. Some information is, of course, duplicated in the Layer 2 frame header. Since the sender doesn't actually know the receiver's MAC address, it fills in the broadcast address `FF:FF:FF:FF:FF:FF`.

The ARP Reply then contains similar information. The sender and receiver fields are swapped and the missing MAC address is filled in.

When the first device receives the ARP Reply in response, it puts the information in its ARP Cache. This cache is simply a local table of IP and MAC addresses for all devices it has communicated with recently. This cache allows the first device to avoid

another ARP exchange as long as the two devices are in contact. However, if a device is not heard from in a standard timeout period, it is removed from the table. This period is usually about 20 minutes, but individual devices can define it locally.

ICMP

The first kind of ICMP packet most people think of is a *ping*. The ping function is an echo request and response facility that allows one to test whether certain devices are reachable on the network. ICMP actually has a wide range of other uses, particularly for reporting network errors.

The ping function is relatively simple. One device sends an ICMP echo request packet to another. The receiving device then responds to this packet with an echo response. This response has many uses, particularly in network management. It is also frequently used by applications that want to verify that a server is available before starting a session. For network management, it provides a simple way to measure end-to-end latency in the network—by taking the time difference between sending the request and receiving the response packet.

ICMP packets also provide a way to report several important error conditions. For example, one fairly common error situation is to have a packet dropped because there is no route available to the destination.

Another important example is when an IP packet is too large to pass through a particular section of the network. Ordinarily, this is not a problem because the router simply breaks up the packet into fragments and passes it along. However, some applications set a flag in their packets to prevent them from being fragmented. In this case, the router has no choice but to drop the packet.

In each of these cases, the router that drops the packet alerts the source device of the problem by sending a special ICMP message. This message allows the application or the user to take appropriate action to fix the problem. In the case of the large packet, the router might simply try again using smaller packets, for example.

Another common and important type of ICMP message is the ICMP Redirect. The redirect is most frequently seen when two or more routers are on the same network segment as the end device. If these routers handle different sets of IP addresses, the end device could inadvertently send a packet to the wrong router. This is particularly common if one of the routers is configured as the default gateway for the end device.

When this happens, the first router simply forwards the packet over to the other router and sends an ICMP redirect message. This message tells the end device that it has delivered the packet, but that, for future reference, another router has a more direct path. The end device should then update its internal routing table to use this second router the next time it sends such a packet.

This issue is particularly important for the network designer to understand because some devices do not respond correctly to ICMP redirection. In these cases, it is often necessary to configure the routers to not send these messages and just to forward the packets. Otherwise, the segment can suffer from extra congestion due to all of the redirection messages—one for every application packet.

In general, I prefer to only have one router on any end device segment, configured as the default gateway for all end devices. As I've mentioned earlier, this router can be made redundant by adding a second router and using HSRP or VRRP. As far as the end devices are concerned, there is only one way off the segment. Network segments built this way should never see any ICMP Redirect messages.

Network Address Translation

One common feature of many firewalls is Network Address Translation (NAT). Many routers now offer NAT as an optional service. NAT can function in several different ways, but they all involve rewriting the source address in IP packets. NAT is also sometimes called address masquerading.

Figure 5-2 shows a typical NAT implementation. The protected internal network is on the inside of the firewall, to the left. The external network is to the right. Perhaps the most important feature here is that the IP addressing of the internal network is completely unregistered.

I will discuss IP addressing schemes shortly, but one common scheme involves using an open range of addresses that cannot be owned by any organization. These addresses are, in effect, public domain. Because of this, they can never be presented to a public network. Since any organization is allowed to use the IP address range `10.x.x.x`, for example, then it is impossible for the public network to know which of the millions of sites using this range is the right one.

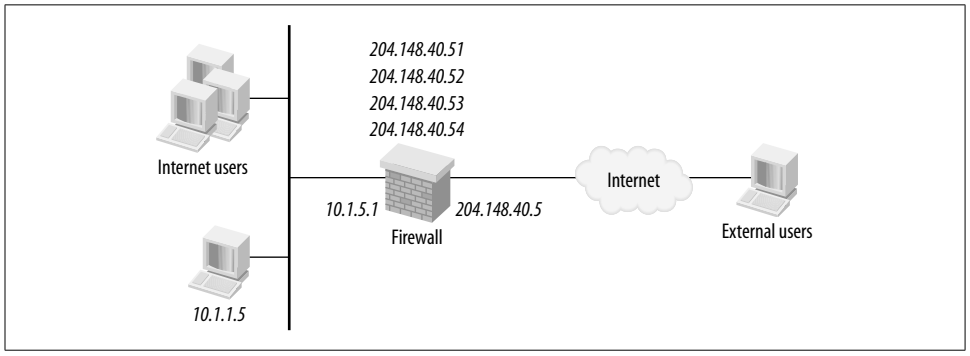

Figure 5-2. Network Address Translation

Figure 5-2 has the interior of the network using 10.x.x.x for its address range. This means that the firewall must rewrite every packet that passes through it so that the 10.x.x.x addresses on the inside are changed to legitimate addresses on the outside. A workstation on the inside of the firewall uses the IP address 10.1.1.5. The outside of the firewall has the legitimate registered address 204.148.40.5. For the example, assume that this organization has registered the entire range 204.148.40.x.

Every packet passing through the firewall from 10.1.1.5 must have its source address rewritten to something in the range 204.148.40.x. There are three common ways to do this. You can make every internal device appear with the same address as the outside of the firewall, 204.148.40.5. The second option is to create a range of legitimate addresses, such as 204.148.40.51, 52, 53, and 54. These addresses are then uniquely and permanently assigned to specific internal devices, so 10.1.1.5 will always appear on the outside of the firewall, 204.148.40.51, 10.1.1.6 will always appear as 204.148.40.52, and so forth.

The last commonly available option is to assign a pool of address such as 204.148.40.51-54, as in the previous example. This time, the addresses will be dynamically associated with internal devices. When one device wants to make a connection to the Internet, it gets whatever address is the next one available in the list. The next internal device gets the second address from the list, and so forth. This technique makes better use of the address resources than the previous example because an idle device returns its external address to the pool.

The most common method, however, is the first one, in which every internal device has the same IP address on the outside of the firewall. This situation makes the entire network look to the outside world as if it is a single, very busy device. Internally, the firewall must do a very difficult task, however. It must keep track of which of the many sessions are associated with which internal devices.

This method is considerably easier to use with TCP than with UDP-type services because the firewall can keep track of the different TCP sessions. With UDP, the firewall has to try to associate incoming packets with previous outgoing UDP packets. If a particular internal device sends out a UDP packet on a particular port number, then the firewall simply waits a short time for a response from that same external IP address and directs it to the originating internal device.

The problem becomes even harder for ICMP packets such as ping, though. The firewall has only the IP address of the external device to connect to the originating internal device. If several people all try to ping the same external device at the same time, it can be very difficult for the firewall to figure out which response packets to direct to which internal devices. To the outside world, all packets look like they came from the firewall itself; they all have the same source IP address.

Not only must the firewall rewrite the source address of every outgoing packet, but it must also rewrite the destination address of all incoming packets. If it doesn't know how to rewrite the address, then most firewalls take the safe approach—they assume that the incoming packet is unsolicited and drop it.

NAT can cause serious confusion in some places. Specifically, some applications include the IP address information somewhere inside the packet's data segment. One common example of this is SNMP. If one attempts to manage a network through a firewall, NAT can become extremely confusing.

The worst case comes when a network management service vendor tries to manage the networks of several different clients who all use the same IP address ranges. The vendor may try to use NAT to rewrite the addresses of the monitored client machines, but data contained inside the SNMP packets indicate the real source address, and the conflicting address ranges have to be removed by other means.

A worse example is seen in the applications of certain information service providers. Some of these applications work by means of TCP connections from the client workstation to the vendor's server. The TCP connection starts with a packet from the client workstation that passes through the firewall. As I discussed earlier, the packet's header contains four key pieces of information: the source and destination IP addresses and the source and destination TCP port numbers. The firewall rewrites the source address and often also rewrites the source TCP port. The server then responds with a packet addressed back to the modified source address (which is delivered to the firewall) and the modified source port. This information is rewritten by the firewall to the original values and directed back to the client workstation.

If the source IP address and port number are also contained within the data part of the packet, perhaps for some sort of authentication system, then the server has considerable room to be confused. It cannot communicate directly with the real source IP address indicated. The real source TCP port number is also of no use to it.

Unfortunately, there is no universal solution to this problem. Some firewalls are able to work around parts of the problem—for example, by maintaining the source TCP port number. But in the worst examples, the only way around the problem is to simply eliminate the firewall, eliminate NAT, and use a standard router with extensive packet filtering to implement the required security precautions.

Multiple Subnet Broadcast

On any individual subnet, you can issue a broadcast to every other device on the subnet by doing two things. First, on the data link layer (Layer 2), you set the MAC address to the appropriate broadcast address for that medium. For Ethernet and Token Ring, the broadcast address is `FF-FF-FF-FF-FF-FF`—that is, all bits are set to one. Note that this address is consistent with what I mentioned earlier when talking

about multicast addresses on Ethernet. Any time the lowest order bit of the destination MAC address is set to one, the packet is either a multicast or broadcast. Then, on the IP layer, you just set the destination address of the packet to be the subnet address followed by all ones or all zeros for the host portion of the address.

In fact, the standard prefers using all ones for the broadcast addresses, but both are used in practice. If the subnet uses a mask of `255.255.255.0`, as in `10.1.2.0`, then the broadcast address for this subnet would be `10.1.2.255` (all ones, the preferred version) or `10.1.2.0` (all zeros). Similarly, if the mask were `255.255.255.240` for the subnet address `10.1.2.32`, then the all-ones broadcast address would be `10.1.2.47`. The addresses in this subnet that are available for hosts range from `10.1.2.33` to `10.1.2.46`.

The IP standard defines another type of broadcast called the all-subnets broadcast, which is seldom used in practice. It is considered optional, but on most equipment it must be explicitly disabled if it is not required. The all-subnets broadcast is exactly what it sounds like. It allows a broadcast to be sent simultaneously to every subnet in a network. The address for the all-subnets broadcast is simply the broadcast address for the entire network address. The previous example had a subnet of `10.1.2.32` with a mask of `255.255.255.240`. But this is a subnet of the Class A network `10.0.0.0`. Thus, you can send an all-subnets broadcast by addressing a packet to `10.255.255.255`.

If you were dealing with a subnetted Class C network such as `192.168.1.0`, then you have a mask of `255.255.255.0` for the whole network. The subnets may have a mask of `255.255.255.224`, for example. Then the subnets would be as shown in Table 5-4.

Table 5-4. Example subnets on a Class C network

Subnet	Binary representation of last octet	Comment
192.168.1.0	000-00000	All zeros in the network portion of the address
192.168.1.32	001-00000	First nonzero subnet
192.168.1.64	010-00000	
192.168.1.96	011-00000	
192.168.1.128	100-00000	
192.168.1.160	101-00000	
192.168.1.192	110-00000	
192.168.1.224	111-00000	All ones in the network portion of the address

This table should make the mechanics of subnetting clearer. Just as the all-zeros or all-ones addresses in each subnet are not used for host addresses, the all-zeros and all-ones subnet addresses are also problematic. Specifically, if you want to do any all-subnets broadcasting, you cannot use these networks. However, all-subnets broadcasting becomes ill-defined with CIDR.

If you look back at Table 5-1, it becomes clear why the subnet masks 255.255.255.192 and 255.255.255.128 are rarely used. The bit pattern for the number 192 is 11000000. If you subnet a Class C network, only the first two bits of the last octet are available for indicating subnets. If you don't use the all-zeros or all-ones subnets, you are left with only 01-000000 and 10-000000, which are 64 and 128, respectively. The situation is even worse if you want to use a mask of 255.255.255.128 on a Class C address because the bit pattern for 128 is 10000000, leaving you only one bit for selecting the subnet. This bit can be either one or zero, and that means it is always either all ones or all zeros, and therefore possibly reserved for broadcasts.

There are three important caveats to all of this. Because multiple subnet broadcasting is optional, you can still use the all-ones or all-zeros subnets if you just disable the feature on every router in the network. Since the routers are the only devices that care about propagating any packet between subnets, they are the only devices that need to be affected by this change.

The second caveat is that only subnets of Class C networks are covered here. If you subnet a Class A network, then you need ensure that you have a nonzero or nonone bit somewhere in the subnet address. However, this is a dangerous strategy. There have been a number of non-compliant IP implementations over the years, and the inability to distinguish properly between IP address classes is a bug that has appeared in some of these flawed implementations. In particular, some implementations assume that every subnet is a subnet of a Class C, regardless of the first octet in the address.

The third caveat is that you can use CIDR. As mentioned earlier, traffic passing through the Internet already assumes classless addressing. However, many organizations still use class-based addressing internally. If you enable classless routing, then the multiple subnet broadcast option also automatically disappears in most CIDR implementations because there is no longer any way to define a unique broadcast address for the entire network.* If the designer wants larger subnets, such as a 255.255.255.128, or even larger subnets, as in 255.255.254.0, it is best to explicitly disable the all-subnets broadcast feature and enable classless routing on all routers in the network.

On some types of routers, the command to disable multiple subnet broadcasting takes the approach of allowing the all-zeros subnet addresses. But it should be clear that this is another way of saying the same thing, since you can't have all-subnets broadcasting if you don't reserve the all-zeros and all-ones subnet addresses for that purpose.

* Note that the CIDR documents do not rule out an all-subnets broadcast. RFC 1700, when describing CIDR, states that all-subnets broadcasts still exist. However, it is not fully defined, and I am not aware of any working implementations or of any useful applications. Using multicast would probably be a better way to accomplish the same thing.

General IP Design Strategies

Until now, I have looked only at theoretical ideas about how subnetting works. Now I want to talk about how to use it in a real network. The first step, before anything else can be done, is to decide how many segments are required and how many devices these segments need to support. These estimates need only be rough ballpark estimates because a good designer always assumes that a network will grow in time. This estimate constrains what sort of IP-address range is required. Generally, the network should be built out of subnets of a larger network because you will want to take advantage of route summarization later. Simply allocating a new distinct Class C network for every user segment is not useful.

Unregistered Addresses

At one time, there was a hot debate about the use of unregistered addresses on large LANs. Many organizations developed internal policies that forbade the use of unregistered addresses on principle. Before the advent of firewalls with NAT, it would have been impossible to connect these networks to the public Internet (or even to build nonpublic shared Internets between collaborating organizations) without the IETF centrally controlling all IP address allocations.

This sort of policy had an unfortunate side effect that nearly destroyed the Internet. An IP address has only 4 octets, so there can be at most 4,294,967,295 devices. Four billion sounds like it should be enough, but remember that the first half of these addresses are allocated as Class A networks, of which only 128 are possible (and some are reserved, as mentioned above). The next quarter includes the 16,384 possible Class B addresses (and again, some of these are reserved). Thus, three quarters of the available address range is used up on just 16 thousand large companies, universities, and government agencies. The Internet has many millions of participants, though, and they all must have registered IP addresses. Clearly, it isn't a possible, practical, or responsible use of scarce resources to use registered addresses on internal corporate networks.

The alternative is using unregistered addresses, but you have to be careful with unregistered addresses. If you arbitrarily pick an address range for internal use, the chances are good that this range is already in use somewhere on the Internet. As long as you hide everything behind a firewall and use NAT to hide your unregistered address, you won't conflict openly with anything. But one day you want to exchange email with whoever actually owns this address range or even connect to their web site, it will not work.

There is an easy resolution to this problem: you just need to use addresses that you know are not in use and never will be. The IETF set aside several ranges of addresses for exactly this purpose, and they are documented in RFC 1918. The allowed ranges are shown in Table 5-5.

Table 5-5. RFC-allowed unregistered IP addresses

Class	Network	Mask	Comment
Class A	10.0.0.0	255.0.0.0	One large Class A network
Class B	172.16.0.0 through 172.31.0.0	255.255.0.0	16 Class B networks
Class C	192.168.0.0 through 192.168.255.0	255.255.255.0	255 Class C networks

Anybody is free to use these addresses for anything they like, as long as they don't connect them directly to the Internet. For a very small LAN, such as a home or small office, it makes sense to use one of the `192.168` addresses. In the author's home LAN, I use `192.168.1.0`, for example, with a firewall to connect to the Internet. The Internet Service Provider (ISP) supplies a registered address for the outside interface of the firewall. For larger networks where a Class B is required, the organization is free to pick from any of the 16 indicated unregistered addresses. There is only one unregistered Class A network, so almost every large network in the world uses `10.0.0.0` for its internal addressing. This doesn't cause any problems unless these organizations need to communicate directly with one another without intervening firewalls, which sometimes happens, particularly when one organization provides some sort of network service to another, as might occur with information service providers, network-management service providers, and corporate mergers. When conflicts like this occur, the best way to get around them is to carve off separate sections of the network interconnected by firewalls performing NAT.

IP Addressing Schemes

A successful IP addressing scheme operates on two levels. It works on a global level, allowing related groups of devices to share common ranges of addresses. It also works on a local level, ensuring that addresses are available for all local devices, without wasting addresses.

The global issue assumes that you can break up the large network into connected regions. Having done so, you should summarize routing information between these regions. To make route summarization work in the final network, you need a routing protocol that is able to do this work for you. Thus, a key part of any successful IP addressing scheme is understanding the specific routing protocol or protocols to be used.

Another important global-scale issue is the network's physical geography. An organization with a branch-office WAN usually needs a large number of small subnets for each of the branch offices. It also probably has a similar large number of point-to-point circuits (perhaps Frame Relay or ATM virtual circuits) for the actual WAN connections.

However, an organization that is concentrated on a single campus, perhaps with a small number of satellite sites, needs to break up its address ranges in a completely different way. Many organizations are a hybrid of these two extremes, having a few extremely large sites and a large number of small sites. Other organizations may start off in one extreme and, through growth, mergers, and acquisitions, find themselves at the other end. I can't really recommend a single IP addressing strategy that suits every organization, but I can talk about some principles that go into building a good strategy:

- Create large, yet easily summarized, chunks
- Set standard subnet masks for common uses
- Ensure that there is enough capacity in each chunk for everything it needs to do
- Provide enough flexibility to allow integration of new networks and new technologies

Easily summarized ranges of addresses

Take these points one at a time. First, creating large, yet easily summarized, chunks is relatively straightforward. Summarization makes it easier to build the network in distinct modules. This means that routers can deal with all of the routes for a particular section of the network with a single routing table entry. This ability is useful, no matter what sort of dynamic (or static) routing protocol is used in the network.

To be summarized, you have to be able to write the chunk of addresses with a single simple netmask. For example, if you use the `10.0.0.0` unregistered Class A range, then you might make your chunks by changing the second octet. The first chunk might be `10.1.0.0` with a mask of `255.255.0.0`. This chunk will usually be written `10.1.0.0/16` to indicate 16 bits of mask. Then the second chunk would be `10.2.0.0/16`, and so forth. If the mask turns out to be too small for the requirements of the network, it is easy enough to work with a shorter mask. Then you might summarize in groups of four, as in `10.4.0.0/14`, `10.8.0.0/14` and so forth. Here, the mask is `255.252.0.0`.

Another approach to creating easily summarized chunks of addresses uses the unregistered Class B range of addresses. In this case, you might simply start with `172.16.0.0/16` for the first chunk, `172.17.0.0/16` for the second, and so forth. Remember that only 16 of these unregistered Class B ranges are available. Instead, you might make your chunks smaller, as in `172.16.0.0/18`, `172.16.64.0/18`, `172.16.128.0/18`, and `172.16.192.0/18`.

The two key issues here are figuring out how many of these chunks are required and how big they must be to accommodate the network's requirements. If the number of chunks becomes too large, then you will need to create a hierarchy of address ranges. As I will discuss in Chapter 6, these chunks are appropriate for the size of an Open Shortened Path First (OSPF) area, but if you create too many areas, you need to be

able to break your network into multiple Autonomous Systems (ASes). Then you will also require route summarization between ASes. I define these terms in Chapter 6, but for now you can just think of an OSPF area as an easily summarized group of addresses and of an AS as an easily summarized group of areas.

For a quick example of how this might be done, suppose you want to use `10.0.0.0` for the network. Then you might make your OSPF areas with a mask of `255.255.0.0`, so each area has the same number of addresses as one Class B network—they will be denoted `10.0.0.0/16`, `10.1.0.0/16`, `10.2.0.0/16`, and so forth. You might decide that for performance reasons you need to restrict the number of areas within an AS to a number like 50, for example. Unfortunately, 50 is not a nice "round" binary number, but it is not far from 64, which is.

Providing a few too many potential areas may turn out to be useful later, if you have to make one AS slightly larger than the others. It is not a bad thing to have to go to 64. In this case, the ASes are summarized on a mask of `255.192.0.0`. The first one will be `10.0.0.0/10`, the second will be `10.64.0.0/10`, and so forth.

One final note on summarizing—the chunks do not all need to be the same size. One area can have small and large subnets, as long as the area can be summarized. Similarly, one AS can have small and large areas, as long as you can still summarize every area. You can even mix differently sized AS, as long as they also can be summarized easily.

The first AS could be `10.0.0.0/10`, as noted previously. The second and third could be `10.64.0.0/11` and `10.96.0.0/11`. The second 10-bit mask range is broken into two 11-bit mask ranges. Breaking up the ranges this way—by subdividing some of the larger chunks with a larger mask—is the best way to look at the problem. The same idea applies to subdividing area-sized chunks that are larger than required.

Dynamic routing protocols don't require this sort of summarization, but summarizing routes will result in a more stable and easily administered network.

Sufficient capacity in each range

How big does each chunk of addresses need to be? As mentioned before, it is easier to subdivide ranges of addresses than it is to merge them. You probably want to err on the large side, if you can. The only way to answer the question is to decide what you're going to put in this range. For the time being, suppose that the address range is for an OSPF area. The same reasoning applies to sizing OSPF AS, but on a larger scale. If you use a different dynamic routing protocol, such as RIP or EIGRP, the differences are again just a matter of the appropriate scales for these protocols. Focusing on the OSPF area version of the problem, the usual rule for area size is 50 routers in an area.

I will discuss this in more detail later when I talk about OSPF. As you will also see later, there must be a Core or backbone area that all of the other areas connect to. Thus, you have to be concerned about sizing the Core area as well as the peripheral areas.

The largest hierarchical LAN designs, discussed in Chapter 3, had two routers in each VLAN Distribution Area and a central Core with a handful of routers. Even with this design, the network would probably need at least 15 VLAN Distribution Areas before needing to be broken up into OSPF areas. Conversely, OSPF area structure becomes important very quickly in even a modest-sized WAN. The sizes of your LAN and WAN OSPF areas will probably be completely different, and one certainly wouldn't expect them to have the same sort of internal structure.

This book is about building large-scale LANs, so I will carry on with a relatively simple example involving a large campus network that connects 100 departments. Each department is its own VLAN Distribution Area in your hierarchical design model, so each department has two routers and several VLANs.

A VLAN Distribution Area is far too small to be a good OSPF area. However, all routers in each OSPF area must connect to the Core area through a small number (I will assume two) of Area Border Routers (ABRs). The main constraint in the size of each OSPF area is not the rule of 50 routers per area. Rather, you will quickly run into bandwidth limitations on those ABRs if you connect too many VLAN Distribution routers to them, so you might want to set a limit of 10 VLAN Distribution Areas per OSPF area, or perhaps only 5. A detailed bandwidth requirement study would yield the most appropriate topology.

Suppose that the network will have five VLAN Distribution Areas per OSPF area. You need to look at how many VLANs live in each Distribution Area and the netmask of each VLAN. If you know you have up to 25 VLANs, each with a mask of `255.255.255.0`, then you can finish the puzzle. You need about 125 Class C-sized subnets in each OSPF area, and you need this chunk to be summarized. That number is easily accommodated in a Class B-sized range. With a mask of `255.255.0.0`, you could fit in 256 subnets.

Note that this example implies that a mask one bit longer could have been used to accommodate 128 subnets. However, as I mentioned earlier, it is good to err on the high side in these sorts of estimates. The difference between 125 and 128 is only a 2% margin of error, which is far too close for such back-of-the-envelope estimates.

The whole campus has 100 departments and a total of 20 departmental OSPF areas, each containing 5 departments. In addition, the network has a Core OSPF area, with a total of 21 areas. If each area has a mask of `255.255.0.0`, then the whole network has to be able to accommodate 21 masks of this size. Clearly, this network won't be able to use the `172.16.0.0/16-172.31.0.0/16` set of Class B addresses. There is more than enough room in the `10.0.0.0/8` Class A network.

This example shows the general thought process that needs to be followed when finding the appropriate sizes for the area-sized chunks of addresses. It is also easily extended to AS-sized chunks of addresses. Suppose, for example, that bandwidth and stability issues force the network engineer to break up the Core of this example campus network. To make the example interesting, suppose that the engineer has to break the network into three ASes.

There are 20 OSPF areas to divide among these three, which could mean two sets of 7 and a 6. Each of these areas has its own Core area, giving two 8s and a 7. The nearest round number in binary is 8. There is no room for growth, so once again, it is good to err on the large side and use groups of 16. This means that the ASes will have a summarization mask of `255.240.0.0`. The first one would be `10.0.0.0/12`, the second `10.16.0.0/12`, and the third `10.32.0.0/12`.

Standard subnet masks for common uses

One of the most useful things network designers can do to simplify the design of an IP network is to set up rules for how to use subnets. There are actually three types of rules:

- What subnet masks to use for what functions
- How to select a subnet from the larger group of addresses for the area
- How to allocate the addresses within the subnet

The fewer different subnet masks in use in a network, the easier it is to work with the network. Many people, particularly less experienced network personnel, find the binary arithmetic for subnetting confusing.

In many networks, it is possible to get away with only three different subnet masks. For point-to-point links that can only physically support two devices, you can safely use the longest mask, `255.255.255.252`. For most regular LAN segments, you can use `255.255.255.0`, which is the same size as a Class C, and relatively easy to understand. Then you can allocate one other netmask for special subnets that are guaranteed to remain small, but nonetheless contain more than two devices. A good mask for this purpose is `255.255.255.240`, which supports up to 14 devices.

Since there are broadcast-efficiency issues on larger LANs, it is best to try to keep the number of devices in a VLAN below a reasonable threshold. A good natural number for this purpose is the 254 host maximum allowed by the 24-bit mask, `255.255.255.0`. Nonetheless, many organizations like to expand their VLAN-addressing range by using a mask of `255.255.254.0` or even `255.255.252.0`. There is certainly nothing wrong with doing this. But if a network uses a mixture of VLANs with masks of `255.255.252.0` and `255.255.255.0`, it is very easy to get confused in the heat of troubleshooting a difficult problem. For this reason, I tend to avoid these larger masks. I also feel that broadcast issues make Class C–sized subnets more efficient in a VLAN, but this latter issue will not be true on every network.

Many organizations also like to try to improve their address-allocation efficiency by using other in-between-sized subnet masks. For example, for smaller user LAN segments, they might opt to use a mask of 255.255.255.224. This mask undoubtedly winds up being necessary when trying to address a large network with a single Class B address. For example, if a network designer insisted on using a registered Class B range for a network, he might find that this kind of measure is needed to avoid running out of addresses. Getting into this sort of crunch using the unregistered Class A 10.0.0.0 would take either a monstrously huge network or terrible inefficiency.

Suppose you allocate an OSPF area's address range to a set of user VLANs. Suppose you have selected a standard netmask for all such subnets, but you also need to decide how to allocate these addresses from the larger range. This allocation is largely arbitrary, but it is useful to have a common standard for how to do it. The usual way to do this is to divide the area range into different groups according to netmask. For example, suppose the area range has a mask of 255.255.0.0 and that three different types of masks are in use—255.255.255.0 (24 bits), 255.255.224.0 (27 bits), and 255.255.252.0 (30 bits).

The range consists of 255 Class C–sized units. The first mask size uses one of these units for every subnet. The second one allows you to fit up to 8 subnets into each unit. You can also fit 64 of the smallest subnets into each unit. Then work out how many of each type of subnet you expect to require.

The smallest-sized subnets actually have two uses. It is useful to assign a unique internal loopback IP address to every router. Some networks use a mask of 255.255.255.255 for this purpose, but the rules hand over one entire Class C–sized group of addresses for these addresses. There never should be more than about 50 devices in any OSPF area. Since 64 30-bit subnets are in one of these groups, and since keeping the number of different masks to a minimum is a good idea, it makes sense to use a mask of 255.255.255.252 for these loopback addresses. You then need to see how many real point-to-point subnets are needed. This step requires a better idea of the network topology. The rules should be as general as possible. In effect, I am talking about the worst cases, so I can follow the rule no matter how much the future surprises me with new technology.

I might want to say that up to 50 routers will be in an OSPF area and perhaps 3 point-to-point circuits on each one. This tells me to set aside the first 4 Class C-sized groups for 30-bit subnets. Then I need to figure out how many 27-bit subnets I will require. I can fit 8 of these subnets into one Class C–sized group, so if I think that 64 of these will be enough, then perhaps I can reserve the next 8 groups. And this will leave the remaining 242 groups for 24-bit subnets.

Note that throughout these arguments I made an effort to break up the groups along bit-mask lines. I could have said that I wanted 5 groups of 30-bit subnets, but I chose 4 groups to keep the subgroups aligned in sets that the network can easily summarize with another netmask. I did this not because I have any foreseeable need to do

so, but because one day I might have to break up an area into parts. If that happens, I want to make things as easy to split up as possible. At the same time, I don't want to make life more complicated if that split is not required.

You could make up a scheme where, for example, every sixteenth group contains 30-bit subnets and the next two are used for 27-bit subnets. This scheme would work, and it might make subdividing the area somewhat easier. However, subdividing an area will be hard no matter what you do, so it's more important to make everyday life easier.

Finally, the network designer needs to have standards for how she uses the addresses within a subnet. This depends not only on the subnet mask, but also on its use. Once again, if a small number of generally applicable rules can be made up, then trouble-shooting a problem at 3 A.M. will be much easier.

A common example of this sort of rule involves the default gateway for the subnet. Most network designers like to make the first address in the subnet belong to the main router to get off this subnet. For the 24-bit subnet (`255.255.255.0`) `10.1.2.0/24`, this address would be `10.1.2.1`. In a subnet that uses HSRP or VRRP, this default gateway address would be the virtual or standby address. The real router interfaces would then have the next two addresses, `10.1.2.2` and `10.2.2.3`, respectively. Many designers like to reserve a block of addresses at the start of the range just for network devices.

In a 30-bit point-to-point subnet (`255.255.255.252`) such as `10.1.2.4/30`, only two addresses are available, `10.1.2.5` and `10.1.2.6`. Devising a general rule for deciding which device gets the lower number is useful. I like to use the same rule mentioned earlier and make the lower number the address of the device leading to the Core. If this address is used to connect to a remote branch, then the remote side gets `10.1.2.6` and the head-office side will get `10.1.2.5`. Sometimes this link might be a connection between two remote sites or two Core devices. In this case, which router gets the lower number becomes arbitrary. In the case of point-to-point links between a router and a host, the router gets the lower number.

Establishing specific rules for how the addresses are allocated can be useful for any VLAN. Many organizations have rules so specific that it is possible to tell from just looking at the IP address whether the device in question is a user workstation, a server, a printer, or a network device. This knowledge can greatly simplify trouble-shooting and implementation.

Flexibility for future requirements

So far, I have tried to encourage designers to leave extra space. You need extra addresses in each subnet, extra subnets in each area, and extra room for more areas in the network. Network growth is driven purely by business factors that are largely unpredictable or, at least, unknown to the network designer.

One of the most profound challenges that a network designer can face is the acquisition of another company. This situation usually involves merging two networks that, in all likelihood, share address space and use conflicting standards. Even if this doesn't happen, healthy organizations tend to grow over time, which means that their networks must have growth capacity. A good IP addressing strategy always involves carefully overestimating the requirements, just in case.

The Default Gateway Question

The default gateway on any subnet is the IP address of the router that gets you off the segment. In fact, many routers may exist on a subnet. These routers may all lead to different destinations, but the default gateway is the one that you send a packet to when you don't know which one of the other routers can handle it.

In hierarchical network architectures, it is not common to put several routers on a segment. In this sort of design, it is generally best to use two routers and HSRP or VRRP for redundancy instead. But in a general network it is possible to have multiple routers all leading to different destinations.

The end devices need to have some sort of local routing table. In its simplest form, this routing table says two things. First, it directs all packets destined for the local subnet to just use the network interface card. Second, it contains a route for the default gateway, often expressed as a route to the IP address `0.0.0.0`, with a mask of `0.0.0.0`. This default gateway route is traditionally handled in one of two ways. Either it points to the local router, or it simply directs everything to use its own LAN interface without specifying a next hop. This second option requires the local router to act as an ARP proxy device for the remote networks it can route to. When the end station wants to send the packet to the unknown network, it first sends out an ARP packet for the destination device. That device is not actually on the local LAN segment, so it cannot respond to this ARP, but the router that knows how to get there responds for it. In proxy ARP, the router responds to the ARP packet with its own MAC address. The end device then communicates directly with the router at Layer 2 and the packets are routed normally.

At one time, this second option was the only way to reliably give a LAN segment router redundancy. If one router for the segment died, the second one would simply take over the traffic. Both would be configured for proxy ARP and both would handle live traffic all the time under normal operating conditions.

There are two problems with this strategy. The first is that every ARP query is a broadcast. Even in a fully switched VLAN architecture, every time a device wants to communicate outside of its subnet, it must disturb every other device in the VLAN. This disturbance is unlikely to cause large traffic overhead, but it is nonetheless inefficient. Furthermore, because it must ARP for every off-segment host separately, there is a short additional latency in every call setup.

The second problem is more serious. Most end devices use a simple ARP cache system that allows only one MAC address to be mapped to each destination IP address. If one router fails, the second will not be able to take over. Rather, the end device will continue trying the first router until the ARP cache entry times out. This timeout period is typically at least 5 minutes and often as long as 20. Clearly this time is not good enough if the network actually requires a robust fault-recovery system. But a shorter time is clearly inefficient.

This proxy ARP approach does give a convenient way to build IP-level fault tolerance for a LAN segment. However, the advent of VRRP and HSRP provides a much quicker and more efficient way of achieving the same result. In a hierarchical LAN design, the best high-availability topology involves two routers running VRRP or HSRP. Every end device on the subnet then treats the virtual address shared by these two routers as the default gateway.

DNS and DHCP

Two important IP-related applications are used in most large-scale LANs. Domain Name Service (DNS) is an application that provides mapping between host names and the corresponding IP addresses. Dynamic Host Configuration Protocol (DHCP) is a facility that provides a way to dynamically configure IP devices when they connect to the network.

The DNS client lookup procedure is built into just about every operating system. This procedure allows you to connect to your favourite web site by its name rather than having to remember the IP address. When you look up an arbitrary address on the Internet, your computer sends out a query to a preconfigured DNS server IP address. This query asks the DNS server to convert (or resolve) this name into an address. Once your local computer has the information it requires, it stores it so it won't need to ask again.

The greatest advantage to using DNS this way is not that it allows a human to remember the name rather than the address, although this feature is convenient. Rather, it is important because it allows the administrator to change IP addresses with relative ease. For example, if it is necessary to take the server offline and replace it with another, the administrator can simply set DNS to map the same name to a new address.

For efficiency, the end devices that request this name-to-address conversion generally store it in a local cache. Thus, DNS provides the ability to associate a maximum age for a particular address. For example, when the DNS server responds with the IP address, it may also tell the client device that it should only remember this address for five minutes. Once this period is over, if the client connects to this device again, it needs to do another lookup.

The naming system used by DNS is hierarchical. The various segments are separated by dots. For example, there might be a web server named *www.oreilly.com*. In this case, the top-level domain is *.com*. There are several top-level domains such as *.org*, *.net*, and *.gov*, as well as country-specific codes such as *.us*, *.uk*, and *.ca*.

The next field to the left defines the organizational domain name. There can be many hosts within that organization, one of which is called "www". In fact, DNS allows the administrators of the local domain to define more layers of hierarchy.

DHCP is a protocol that makes it possible to automatically configure end devices. The most common things to include in this configuration are the device's IP address, mask, and default gateway. It is also possible to configure several other kinds of information, such as the addresses of the DNS servers, time servers, database, or application servers. Indeed, the protocol itself is highly flexible and makes it possible (in theory) to configure just about anything the end device might need.

The interesting thing about DHCP is that, in the common example of setting up an IP address, the client device doesn't have enough information to get onto the network when it starts out. It doesn't have an IP address and it doesn't know where its default gateway is. In general, it doesn't even know the IP address of the DHCP server. All this device can really do is send out an all-hosts broadcast to `255.255.255.255` looking for a DHCP server. For its source address, it has to use the generic source address `0.0.0.0`, because it doesn't know anything else to use. In the default situation, the router for this network segment refuses to pass along a broadcast of this type. You either need to have a DHCP server for every segment or the router needs to cooperate.

Usually, the router is set up to automatically forward these broadcasts to a particular IP address, which will be the DHCP server. This IP address could be somewhere very far away in the network, perhaps several hops away. The router has to not only forward the client's request packet through the network to the server, but it also has to be able to receive the configuration information from the server so it can pass it along to the client. Most routers have the ability to pass this information along. On Cisco routers, doing this requires the use of the IP Helper Address feature.