172.65.48.120 to binary, first convert 172; then convert 65; then convert 48; and finally, convert 120.

- The Programmer Calculator has several features that are designed specifically for binary calculations, such as AND, OR, XOR, and so on.
- The Calculator program in Windows versions prior to Windows 7 does not have the programmer mode. However, those versions do offer a scientific mode that provides binary, hexadecimal, and decimal conversions.

# Introducing IP Addresses

An *IP address* is a number that uniquely identifies every host on an IP network. IP addresses operate at the network layer of the TCP/IP protocol stack, so they are independent of lower-level data link layer MAC addresses, such as Ethernet MAC addresses.

IP addresses are 32-bit binary numbers, which means that theoretically, a maximum of something in the neighborhood of 4 billion unique host addresses can exist throughout the Internet. You'd think that would be enough, but TCP/IP places certain restrictions on how IP addresses are allocated. These restrictions severely limit the total number of usable IP addresses. Many experts predict that we will run out of IP addresses soon. However, new techniques for working with IP addresses have helped to alleviate this problem, and a standard for 128-bit IP addresses has been adopted, though it still is not yet in widespread use.

## Networks and hosts

IP stands for *Internet Protocol*, and its primary purpose is to enable communications between networks. As a result, a 32-bit IP address actually consists of two parts:

- **The network ID (or network address):** Identifies the network on which a host computer can be found
- **The host ID (or host address):** Identifies a specific device on the network indicated by the network ID

Most of the complexity of working with IP addresses has to do with figuring out which part of the complete 32-bit IP address is the network ID and which part is the host ID, as described in the following sections.


As I describe the details of how host IDs are assigned, you may notice that two host addresses seem to be unaccounted for. For example, the Class C addressing scheme, which uses eight bits for the host ID, allows only 254 hosts — not the 256 hosts you'd expect. The host ID can't be 0 (the host ID is all zeros) because that address is always reserved to represent the network itself. And the host ID can't be 255 (the host ID is all ones) because that host ID is reserved for use as a broadcast request that's intended for all hosts on the network.

## The dotted-decimal dance

IP addresses are usually represented in a format known as *dotted-decimal notation.* In dotted-decimal notation, each group of eight bits — an *octet* — is represented by its decimal equivalent. For example, consider the following binary IP address:

```
11000000101010001000100000011100
```

To convert this value to dotted-decimal notation, first divide it into four octets, as follows:

```
11000000 10101000 10001000 00011100
```

Then, convert each of the octets to its decimal equivalent:

```
11000000 10101000 10001000 00011100
192 168 136 28
```

Then, use periods to separate the four decimal numbers, like this:

```
192.168.136.28
```

This is the format in which you'll usually see IP addresses represented.

Figure 3-2 shows how the 32 bits of an IP address are broken down into four octets of eight bits each. As you can see, the four octets of an IP address are often referred to as *w, x, y,* and *z.*

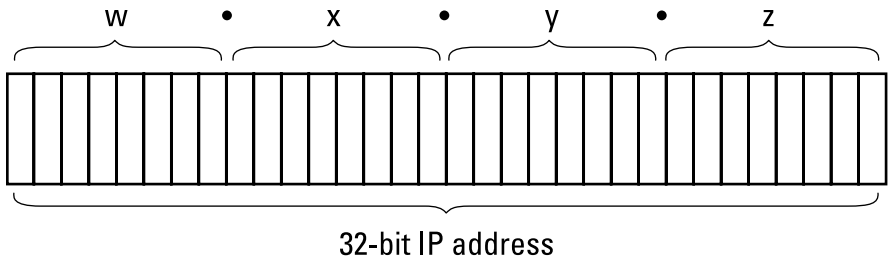


**FIGURE 3-2:** Octets and dotted-decimal notation.

# Classifying IP Addresses

When the original designers of the IP protocol created the IP addressing scheme, they could have assigned an arbitrary number of IP address bits for the network ID. The remaining bits would then be used for the host ID. For example, suppose that the designers decided that half of the address (16 bits) would be used for the network, and the remaining 16 bits would be used for the host ID. The result of that scheme would be that the Internet could have a total of 65,536 networks, and each of those networks could have 65,536 hosts.

In the early days of the Internet, this scheme probably seemed like several orders of magnitude more than would ever be needed. However, the IP designers realized from the start that few networks would actually have tens of thousands of hosts. Suppose that a network of 1,000 computers joins the Internet and is assigned one of these hypothetical network IDs. Because that network will use only 1,000 of its 65,536 host addresses, more than 64,000 IP addresses would be wasted.

As a solution to this problem, the idea of IP address classes was introduced. The IP protocol defines five different address classes: A, B, C, D, and E. Each of the first three classes, A–C, uses a different size for the network ID and host ID portion of the address. Class D is for a special type of address called a *multicast address.* Class E is an experimental address class that isn't used.

The first four bits of the IP address are used to determine into which class a particular address fits, as follows:

- **Class A:** The first bit is zero.
- **Class B:** The first bit is one, and the second bit is zero.
- **Class C:** The first two bits are both one, and the third bit is zero.
- **Class D:** The first three bits are all one, and the fourth bit is zero.
- **Class E:** The first four bits are all one.

Because Class D and E addresses are reserved for special purposes, I focus the rest of the discussion here on Class A, B, and C addresses. Table 3-3 summarizes the details of each address class.

**TABLE 3-3** IP Address Classes

| Class | Address Number Range | Starting Bits | Length of Network ID | Number of Networks | Hosts |
|---|---|---|---|---|---|
| A | 1–126.*x.y.z* | 0 | 8 | 126 | 16,777,214 |
| B | 128–191.*x.y.z* | 10 | 16 | 16,384 | 65,534 |
| C | 192–223.*x.y.z* | 110 | 24 | 2,097,152 | 254 |

## Class A addresses

Class A addresses are designed for very large networks. In a Class A address, the first octet of the address is the network ID, and the remaining three octets are the host ID. Because only eight bits are allocated to the network ID and the first of these bits is used to indicate that the address is a Class A address, only 126 Class A networks can exist in the entire Internet. However, each Class A network can accommodate more than 16 million hosts.

Only about 40 Class A addresses are actually assigned to companies or organizations. The rest are either reserved for use by the Internet Assigned Numbers Authority (IANA) or are assigned to organizations that manage IP assignments for geographic regions such as Europe, Asia, and Latin America.

Just for fun, Table 3-4 lists some of the better-known Class A networks (as of January 2018 — these change frequently). You'll probably recognize many of them. In case you're interested, you can find a complete list of all the Class A address assignments at `www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml`.

You may have noticed in Table 3-3 that Class A addresses end with `126.x.y.z`, and Class B addresses begin with `128.x.y.z`. What happened to `127.x.y.z`? This special range of addresses is reserved for loopback testing, so these addresses aren't assigned to public networks.

**TABLE 3-4** Some Well-Known Class A Networks

| Network | Description | Network | Description |
|---|---|---|---|
| 3 | General Electric Company | | |
| 4 | Level 3 Communications | 22 | Defense Information Systems Agency |
| 6 | Army Information Systems Center | 25 | UK Ministry of Defense |
| 8 | Level-3 Communications | 26 | Defense Information Systems Agency |
| 9 | IBM | 28 | Decision Sciences Institute (North) |
| 11 | DoD Intel Information Systems | 29–30 | Defense Information Systems Agency |
| 12 | AT&T Bell Laboratories | 33 | DLA Systems Automation Center |
| 15 | Hewlett-Packard Company | 44 | 44Amateur Radio Digital Communications |
| 16 | Digital Equipment Corporation | 48 | Prudential Securities, Inc. |
| 17 | Apple Computer, Inc. | 53 | Daimler |
| 18 | MIT | 55 | DoD Network Information Center |
| 19 | Ford Motor Company | 56 | U.S. Postal Service |
| 20 | Computer Sciences Corporation | | |

## Class B addresses

In a Class B address, the first two octets of the IP address are used as the network ID, and the second two octets are used as the host ID. Thus, a Class B address comes close to my hypothetical scheme of splitting the address down the middle, using half for the network ID and half for the host ID. It isn't identical to this scheme, however, because the first two bits of the first octet are required to be 10, in order to indicate that the address is a Class B address. As a result, a total of 16,384 Class B networks can exist. All Class B addresses fall within the range `128.`*`x`*`.`*`y`*`.`*`z`* to `191.`*`x`*`.`*`y`*`.`*`z`*. Each Class B address can accommodate more than 65,000 hosts.

The problem with Class B networks is that even though they are much smaller than Class A networks, they still allocate far too many host IDs. Very few networks have tens of thousands of hosts. Thus, careless assignment of Class B addresses can lead to a large percentage of the available host addresses being wasted on organizations that don't need them.

## Class C addresses

In a Class C address, the first three octets are used for the network ID, and the fourth octet is used for the host ID. With only eight bits for the host ID, each Class C network can accommodate only 254 hosts. However, with 24 network ID bits, Class C addresses allow for more than 2 million networks.

The problem with Class C networks is that they're too small. Although few organizations need the tens of thousands of host addresses provided by a Class B address, many organizations need more than a few hundred. The large discrepancy between Class B networks and Class C networks is what led to the development of *subnetting*, which I describe in the next section.

# Subnetting

*Subnetting* is a technique that lets network administrators use the 32 bits available in an IP address more efficiently by creating networks that aren't limited to the scales provided by Class A, B, and C IP addresses. With subnetting, you can create networks with more realistic host limits.

Subnetting provides a more flexible way to designate which portion of an IP address represents the network ID and which portion represents the host ID. With standard IP address classes, only three possible network ID sizes exist: 8 bits for Class A, 16 bits for Class B, and 24 bits for Class C. Subnetting lets you select an arbitrary number of bits to use for the network ID.

Two reasons compel people to use subnetting. The first is to allocate the limited IP address space more efficiently. If the Internet were limited to Class A, B, or C addresses, every network would be allocated 254, 64 thousand, or 16 million IP addresses for host devices. Although many networks with more than 254 devices exist, few (if any) exist with 64 thousand, let alone 16 million. Unfortunately, any network with more than 254 devices would need a Class B allocation and probably waste tens of thousands of IP addresses.

The second reason for subnetting is that even if a single organization has thousands of network devices, operating all those devices with the same network ID would slow the network to a crawl. The way TCP/IP works dictates that all the computers with the same network ID must be on the same physical network. The physical network comprises a single *broadcast domain*, which means that a single network medium must carry all the traffic for the network. For performance reasons, networks are usually segmented into broadcast domains that are smaller than even Class C addresses provide.

# Subnets

A *subnet* is a network that falls within a Class A, B, or C network. Subnets are created by using one or more of the Class A, B, or C host bits to extend the network ID. Thus, instead of the standard 8-, 16-, or 24-bit network ID, subnets can have network IDs of any length.

Figure 3-3 shows an example of a network before and after subnetting has been applied. In the unsubnetted network, the network has been assigned the Class B address `144.28.0.0`. All the devices on this network must share the same broadcast domain.

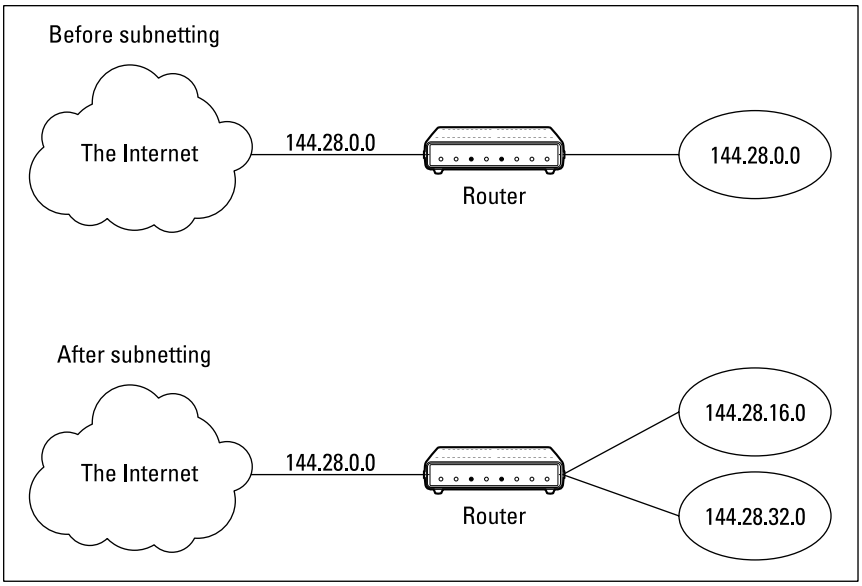


**FIGURE 3-3:** A network before and after subnetting.

In the second network, the first four bits of the host ID are used to divide the network into two small networks, identified as subnets `16` and `32`. To the outside world (that is, on the other side of the router), these two networks still appear to be a single network identified as `144.28.0.0`. For example, the outside world considers the device at `144.28.16.22` to belong to the `144.28.0.0` network. As a result, a packet sent to this device will be delivered to the router at `144.28.0.0`. The router then considers the subnet portion of the host ID to decide whether to route the packet to subnet `16` or subnet `32`.

# Subnet masks

For subnetting to work, the router must be told which portion of the host ID should be used for the subnet network ID. This little sleight of hand is accomplished by

using another 32-bit number, known as a *subnet mask.* Those IP address bits that represent the network ID are represented by a 1 in the mask, and those bits that represent the host ID appear as a 0 in the mask. As a result, a subnet mask always has a consecutive string of ones on the left, followed by a string of zeros.

For example, the subnet mask for the subnet shown in Figure 3-3, where the network ID consists of the 16-bit network ID plus an additional 4-bit subnet ID, would look like this:

```
11111111 11111111 11110000 00000000
```

In other words, the first 20 bits are ones, and the remaining 12 bits are zeros. Thus, the complete network ID is 20 bits in length, and the actual host ID portion of the subnetted address is 12 bits in length.

To determine the network ID of an IP address, the router must have both the IP address and the subnet mask. The router then performs a bitwise operation called a *logical AND* on the IP address in order to extract the network ID. To perform a logical AND, each bit in the IP address is compared with the corresponding bit in the subnet mask. If both bits are 1, the resulting bit in the network ID is set to 1. If either of the bits are 0, the resulting bit is set to 0.

For example, here's how the network address is extracted from an IP address using the 20-bit subnet mask from the previous example:

```
144 . 28 . 16 . 17
IP address: 10010000 00011100 00010000 00010001
Subnet mask: 11111111 11111111 11110000 00000000
Network ID: 10010000 00011100 00010000 00000000
 144  .  28 . 16 . 0
```

Thus, the network ID for this subnet is `144.28.16.0`.

The subnet mask itself is usually represented in dotted-decimal notation. As a result, the 20-bit subnet mask used in the previous example would be represented as `255.255.240.0`:

```
Subnet mask: 11111111 11111111 11110000 00000000
 255 . 255 . 240 .  0
```


TIP

Don't confuse a subnet mask with an IP address. A subnet mask doesn't represent any device or network on the Internet. It's just a way of indicating which portion of an IP address should be used to determine the network ID. (You can spot a subnet mask right away because the first octet is always `255`, and `255` is not a valid first octet for any class of IP address.)

## Network prefix notation

Because a subnet mask always begins with a consecutive sequence of ones to indicate which bits to use for the network ID, you can use a shorthand notation — a *network prefix* — to indicate how many bits of an IP address represent the network ID. The network prefix is indicated with a slash immediately after the IP address, followed by the number of network ID bits to use. For example, the IP address `144.28.16.17` with the subnet mask `255.255.240.0` can be represented as `144.28.16.17/20` because the subnet mask `255.255.240.0` has 20 network ID bits.

Network prefix notation is also called *classless interdomain routing* notation (*CIDR*, for short) because it provides a way of indicating which portion of an address is the network ID and which is the host ID without relying on standard address classes.

## Default subnets

The *default subnet masks* are three subnet masks that correspond to the standard Class A, B, and C address assignments. These default masks are summarized in Table 3-5.

**TABLE 3-5** The Default Subnet Masks

| Class | Binary | Dotted-Decimal | Network Prefix |
|---|---|---|---|
| A | `11111111 00000000 00000000 00000000` | `255.0.0.0` | `/8` |
| B | `11111111 11111111 00000000 00000000` | `255.255.0.0` | `/16` |
| C | `11111111 11111111 11111111 00000000` | `255.255.255.0` | `/24` |


TIP

Keep in mind that a subnet mask is not actually required to use one of these defaults because the IP address class can be determined by examining the first three bits of the IP address. If the first bit is 0, the address is Class A, and the subnet mask `255.0.0` is applied. If the first two bits are 10, the address is Class B, and `255.255.0.0` is used. If the first three bits are 110, the Class C default mask `255.255.255.0` is used.

## The great subnet roundup

You should know about a few additional restrictions that are placed on subnets and subnet masks. In particular

- **The minimum number of network ID bits is eight.** As a result, the first octet of a subnet mask is always 255.

- **The maximum number of network ID bits is 30.** You have to leave at least two bits for the host ID portion of the address to allow for at least two hosts. If you use all 32 bits for the network ID, that leaves no bits for the host ID. Obviously, that won't work. Leaving just one bit for the host ID won't work, either, because a host ID of all ones is reserved for a broadcast address, and all zeros refers to the network itself. Thus, if you use 31 bits for the network ID and leave only 1 for the host ID, host ID `1` would be used for the broadcast address, and host ID `0` would be the network itself, leaving no room for actual hosts. That's why the maximum network ID size is 30 bits.
- **Because the network ID portion of a subnet mask is always composed of consecutive bits set to 1, only eight values are possible for each octet of a subnet mask:** `0`, `128`, `192`, `224`, `248`, `252`, `254`, and `255`.
- **A subnet address can't be all zeros or all ones.** Thus, the number of unique subnet addresses is two less than two raised to the number of subnet address bits. For example, with three subnet address bits, six unique subnet addresses are possible ($2^3 - 2 = 6$). This implies that you must have at least two subnet bits. (If a single-bit subnet mask were allowed, it would violate the "can't be all zeros or all ones" rule because the only two allowed values would be 0 or 1.)

# IP block parties

A subnet can be thought of as a range or block of IP addresses that have a common network ID. For example, the CIDR `192.168.1.0/28` represents the following block of 14 IP addresses:

```
192.168.1.1 192.168.1.2 192.168.1.3 192.168.1.4
192.168.1.5 192.168.1.6 192.168.1.7 192.168.1.8
192.168.1.9 192.168.1.10 192.168.1.11 192.168.1.12
192.168.1.13 192.168.1.14
```

Given an IP address in CIDR notation, it's useful to be able to determine the range of actual IP addresses that the CIDR represents. This matter is straightforward when the octet within which the network ID mask ends happens to be `0`, as in the preceding example. You just determine how many host IDs are allowed based on the size of the network ID and count them off.

However, what if the octet where the network ID mask ends is not `0`? For example, what are the valid IP addresses for `192.168.1.100` when the subnet mask is `255.255.255.240`? In that case, the calculation is a little harder. The first step is to determine the actual network ID. You can do that by converting both the IP address and the subnet mask to binary and then extracting the network ID as in this example:

```
  192 . 168 . 1 .  100
IP address: 11000000 10101000 00000001 01100100
Subnet mask: 11111111 11111111 11111111 11110000
Network ID: 11000000 10101000 00000001 01100000
 192 . 168 . 1 .  96
```

As a result, the network ID is `192.168.1.96`.

Next, determine the number of allowable hosts in the subnet based on the network prefix. You can calculate this by subtracting the last octet of the subnet mask from 254. In this case, the number of allowable hosts is 14.

To determine the first IP address in the block, add 1 to the network ID. Thus, the first IP address in my example is `192.168.1.97`. To determine the last IP address in the block, add the number of hosts to the network ID. In my example, the last IP address is `192.168.1.110`. As a result, the `192.168.1.100` with subnet mask `255.255.255.240` designates the following block of IP addresses:

```
192.168.1.97 192.168.1.98 192.168.1.99 192.168.1.100
192.168.1.101 192.168.1.102 192.168.1.103 192.168.1.104
192.168.1.105 192.168.1.106 192.168.1.107 192.168.1.108
192.168.1.109 192.168.1.110
```

# Private and public addresses

Any host with a direct connection to the Internet must have a globally unique IP address. However, not all hosts are connected directly to the Internet. Some are on networks that aren't connected to the Internet. Some hosts are hidden behind firewalls, so their Internet connection is indirect.

Several blocks of IP addresses are set aside just for this purpose, for use on private networks that are not connected to the Internet or to use on networks that are hidden behind a firewall. Three such ranges of addresses exist, summarized in Table 3-6. Whenever you create a private TCP/IP network, you should use IP addresses from one of these ranges.

**TABLE 3-6** Private Address Spaces

| CIDR | Subnet Mask | Address Range |
|---|---|---|
| 10.0.0.0/8 | 255.0.0.0 | 10.0.0.1–10.255.255.254 |
| 172.16.0.0/12 | 255.240.0.0 | 172.16.1.1–172.31.255.254 |
| 192.168.0.0/16 | 255.255.0.0 | 192.168.0.1–192.168.255.254 |

# Network Address Translation

Many firewalls use a technique called *network address translation* (NAT) to hide the actual IP address of a host from the outside world. When that's the case, the NAT device must use a globally unique IP to represent the host to the Internet. Behind the firewall, though, the host can use any IP address it wants. When packets cross the firewall, the NAT device translates the private IP address to the public IP address and vice versa.

One of the benefits of NAT is that it helps to slow down the rate at which the IP address space is assigned. That's because a NAT device can use a single public IP address for more than one host. It does so by keeping track of outgoing packets so that it can match incoming packets with the correct host. To understand how this works, consider the following sequence of steps:

1. A host whose private address is `192.168.1.100` sends a request to `216.239.57.99`, which as of January 2018 happens to be `www.google.com`. The NAT device changes the source IP address of the packet to `208.23.110.22`, the IP address of the firewall. That way, Google will send its reply back to the firewall router. The NAT records that `192.168.1.100` sent a request to `216.239.57.99`.
2. Now another host, at address `192.168.1.107`, sends a request to `207.46.134.190`, which happens to be `www.microsoft.com`. The NAT device changes the source of this request to `208.23.110.22` so that Microsoft will reply to the firewall router. The NAT records that `192.168.1.107` sent a request to `207.46.134.190`.
3. A few seconds later, the firewall receives a reply from `216.239.57.99`. The destination address in the reply is `208.23.110.22`, the address of the firewall. To determine to whom to forward the reply, the firewall checks its records to see who is waiting for a reply from `216.239.57.99`. It discovers that `192.168.1.100` is waiting for that reply, so it changes the destination address to `192.168.1.100` and sends the packet on.

Actually, the process is a little more complicated than that, because it's very likely that two or more users may have pending requests from the same public IP. In that case, the NAT device uses other techniques to figure out to which user each incoming packet should be delivered.