25

IPV6 ADDRESSING

The primary motivation for creating Internet Protocol version 6 (IPv6) was to rectify the addressing problems in version 4 (IPv4). Along with acquiring more addresses, the IPv6 designers desired a way of interpreting, assigning, and using addresses in a way that was more consonant with modern internetworking. So, it's no surprise that many of the changes in IPv6 are associated with IP addressing. The IPv6 addressing scheme is similar in concept to IPv4 addressing, but has been completely overhauled to create an addressing system that's capable of supporting continued Internet expansion and new applications for the foreseeable future.

This chapter describes the concepts and methods associated with addressing under IPv6. I begin with a look at some addressing generalities in IPv6, including the addressing model, address types' size, and address space. I discuss the unique and sometimes confusing representations and notations used for IPv6 addresses and prefixes. Then I look at how addresses are arranged and allocated into types, beginning with an overall look at address space composition and then at the global unicast address format. I describe the new methods used for mapping IP addresses to underlying physical network addresses. I then describe special IPv6 addressing issues, including reserved and private addresses, IPv4 address embedding, anycast and multicast addresses, and autoconfiguration and renumbering of addresses.

Addressing under IPv6 is outlined in the main IPv6 RFC, RFC 2460, "Internet Protocol, Version 6 (IPv6) Specification." However, most of the details of IPv6 addressing are contained in two other standards: RFC 3513, "Internet Protocol Version 6 (IPv6) Addressing Architecture," and RFC 3587, "IPv6 Global Unicast Address Format." These replaced the 1998 standards RFC 2373, "IP Version 6 Addressing Architecture," and RFC 2374, "An IPv6 Aggregatable Global Unicast Address Format."

BACKGROUND INFORMATION *As with the other IPv6 chapters in this book, my look at addressing is based somewhat on a contrast to how addressing is done in IPv4. I strongly recommend a thorough understanding of IPv4 addressing, including classless addressing using Classless Inter-Domain Routing (CIDR), as presented in Chapters 16 through 23, before proceeding here. As with the IPv4 addressing sections, familiarity with how binary numbers work, and conversion between binary and decimal numbers is also a good idea. Chapter 4, which provides some background on data representation and the mathematics of computing, may be of assistance in that respect.*

IPv6 Addressing Overview: Addressing Model, Address Types, and Address Size

As you saw in the previous chapter, IPv6 represents a significant update to IP, but its modifications and additions are made without changing the core nature of how IP works. Addressing is the place where most of the differences between IPv4 and IPv6 are seen, but the changes are mostly in how addresses are implemented and used. The overall model used for IP addressing in IPv6 is pretty much the same as it was in IPv4; some aspects have not changed at all, while others have changed only slightly.

IPv6 Addressing Model Characteristics

Here are some of the general characteristics of the IPv6 addressing model that are basically the same as in IPv4:

Core Functions of Addressing The two main functions of addressing are still network interface identification and routing. Routing is facilitated through the structure of addresses on the internetwork.

Network Layer Addressing IPv6 addresses are still the ones associated with the network layer in TCP/IP networks and are distinct from data link layer (also sometimes called *physical*) addresses.

Number of IP Addresses per Device Addresses are still assigned to network interfaces, so a regular host like a PC will usually have one (unicast) address, and routers will have more than one for each of the physical networks to which it connects.

Address Interpretation and Prefix Representation IPv6 addresses are like classless IPv4 addresses in that they are interpreted as having a network identifier part and a host identifier part (a network ID and a host ID), but that the delineation is not encoded into the address itself. A prefix-length number, using CIDR-like notation, is used to indicate the length of the network ID (prefix length).

Private and Public Addresses Both types of addresses exist in IPv6, though they are defined and used somewhat differently.

IPv6 Supported Address Types

One important change in the addressing model of IPv6 is the *address types* supported. IPv4 supported three address types: unicast, multicast, and broadcast. Of these, the vast majority of actual traffic was unicast. IP multicast support was not widely deployed until many years after the Internet was established and it continues to be hampered by various issues. Use of broadcast in IP had to be severely restricted for performance reasons (we don't want any device to be able to broadcast across the entire Internet!).

IPv6 also supports three address types, but with the following changes:

Unicast Addresses These are standard unicast addresses as in IPv4, one per host interface.

Multicast Addresses These are addresses that represent various groups of IP devices. A message sent to a multicast address goes to all devices in the group. IPv6 includes much better multicast features and many more multicast addresses than IPv4. Since multicast under IPv4 was hampered in large part due to lack of support of the feature by many hardware devices, support for multicasting is a required, not optional, part of IPv6.

Anycast Addresses Anycast addressing is used when a message must be sent to any member of a group, but does not need to be sent to all of them. Usually the member of the group that is easiest to reach will be sent the message. One common example of how anycast addressing could be used is in load sharing among a group of routers in an organization.

Broadcast addressing as a distinct addressing method is gone in IPv6. Broadcast functionality is implemented using multicast addressing to groups of devices. A multicast group to which all nodes belong can be used for broadcasting in a network, for example.

KEY CONCEPT IPv6 has unicast and multicast addresses like IPv4. There is, however, no distinct concept of a broadcast address in IPv6. A new type of address, the *anycast* address, has been added to allow a message to be sent to any one member of a group of devices.

An important implication of the creation of anycast addressing is removal of the strict uniqueness requirement for IP addresses. Anycast is accomplished by assigning the same IP address to more than one device. The devices must also be specifically told that they are sharing an anycast address, but the addresses themselves are structurally the same as unicast addresses.

The bulk of the remainder of this chapter focuses on unicast addressing, since it is by far the most important type. Multicast and anycast addressing are given special attention in a separate section later in this chapter.

IPv6 Address Size and Address Space

Of all the changes introduced in IPv6, easily the most celebrated is the increase in the size of IP addresses, which resulted in a corresponding massive increase in the size of the address space as well. It's not surprising that these sizes were increased compared to IPv4—everyone has known for years that the IPv4 address space was too small to support the future of the Internet. What's remarkable is the level of increase and the implications for how Internet addresses are used.

In IPv4, IP addresses are 32 bits long; these are usually grouped into 4 octets of 8 bits each. The theoretical IPv4 address space is 2^{32}, or 4,294,967,296 addresses. To increase this address space, we simply increase the size of addresses; each extra bit we give to the address size doubles the address space. Based on this, some folks expected the IPv6 address size to increase from 32 to 48 bits, or perhaps 64 bits. Either of these numbers would have given a rather large number of addresses.

However, IPv6 addressing doesn't use either of these figures. Instead, the IP address size jumps all the way to 128 bits, or 16 8-bit octets/bytes. The size of the IPv6 address space is, quite literally, astronomical. Like the numbers that describe the number of stars in a galaxy or the distance to the furthest pulsars, the number of addresses that can be supported in IPv6 is mind-boggling. See Figure 25-1 for an idea of what I mean by *astronomical*.

Since IPv6 addresses are 128 bits long, the theoretical address space, if all addresses were used, is 2^{128} addresses. This number, when expanded out, is 340,282,366,920,938,463,463,374,607,431,768,211,456, which is normally expressed in scientific notation as about $3.4*10^{38}$ addresses. Whoa! That's about 340 trillion, *trillion, trillion* addresses. As I said, it's pretty hard to grasp just how large this number is. Consider these comparisons:

- It's enough addresses for many trillions of addresses to be assigned to every human being on the planet.
- The Earth is about 4.5 billion years old. If you had been assigning IPv6 addresses at a rate of 1 billion per second since the Earth was formed, you would have by now used up less than one trillionth of the address space.
- The Earth's surface area is about 510 trillion square meters. If a typical computer has a footprint of about one-tenth of a square meter, you would have to stack computers 10 billion high—blanketing the entire surface of the Earth—to use up that same trillionth of the address space.

OK, I think you get the idea. It's clear that one goal of the decision to go to 128-bit addresses is to make sure that we will never run out of address space again, and it seems quite likely that this will be the case.

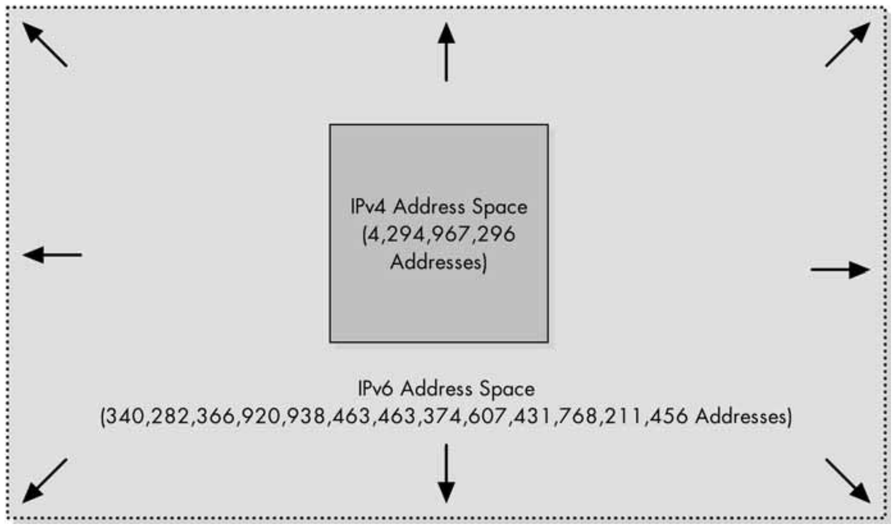
IPv4 Address Space
(4,294,967,296
Addresses)
IPv6 Address Space
(340,282,366,920,938,463,463,374,607,431,768,211,456 Addresses)

Figure 25-1: A (poor) representation of relative IPv4 and IPv6 address space sizes *I wanted to make a cool graphic to show the relative sizes of the IPv4 and IPv6 address spaces. You know, where I would show the IPv6 address space as a big box and the IPv4 address space as a tiny one. The problem is that the IPv6 address space is* so *much larger than the IPv4 space that there is no way to show it to scale! To make this diagram to scale, imagine the IPv4 address space is the 1.6-inch square above. In that case, the IPv6 address space would be represented by a square the size of the solar system!*

There are drawbacks to having such a huge address space, too. Consider that even with a 64-bit address, we would have a very large address space; 2^{64} equals 18,446,744,073,709,551,616, or about 18 million trillion. These are still probably more addresses than the Internet will ever need. However, by going to 128 bits instead, this has made dealing with IP addresses unruly (as you'll see in the next section). This has also increased overhead, since every datagram header or other place where IP addresses are referenced must use 16 bytes for each address instead of the 4 bytes that were needed in IPv4, or the 8 bytes that might have been required with a 64-bit address.

KEY CONCEPT The IPv6 address space is really, *really* big!

So why the overkill of going to 128 bits? The main reason is *flexibility*. Even though you can have a couple zillion addresses if we allocate them one at a time, this makes assignment difficult. The developers got rid of class-oriented addressing in IPv4 because it wasted address space. The reality, though, is that being able to waste address space is a useful luxury.

Having 128 bits allows us to divide the address space and assign various purposes to different bit ranges, while still not having to worry about running out of space. Later in this chapter, in the section describing the IPv6 global unicast address format, you'll see one way that those 128 bits are put to good use: They allow you to create a hierarchy of networks while still saving 64 bits for host IDs. This hierarchy has its own advantages.

IPv6 Address and Address Notation and Prefix Representation

Increasing the size of IP addresses from 32 bits to 128 bits expands the address space to a gargantuan size, thereby ensuring that we will never again run out of IP addresses, and thereby allowing flexibility in how they are assigned and used. Unfortunately, there are some drawbacks to this method, and one of them is that 128-bit numbers are very large. The size makes them awkward and difficult to use.

Computers work in binary, and they have no problem dealing with long strings of ones and zeros, but humans find them confusing. Even the 32-bit addresses of IPv4 are cumbersome for us to deal with, which is why we use dotted decimal notation for them unless we need to work in binary (as with subnetting). However, IPv6 addresses are so much larger than IPv4 addresses that it becomes problematic to use dotted decimal notation. To use this notation, we would split the 128 bits into 16 octets and represent each with a decimal number from 0 to 255. However, we would end up not with 4 of these numbers, but *16*. A typical IPv6 address in this notation would appear as follows:

128.91.45.157.220.40.0.0.0.0.252.87.212.200.31.255

The binary and dotted decimal representations of this address are shown near the top of Figure 25-2. In either case, the word *elegant* doesn't exactly spring to mind.

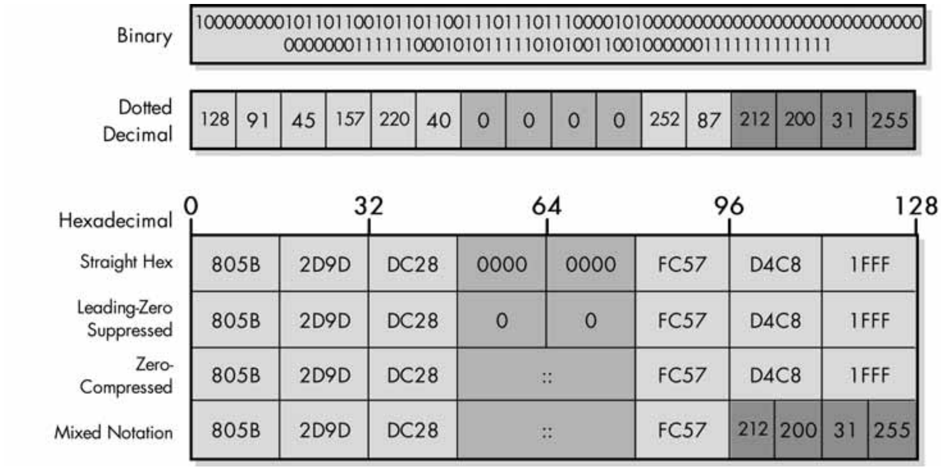

Figure 25-2: Binary, decimal, and hexadecimal representations of IPv6 addresses *The top two rows show binary and dotted decimal representations of an IPv6 address; neither is commonly used (other than by computers themselves!). The top row of the lower table shows the full hexadecimal representation, while the next two rows illustrate zero suppression and compression. The last row shows mixed notation, with the final 32 bits of an IPv6 address shown in dotted decimal notation (212.200.31.255). This is most commonly used for embedded IPv4 addresses.*

IPv6 Address Hexadecimal Notation

To make addresses shorter, the decision was made in IPv6 to change the primary method of expressing addresses to use hexadecimal instead of decimal. The advantage of this is that it requires fewer characters to represent an address, and converting from hexadecimal to binary and back again is much easier than converting from binary to decimal or vice versa. The disadvantage is that many people find hexadecimal difficult to comprehend and work with, especially because the notion of 16 values in each digit is a bit strange.

The hexadecimal notation used for IPv6 addresses is similar to the same method used for IEEE 802 MAC addresses, and for technologies like Ethernet. With these MAC addresses, 48 bits are represented by 6 octets, each octet being a hexadecimal number from 0 to FF, separated by a dash or colon, like this:

0A-A7-94-07-CB-D0

Since IPv6 addresses are larger, they are instead grouped into eight 16-bit *words*, separated by colons, to create what is sometimes called *colon hexadecimal notation*, as shown in Figure 25-2. So, the IPv6 address given in the previous example would be expressed as follows:

805B:2D9D:DC28:0000:0000:FC57:D4C8:1FFF

To keep the address size down, leading zeros can be suppressed in the notation so you can immediately reduce this to the following:

805B:2D9D:DC28:0:0:FC57:D4C8:1FFF

Well, it's definitely shorter than dotted decimal, but still pretty long. When you are dealing with numbers this big, there's only so much you can do. This is part of why the use of Domain Name System (DNS) names for hosts becomes much more important under IPv6 than it is in IPv4: Who could remember a hex address that long?

Zero Compression in IPv6 Addresses

Fortunately, there is a shortcut that can be applied to shorten some addresses even further. This technique is sometimes called *zero compression*. The method allows a single string of contiguous zeros in an IPv6 address to be replaced by double colons. So, for example, the previous address could be expressed as follows:

805B:2D9D:DC28::FC57:D4C8:1FFF

You know how many zeros are replaced by the two colons (::) because you can see how many fully expressed (uncompressed) hexadecimal words are in the address. In this case, there are six, so the :: represents two zero words. To prevent ambiguity, the double colons can appear only once in any IP address, because if it appeared more than once, you could not tell how many zeros were replaced in each instance. So, if the example address were 805B:2D9D:DC28:0:0:FC57:0:0, you could replace either the first pair of zeros or the second, but not both.

Zero compression doesn't make the example much shorter, but due to how IPv6 addresses are structured, long strings of zeros are common. For example, consider this address:

FF00:4501:0:0:0:0:0:32

With compression, this could be shortened as follows:

FF00:4501::32

The technique works even better on special addresses. The full IPv6 loopback address is written as follows:

0:0:0:0:0:0:0:1

With compression, the loopback address looks like this:

::1

For even more fun, consider the especially odd IPv6 unspecified address, as shown here:

0:0:0:0:0:0:0:0

Apply zero compression to an address that is all zeros, and what do you get?

::

No numbers at all! Of course, thinking of :: as an address *does* take some getting used to.

KEY CONCEPT For brevity, IPv6 addresses are represented using eight sets of four hexadecimal digits, a form called *colon hexadecimal notation*. Additional techniques, called *zero suppression* and *zero compression*, are used to reduce the size of displayed addresses further by removing unnecessary zeros from the presentation of the address.

IPv6 Mixed Notation

There is also an alternative notation used in some cases, especially for expressing IPv6 addresses that embed IPv4 addresses (discussed later in this chapter). For these, it is useful to show the IPv4 portion of the address in the older dotted decimal notation, since that's what you use for IPv4. Since embedding uses the last 32 bits for the IPv4 address, the notation has the first 96 bits in colon hexadecimal notation and the last 32 bits in dotted decimal. So, to take the earlier example again, in *mixed notation* it would be shown as follows:

805B:2D9D:DC28::FC57:**212.200.31.255**

This isn't really a great example of mixed notation, because embedding usually involves long strings of zeros followed by the IPv4 address. Thus, zero compression comes in very handy here. Instead of seeing something like this:

0:0:0:0:0:0:212.200.31.255

You will typically see this:

::212.200.31.255

At first glance, this appears to be an IPv4 address. You must keep a close eye on those colons in IPv6!

KEY CONCEPT A special mixed notation is defined for IPv6 addresses whose last 32 bits contain an embedded IPv4 address. In this notation, the first 96 bits are displayed in regular colon hexadecimal notation, and the last 32 bits are displayed in IPv4-style dotted decimal.

IPv6 Address Prefix Length Representation

Like IPv4 classless addresses, IPv6 addresses are fundamentally divided into a number of network ID bits followed by a number of host ID bits. The network identifier is called the *prefix*, and the number of bits used is the *prefix length*. This prefix is represented by adding a slash after the address and then putting the prefix length after the slash. This is the same method used for classless IPv4 addressing with CIDR. For example, if the first 48 bits of the sample address were the network ID (prefix), then we would express this as 805B:2D9D:DC28::FC57:D4C8:1FFF/48.

KEY CONCEPT In IPv6, the size of an address's prefix is indicated by the prefix length that follows the address, separated with a slash, just as it is done in IPv4 classless addressing.

As in IPv4, specifiers for whole networks will typically end in long strings of zeros. These can be replaced by double colons (::) using zero compression. For example, the 48-bit network ID for the previous example is 805B:2D9D:DC28:0:0:0:0:0/48, or 805B:2D9D:DC28::/48. You *must* include the "::" if replacing the trailing zeros.

IPv6 Address Space Allocation

After dealing for so many years with the very small IPv4 address space, the enormous number of addresses in IPv6 must have made the Internet Engineering Task Force (IETF) engineers feel like kids in a candy shop. They were good kids, however, and didn't run wild, grabbing all the candy they could find and gobbling it up. They very carefully considered how to divide the address space for various uses. Of course, when you have this much candy, sharing becomes pretty easy.

As was the case with IPv4, the two primary concerns in deciding how to divide the IPv6 address space were address assignment and routing. The designers of IPv6 wanted to structure the address space to make allocation of addresses to Internet service providers (ISPs), organizations, and individuals as easy as possible.

At first, perhaps ironically, this led the creators of IPv6 back full circle to the use of specific bit sequences to identify different types of addresses, just like the old classful addressing scheme. The address type was indicated by a set of bits at the start of the address, called the format prefix (FP). The format prefix was conceptually identical to the one to four bits used in IPv4 classful addressing to denote address classes, but was variable in length, ranging from three to ten bits. Format prefixes were described in RFC 2373.

In the years following the publication of RFC 2373, the gurus who run the Internet had a change of heart regarding how address blocks should be considered. They still wanted to divide the IPv6 address space into variably sized blocks for different purposes. However, they realized that many people were starting to consider the use of format prefixes to be equivalent to the old class-oriented IPv4 system. Their main concern was that implementers might program into IPv6 hardware logic to make routing decisions based only on the first few bits of the address. This was specifically *not* how IPv6 is supposed to work; for one thing, the allocations are subject to change.

Thus, one of the modifications made in RFC 3513 was to change the language regarding IPv6 address allocations, and specifically, to remove the term *format prefix* from the standard. The allocation of different parts of the address space is still done based on particular patterns of the first three to ten bits of the address to allow certain categories to have more addresses than others. The elimination of the specific term denoting this is intended to convey that these bits should not be given special attention.

Table 25-1 shows the allocations of the IPv6 address space and what fraction of the total address space each represents.

Table 25-1: IPv6 Address Space Allocations

Leading Bits	Fraction of Total IPv6 Address Space	Allocation
0000 0000	1/256	Unassigned (Includes special addresses such as the unspecified and loopback addresses)
0000 0001	1/256	Unassigned
0000 001	1/128	Reserved for NSAP address allocation
0000 01	1/64	Unassigned
0000 1	1/32	Unassigned
0001	1/16	Unassigned
001	1/8	Global unicast addresses
010	1/8	Unassigned
011	1/8	Unassigned
100	1/8	Unassigned
101	1/8	Unassigned
110	1/8	Unassigned
1110	1/16	Unassigned
1111 0	1/32	Unassigned
1111 10	1/64	Unassigned
1111 110	1/128	Unassigned
1111 1110 0	1/512	Unassigned
1111 1110 10	1/1024	Link-local unicast addresses
1111 1110 11	1/1024	Site-local unicast addresses
1111 1111	1/256	Multicast addresses

This is more complicated than the IPv4 classful scheme because there are so many more categories and they range greatly in size, even if most of them are currently unassigned.

An easier way to make sense of this table is to consider the division of the IPv6 address space into *eighths*. Of these eight groups, one (001) has been reserved for unicast addresses; a second (000) has been used to carve out smaller reserved blocks, and a third (111) has been used for sub-blocks for local and multicast addresses. Five are completely unassigned.

You can see that the IPv6 designers have taken great care to allocate only the portion of these "eighths" of the address space that they felt was needed for each type of address. For example, only a small portion of the part of the address space beginning 111 was used, with most of it left aside. In total, only 71/512ths of the address space is assigned right now, or about 14 percent. The other 86 percent is unassigned and kept aside for future use. (Bear in mind that even 1/1024th of the IPv6 address space is gargantuan—it represents trillions of trillions of addresses.)

Later sections in this chapter provide more information on several of these address blocks. Note that the 0000 0000 reserved block is used for several special address types, including the loopback address, the unspecified address, and IPv4 address embedding. The 1111 1111 format prefix identifies multicast addresses; this string is FF in hexadecimal, so any address beginning with FF is a multicast address in IPv6.

IPv6 Global Unicast Address Format

It is anticipated that unicast addressing will be used for the vast majority of Internet traffic under IPv6, as is the case for IPv4. It is for this reason that the largest of the assigned blocks of the IPv6 address space is dedicated to unicast addressing. A full one-eighth slice of the enormous IPv6 address "pie" is assigned to unicast addresses, which are indicated by a 001 in the first three bits of the address. The question is: How do we use the remaining 125 bits in the spacious IP addresses?

Rationale for a Structured Unicast Address Block

When IPv4 was first created, the Internet was rather small, and the model for allocating address blocks was based on a central coordinator: the Internet Assigned Numbers Authority (IANA). Everyone who wanted address blocks would go straight to the central authority. As the Internet grew, this model became impractical. Today, IPv4's classless addressing scheme allows variable-length network IDs and hierarchical assignment of address blocks. Big ISPs get large blocks from the central authority, and then subdivide them and allocate them to their customers, and so on. This is managed by today's ISPs, but there is nothing in the address space that helps manage the allocation process. In turn, each organization has the ability to further subdivide its address allocation to suit its internal requirements.

The designers of IPv6 had the benefit of this experience and realized there would be tremendous advantages to designing the unicast address structure to reflect the overall topology of the Internet. These include the following:

- Easier allocation of address blocks at various levels of the Internet topological hierarchy.
- IP network addresses that automatically reflect the hierarchy by which routers move information across the Internet, thereby allowing routes to be easily aggregated for more efficient routing.

- Flexibility for organizations like ISPs to subdivide their address blocks for customers.
- Flexibility for end-user organizations to subdivide their address blocks to match internal networks, much as subnetting did in IPv4.
- Greater meaning to IP addresses. Instead of just being a string of 128 bits with no structure, it would become possible to look at an address and know certain things about it.

Generic Division of the Unicast Address Space

The most generic way of dividing up the 128 bits of the unicast address space is into three sections, as shown in Table 25-2.

Table 25-2: Generic IPv6 Global Unicast Address Format

Field Name	Size (Bits)	Description
Prefix	*n*	Global Routing Prefix: The network ID or prefix of the address, used for routing.
Subnet ID	*m*	Subnet Identifier: A number that identifies a subnet within the site.
Interface ID	128-*n*-*m*	Interface Identifier: The unique identifier for a particular interface (host or other device). It is unique within the specific prefix and subnet.

The *global routing prefix* and *subnet identifier* represent the two basic levels at which addresses need to be hierarchically constructed: that is, global and site-specific. The routing prefix consists of a number of bits that can be further subdivided according to the needs of Internet registries and ISPs. This subdivision reflects the topography of the Internet as a whole. The subnet ID gives a number of bits to site administrators for creating an internal network structure suiting each administrator's needs.

IPv6 Implementation of the Unicast Address Space

In theory, any size for *n* and *m* (see Table 25-2) could be used. The implementation chosen for IPv6, however, assigns 48 bits to the routing prefix and 16 bits to the subnet identifier. This means 64 bits are available for interface identifiers, which are constructed based on the IEEE EUI-64 format, as described in the next section. Thus, the overall IPv6 unicast address format is constructed as shown in Table 25-3 and illustrated in Figure 25-3.

Table 25-3: IPv6 Global Unicast Address Format

Field Name	Size (Bits)	Description
Prefix	48	Global Routing Prefix: The network ID or prefix of the address that's used for routing. The first three bits are 001 to indicate a unicast address.
Subnet ID	16	Subnet Identifier: A number that identifies a subnet within the site.
Interface ID	64	Interface ID: The unique identifier for a particular interface (host or other device). It is unique within the specific prefix and subnet.

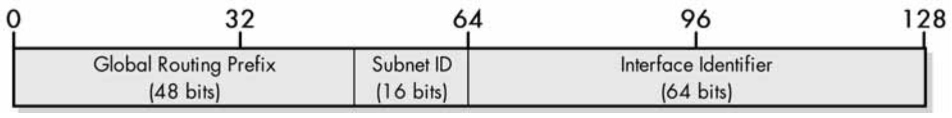

Figure 25-3: *IPv6 global unicast address format*

KEY CONCEPT The part of the IPv6 address space set aside for unicast addresses is structured into an address format that uses the first 48 bits for the *routing prefix* (like a network ID), the next 16 bits for a *subnet ID*, and the final 64 bits for an *interface ID* (like a host ID).

Due to this structure, most end sites (regular companies and organizations, as opposed to ISPs) will be assigned IPv6 networks with a 48-bit prefix. In common parlance, these network identifiers have now come to be called *48s* or */48s*.

The 16 bits of subnet ID allow each site considerable flexibility in creating subnets that reflect the site's network structure. Here are some example uses of the 16 bits:

- A smaller organization can just set all the bits in the subnet ID to zero and have a flat internal structure.
- A medium-sized organization could use all the bits in the subnet ID to perform the equivalent of straight subnetting under IPv4, thereby assigning a different subnet ID to each subnet. There are 16 bits here, and this allows a whopping 65,536 subnets!
- A larger organization can use the bits to create a multiple-level hierarchy of subnets, exactly like IPv4's Variable Length Subnet Masking (VLSM). For example, the company could use two bits to create four subnets. It could then take the next three bits to create eight sub-subnets in some or all of the four subnets. There would still be 11 more bits to create sub-sub-subnets, and so forth.

Original Division of the Global Routing Prefix: Aggregators

The global routing prefix is similarly divided into a hierarchy, but one that has been designed for the use of the entire Internet, like CIDR. There are 45 bits available here (48 bits minus the first three that are fixed at 001). That is a lot. When the unicast address structure was first detailed in RFC 2374, that document described a specific division of the 45 bits based on a two-level hierarchical topology of Internet registries and providers. These organizations were described as follows:

Top-Level Aggregators (TLAs) These refer to the largest Internet organizations, which were to be assigned large blocks of IPv6 addresses from registration authorities.

Next-Level Aggregators (NLAs) These organizations would get blocks of addresses from TLAs and divide them for end-user organizations (sites).

The 45 bits were split between these two uses, with a few bits reserved in the middle to allow expansion of either field if needed. Thus, the RFC 2374 structure for the 45 bits appeared as listed in Table 25-4.

Table 25-4: Historical IPv6 Unicast Routing Prefix Structure

Field Name	Size (Bits)	Description
TLA ID	13	Top-Level Aggregation (TLA) Identifier: A globally unique identifier for the top-level aggregator. There are 13 bits, so there were a maximum of 8,192 TLAs allowed.
RES	8	Reserved: These 8 bits were reserved for future use and set to zero. By leaving these 8 bits between the TLA ID and NLA ID unused, they could later be used to expand either the TLA ID or NLA ID fields as needed.
NLA ID	24	Next-Level Aggregation (NLA) Identifier: Each TLA was given this 24-bit field to generate blocks of addresses for allocation to its customers. The NLA ID is unique for each TLA ID. The use of the 24 bits was left up to the TLA organization.

You'll notice my use of the past tense in the description of the TLA/NLA structure, and that table heading is a pretty big giveaway, too. In August 2003, RFC 3587 was published, which in a nutshell says, "Uh, never mind about all that TLA/NLA stuff." The decision was made that having this structure hardwired into an Internet standard was inflexible, and it made more sense to let the regional Internet registries (APNIC, ARIN, LACNIC, and RIPE) decide for themselves how to use the 45 bits.

NOTE *The obsoleting of the TLA/NLA structure occurred after many years of people getting used to it, so for some time to come, you will still routinely see those terms mentioned in IPv6 descriptions. (This is why I included discussion of them here.)*

A Sample Division of the Global Routing Prefix into Levels

There is no single structure for determining how the 48-bit routing prefix is divided in the global unicast hierarchy. As one example, it might be possible to divide it into three levels, as shown in Table 25-5 and illustrated in Figure 25-4.

Table 25-5: Example IPv6 Unicast Routing Prefix Structure

Field Name	Size (Bits)	Description
(Unicast Indicator)	3	Each unicast address starts with 001; there is no official name for this (it used to be called the *format prefix*).
Level1 ID	10	Level 1 Identifier: The identifier of the highest level in the hierarchy. This would be used for assigning the largest blocks of addresses in the global hierarchy to the biggest Internet organizations. The number of level 1 organizations would be 210, or 1,024.
Level2 ID	12	Level 2 Identifier: Each block assigned to a level 1 organization would use 12 bits to create 4,096 address blocks to divide among the lower-level organizations it serves.
Level3 ID	23	Level 3 Identifier: Each level 2 organization has 23 bits to use to divide its level 2 address block. Thus, it could create over 8 million individual /48 address blocks to assign to end-user sites. Alternatively, the 23 bits could be divided further into still lower levels to reflect the structure of the level 2 organization's customers.

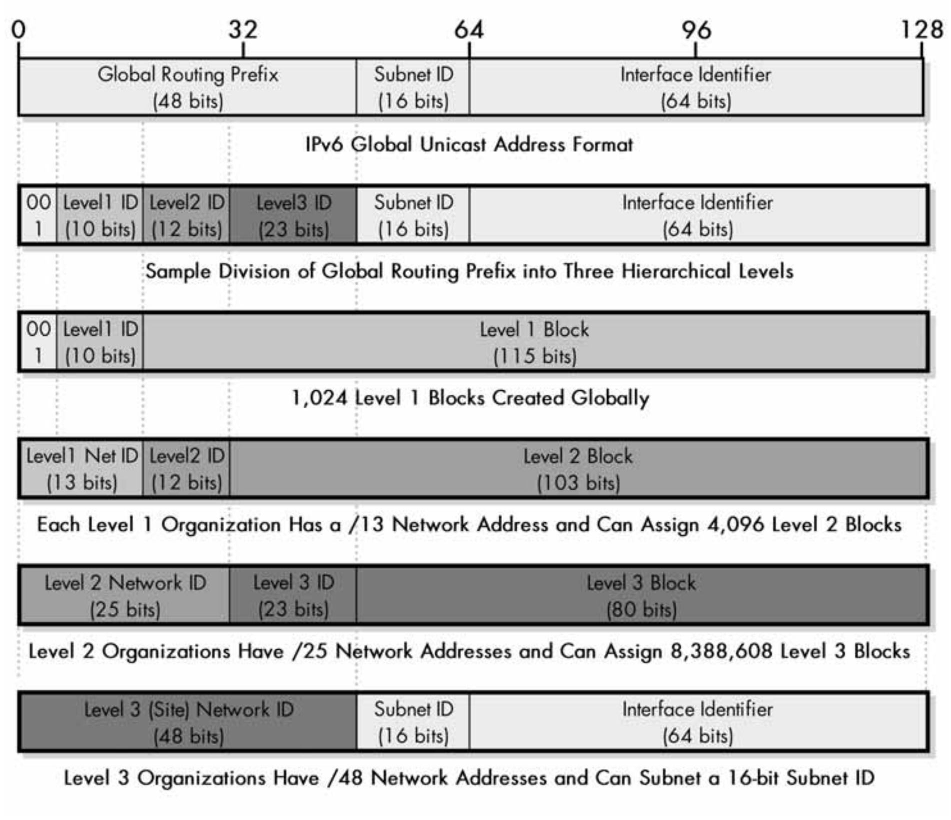
0
32
64
96
128
Global Routing Prefix (48 bits)
Subnet ID (16 bits)
Interface Identifier (64 bits)
IPv6 Global Unicast Address Format
001
Level1 ID (10 bits)
Level2 ID (12 bits)
Level3 ID (23 bits)
Subnet ID (16 bits)
Interface Identifier (64 bits)
Sample Division of Global Routing Prefix into Three Hierarchical Levels
001
Level1 ID (10 bits)
Level 1 Block (115 bits)
1,024 Level 1 Blocks Created Globally
Level1 Net ID (13 bits)
Level2 ID (12 bits)
Level 2 Block (103 bits)
Each Level 1 Organization Has a /13 Network Address and Can Assign 4,096 Level 2 Blocks
Level 2 Network ID (25 bits)
Level 3 ID (23 bits)
Level 3 Block (80 bits)
Level 2 Organizations Have /25 Network Addresses and Can Assign 8,388,608 Level 3 Blocks
Level 3 (Site) Network ID (48 bits)
Subnet ID (16 bits)
Interface Identifier (64 bits)
Level 3 Organizations Have /48 Network Addresses and Can Subnet a 16-bit Subnet ID

Figure 25-4: Example of IPv6 unicast routing prefix structure *The top row shows the global IPv6 unicast address format. The second shows one way to divide the global routing prefix into three levels using 10, 12, and 23 bits, respectively. The third row shows how the first 10 bits are used to create* 2^{10}*, or 1,024, different level 1 blocks. The next row illustrates that for each of these 13-bit prefixes, you could have* 2^{12}*, or 4,096, level 2 blocks. Then, within each 25-bit level 2 ID, you have 23 bits, or 8,388,608, level 3 blocks. At the bottom, a level 3 or /48 would be assigned to an individual organization.*

This is just one possible theoretical way that the bits in a /48 network address could be assigned. As you can see, with so many bits, there is a lot of flexibility. In the previous scheme, you can have over four million level 2 organizations, each of which can assign eight million /48 addresses. And each of those is equivalent in size to an IPv4 Class B address (over 65,000 hosts)!

The removal of RFC 2374's fixed structure for the global routing prefix is consistent with the IPv6 development team's efforts to emphasize that bit fields and structures are used only for allocating addresses and not for routing purposes. The addresses themselves, once created, are not interpreted by hardware on an internetwork based on this format. To routers, the only structure that matters is the division between the network ID and host ID, given by the prefix length that trails the IP address, and this division can occur at any bit boundary. These hardware devices just see 128 bits of an IP address and use it without any knowledge of hierarchical address divisions or levels.

Incidentally, the key to obtaining the allocation benefits of the aggregatable unicast address format is the abundance of bits available to us under IPv6. The ability to have these hierarchical levels while still allowing 64 bits for the interface identifier is one of the main reasons why IPv6 designers went all the way from 32 bits to 128 bits for address size. By creating this structure, we maintain flexibility, while avoiding the potential chaos of trying to allocate many different network sizes within the 128 bits.

Note that anycast addresses are structured in the same way as unicast addresses, so they are allocated according to this same model. (Multicast addresses are not.)

IPv6 Interface Identifiers and Physical Address Mapping

In IPv4, IP addresses have no relationship to the addresses used for underlying data link layer network technologies. A host that connects to a TCP/IP network using an Ethernet network interface card (NIC) has an Ethernet MAC address and an IP address, but the two numbers are distinct and unrelated in any way. IP addresses are assigned manually by administrators without any regard for the underlying physical address.

With the overhaul of addressing in IPv6, an opportunity presented itself to create a better way of mapping IP unicast addresses and physical network addresses. Implementing this superior mapping technique was one of the reasons why IPv6 addresses were made so large. With 128 total bits, even with a full 45 bits reserved for the network prefix and 16 bits for the site subnet, we are still left with 64 bits to use for the *interface identifier (interface ID)*, which is analogous to the host ID under IPv4.

Having so many bits at our disposal gives us great flexibility. Instead of using arbitrary, made-up identifiers for hosts, we can base the interface ID on the underlying data link layer hardware address, as long as that address is no greater than 64 bits in length. Since virtually all devices use layer 2 addresses of 64 bits or fewer, there is no problem in using those addresses for the interface ID in IP addresses. This provides an immediate benefit: It makes networks easier to administer, since we don't need to record two arbitrary numbers for each host. The IP address can be derived from the MAC address and the network ID. It also means that we can tell the IP address from the MAC address and vice versa.

The actual mapping from data link layer addresses to IP interface IDs depends on the particular technology. It is essential that all devices on the same network use the same mapping technique, of course. By far, the most common type of layer 2 addresses in networking are IEEE 802 MAC addresses, which are used by Ethernet and other IEEE 802 Project networking technologies. These addresses have 48 bits, arranged into two blocks of 24. The upper 24 bits are arranged into a block called the *organizationally unique identifier (OUI)*, with different values assigned to individual organizations. The lower 24 bits are then used for an identifier for each specific device.

The IEEE has also defined a format called the *64-bit extended unique identifier*, which is abbreviated *EUI-64*. It is similar to the 48-bit MAC format, except that while the OUI remains at 24 bits, the device identifier becomes 40 bits instead of 24. This gives each manufacturer 65,536 times as many device addresses within its OUI.

A form of this format, called *modified EUI-64*, has been adopted for IPv6 interface IDs. To get the modified EUI-64 interface ID for a device, you simply take the

EUI-64 address and change the seventh bit from the left (the universal/local, or U/L, bit) from a 0 to a 1.

Of course, most devices still use the older 48-bit MAC address format. These can be converted to EUI-64 and then modified to EUI-64 form for creating an IPv6 interface ID. The process is as follows:

1. Take the 24-bit OUI portion, the leftmost 24 bits of the Ethernet address, and put them into the leftmost 24 bits of the interface ID. Take the 24-bit local portion (the rightmost 24 bits of the Ethernet address) and put it into the rightmost 24 bits of the interface ID.
2. In the remaining 16 bits in the middle of the interface ID, put the value 11111111 11111110, FFFE in hexadecimal.
3. The address is now in EUI-64 form. Change the universal/local bit (bit 7 from the left, shown in bold in Figure 25-5) from a 0 to a 1. This gives the modified EUI-64 interface ID.

KEY CONCEPT The last 64 bits of IPv6 unicast addresses are used for interface IDs, which are created in a special format called *modified EUI-64*. A simple process can be used to determine the interface ID from the 48-bit MAC address of a device like an Ethernet network interface card. This can then be combined with a network prefix (routing prefix and subnet ID) to determine a corresponding IPv6 address for the device.

Let's take as an example the Ethernet address of 39-A7-94-07-CB-D0. Here are the steps for conversion (illustrated in Figure 25-5):

1. Take 39-A7-94, the first 24 bits of the identifier, and put it into the first (leftmost) 24 bits of the address. The local portion of 07-CB-D0 becomes the last 24 bits of the identifier.
2. The middle 16 bits are given the value FF-FE.
3. Change the seventh bit from 0 to 1, which changes the first octet from 39 to 3B.

The identifier thus becomes 3B-A7-94-FF-FE-07-CB-D0, or in IPv6 colon hexadecimal notation, 3BA7:94FF:FE07:CBD0. The first 64 bits of the device's address are supplied using the global unicast address format.

The only drawback of this technique is that if the physical hardware changes, so does the IPv6 address.

IPv6 Special Addresses: Reserved, Private, Unspecified, and Loopback

Just as certain IPv4 address ranges are designated for reserved, private, and other unusual addresses, a small part of the monstrous IPv6 address space has been set aside for special addresses. The purpose of these addresses and address blocks is to provide addresses for special requirements and private use in IPv6 networks. Since even relatively small pieces of IPv6 are still enormous, setting aside 0.1 percent of the address space for a particular use still generally yields more addresses than anyone will ever need.

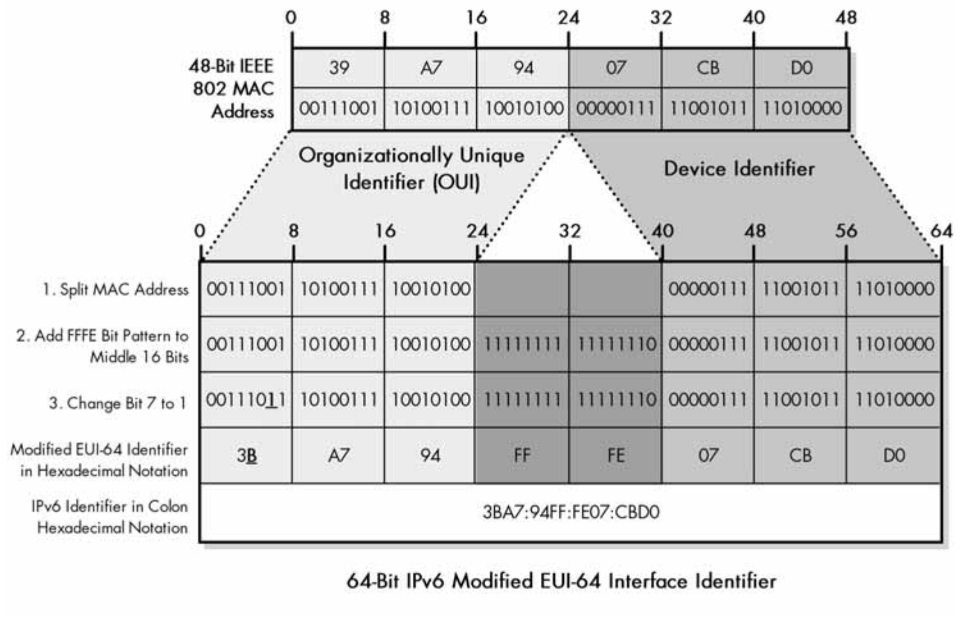

Figure 25-5: *Converting IEEE 802 MAC addresses to IPv6 modified EUI-64 identifiers*

Special Address Types

There are four basic types of special IPv6 addresses:

Reserved Addresses A portion of the address space is set aside as reserved for various uses by the IETF, both present and future. Unlike IPv4, which has many small reserved blocks in various locations in the address space, the reserved block in IPv6 is at the "top" of the address space, beginning with 0000 0000 (or 00 for the first hexadecimal octet). This represents 1/256th of the total address space. Some of the special addresses you'll see shortly come from this block. IPv4 address embedding is also done within this reserved address area.

NOTE *Reserved addresses are not the same as* unassigned *addresses. The latter term just refers to blocks whose use has not yet been determined.*

Private/Unregistered/Nonroutable Addresses A block of addresses is set aside for private addresses, just as in IPv4, except that like everything in IPv6 the private address block in IPv6 is much larger. These private addresses are local only to a particular link or site and, therefore, are never routed outside a particular company's network. Private addresses are indicated by the address having "1111 1110 1" for the first nine bits. Thus, private addresses have a first octet value of FE in hexadecimal, with the next hexadecimal digit being from 8 to F. These addresses are further divided into two types based on their scope: site-local and link-local, as discussed shortly.

Loopback Address Like IPv4, a provision has been made for a special loopback address for testing; datagrams sent to this address "loop back" to the sending device. However, in IPv6, there is just one address for this function, not a whole block (which was never needed in the first place). The loopback address is 0:0:0:0:0:0:0:1, which is normally expressed using zero compression as ::1.

Unspecified Address In IPv4, an IP address of all zeros has a special meaning: It refers to the host itself and is used when a device doesn't know its own address. In IPv6, this concept has been formalized, and the all-zeros address (0:0:0:0:0:0:0:0) is named the *unspecified address*. It is typically used in the source field of a datagram sent by a device seeking to have its IP address configured. Zero compression can be applied to this address; since it is all zeros, the address becomes just ::. (I consider this confusing, myself. I think something like 0::0 is a lot clearer and short enough.)

KEY CONCEPT In IPv6, a special *loopback address*, 0:0:0:0:0:0:0:1 (::1 in compressed form) is set aside for testing purposes. The *unspecified address*, 0:0:0:0:0:0:0:0 (:: in compressed form) is used to indicate an unknown address. A block of *private* or *local* addresses is defined. This block is the set of all addresses beginning with 1111 1110 1 as the first nine bits.

IPv6 Private Addresses Type Scopes

Now let's take a closer look at private addresses. In IPv6, these are called *local-use* addresses, with the name conveying clearly what they are for. They are also sometimes called *link-layer* addresses. You'll recall that IPv4 private addresses were commonly used when public addresses could not be obtained for all devices, sometimes in combination with technologies like Network Address Translation (NAT). In IPv6, trickery like NAT isn't required. Instead, local-use addresses are intended for communication that is inherently designed to be sent to local devices only. For example, neighbor discovery functions using the IPv6 Neighbor Discovery (ND) protocol employ local-use addresses.

The *scope* of local addresses is obviously a local network, not the global scope of public Internet addresses. Local addresses in IPv6 are further divided into two types, reflecting a division of local scope:

Site-Local Addresses These addresses have the scope of an entire site or organization. They allow addressing within an organization without having to use a public prefix. Routers will forward datagrams using site-local addresses within the site, but not addresses outside it to the public Internet. Site-local addresses are differentiated from link-local addresses by having a tenth bit of 1 following the nine starting address bits that are common to all private IPv6 addresses. Thus, they begin with 1111 1110 11. In hexadecimal, site-local addresses begin with FE, and then C to F for the third digit. So, these addresses start with FEC, FED, FEE, or FEF.

Link-Local Addresses These addresses have a smaller scope than site-local addresses; they refer only to a particular physical link (physical network). Routers will not forward datagrams using link-local addresses at all—not even within the organization. These addresses are only for local communication on a particular physical network segment. They can be used for address configuration or for ND functions such as address resolution and ND. Link-local addresses are differentiated from site-local addresses by having a tenth bit of 0 following the nine initial address bits common to all private IPv6 addresses: 1111 1110 1. Thus, site-local addresses begin with FE, and then 8 to B for the third hexadecimal digit. So, these addresses start with FE8, FE9, FEA, or FEB.

KEY CONCEPT IPv6 site-local addresses allow data to be sent only to the devices within a site or organization. They begin with FEC, FED, FEE, or FEF in hexadecimal. IPv6 link-local addresses are used only on a particular local link (physical network), typically for special purposes such as address resolution or Neighbor Discovery (ND). They start with FE8, FE9, FEA, or FEB.

Note that site-local IPv6 addresses are the equivalent of IPv4 private addresses, since they are routed throughout the organization. The concept of link-local scope is new to IPv6.

IPv6/IPv4 Address Embedding

Due to the importance of IP and the significance of the changes made in IPv6, deployment of the newer version of the protocol will not occur all at once. A *transition* from IPv4 to IPv6 will be required. This transition requires careful planning. It is anticipated that the migration from IPv4 to IPv6 will take many years, as I mentioned earlier.

IPv6 is backward-compatible with IPv4 provided that you use special techniques. For example, to enable communication between islands of IPv6 devices connected by IPv4 networks, you may need to employ tunneling. To support IPv4/IPv6 compatibility, a scheme was developed to allow IPv4 addresses to be *embedded* within the IPv6 address structure. This method takes regular IPv4 addresses and puts them in a special IPv6 format, so that they are recognized as being IPv4 addresses by certain IPv6 devices.

Since the IPv6 address space is so much bigger than the one in IPv4, embedding the latter within the former is easy—it's like tucking a compact sedan into the hold of a cargo ship! The embedding address space is part of the reserved address block whose addresses begin with eight 0 bits, but it's only a relatively small part. Two different embedding formats are used to indicate the capabilities of the device that's using the embedded address:

IPv4-Compatible IPv6 Addresses These are special addresses assigned to IPv6-capable devices, such as *dual-stack* devices that use both IPv4 and IPv6. They have all zeros for the middle 16 bits; thus, they start off with a string of 96 zeros, followed by the IPv4 address. An example of such an address would be 0:0:0:0:0:0:101.45.75.219 in mixed notation, or more succinctly, ::101.45.75.219. Figure 25-6 illustrates IPv4-compatible IPv6 representation.

IPv4-Mapped IPv6 Addresses These are regular IPv4 addresses that have been mapped into the IPv6 address space. They are used for devices that are IPv4-capable only. They have a set of 16 ones after the initial string of 80 zeros and then the

IPv4 address. So if an IPv4 device has the address 222.1.41.90, such as the one shown in Figure 25-7, it would be represented as 0:0:0:0:0:FFFF:222.1.41.90, or ::FFFF:222.1.41.90.

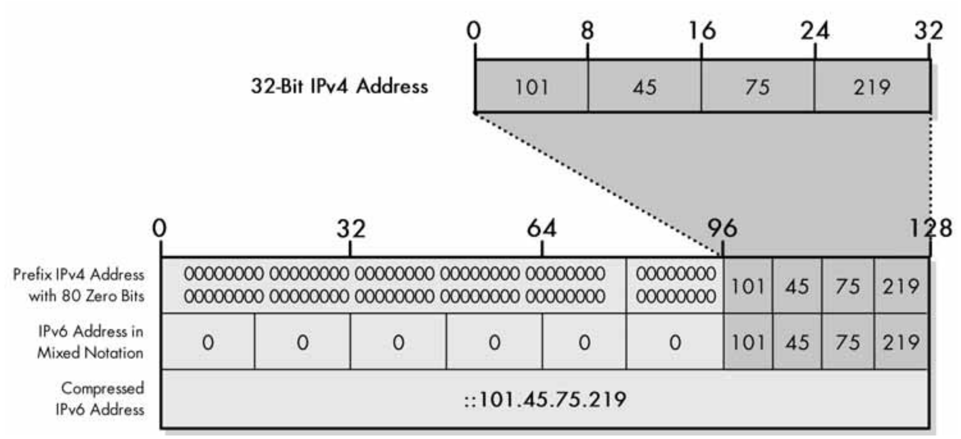

Figure 25-6: *IPv4-compatible embedded IPv6 address representation*

The difference between these two is subtle but important. Both have zeros for the first 80 bits of the address and put the embedded IPv4 address into the last 32 bits of the IPv6 address format. They differ in the value of the 16 remaining bits in between (bits 81 to 96, counting from the left). IPv4-compatible IPv6 addresses are used only for devices that are actually IPv6-aware; the IPv4-compatible address is in addition to its conventional IPv6 address. In contrast, if the FFFF is seen for the 16 bits after the initial 80, this designates a conventional IPv4 devices whose IPv4 address has been mapped into the IPv6 format. It is not an IPv6-capable device.

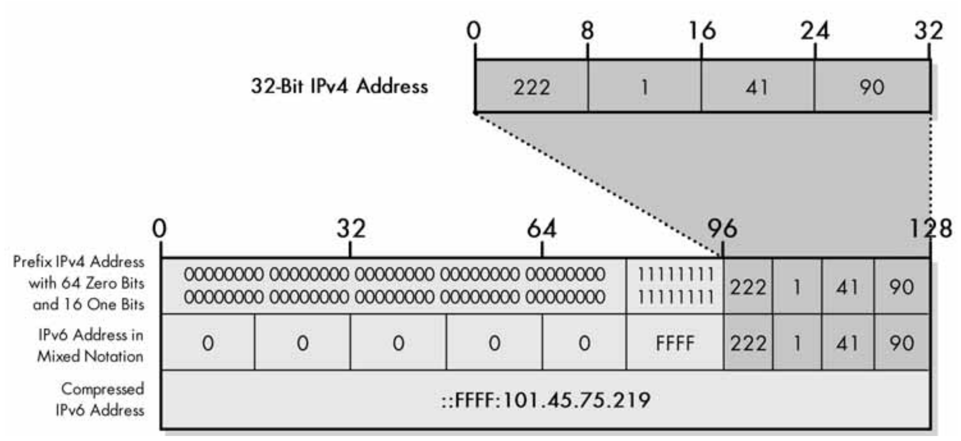

Figure 25-7: *IPv4-mapped embedded IPv6 address representation*

KEY CONCEPT *IPv4 address embedding* is used to create a relationship between an IPv4 address and an IPv6 address to help you transition from IPv4 to IPv6. One type, the *IPv4-compatible IPv6 address*, is used for devices that are compatible with both IPv4 and IPv6; it begins with 96 zero bits. The other, the *IPv4-mapped address*, is used for mapping IPv4 devices that are not compatible with IPv6 into the IPv6 address space; it begins with 80 zeros followed by 16 ones.

IPv6 Multicast and Anycast Addressing

One of the most significant modifications in the general addressing model in IPv6 was a change to the basic types of addresses and how they were used. Unicast addresses are still the choice for the vast majority of communications as in IPv4, but the "bulk" addressing methods are different in IPv6. Broadcast as a specific addressing type has been eliminated. Instead, support for multicast addressing has been expanded and made a required part of the protocol, and a new type of addressing called *anycast* has been implemented.

IPv6 Multicast Addresses

Let's start by looking at multicast under IPv6. Multicasting is used to allow a single device to send a datagram to a group of recipients. IPv4 supported multicast addressing using the Class D address block in the classful addressing scheme (see Chapter 17). Under IPv6, multicast addresses are allocated from the multicast block. This is 1/256th of the address space, and it consists of all addresses that begin with 1111 1111. Thus, any address starting with FF in colon hexadecimal notation is an IPv6 multicast address.

The remaining 120 bits of address space are enough to allow the definition of, well, a gazillion or three multicast addresses. (OK, it's officially about 1.3 trillion trillion trillion addresses.) The allocation of unicast addresses was organized by using a special format to divide these many bits, and the same thing was done for multicast addresses. The format for multicast addresses is explained in Table 25-6 and illustrated in Figure 25-8.

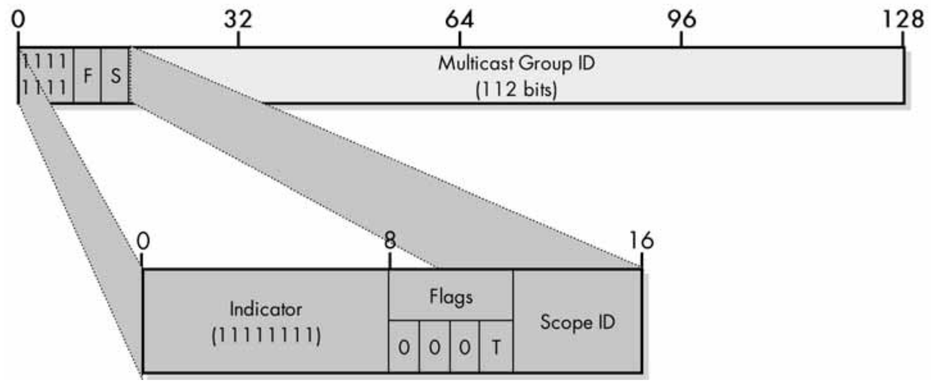

Figure 25-8: *IPv6 multicast address format*

Table 25-6: IPv6 Multicast Address Format

Field Name	Size (Bits)	Description
(Indicator)	8	The first eight bits are always 1111 1111, which indicates a multicast address. This used to be called the *format prefix* before the term was dropped (as explained in the section about IPv6 address space allocation earlier in this chapter). The field now has no name.
Flags	4	Four bits are reserved for flags that can be used to indicate the nature of certain multicast addresses. Currently, the first three of these are unused and set to zero. The fourth is the T (Transient) flag. If left as zero, this marks the multicast address as a permanently assigned, well-known multicast address, as you will see shortly. If set to one, this means this is a *transient* multicast address, meaning that it is not permanently assigned.
Scope ID	4	These four bits are used to define the scope of the multicast address; 16 different values from 0 to 15 are possible. This field allows creation of multicast addresses that are global to the entire Internet, or restricted to smaller spheres of influence such as a specific organization, site, or link. The currently defined values (in decimal) are as follows: 0 = Reserved 1 = Node-Local Scope 2 = Link-Local Scope 5 = Site-Local Scope 8 = Organization-Local Scope 14 = Global Scope 15 = Reserved
Group ID	112	Defines a particular group within each scope level.

Multicast Scopes

The notion of explicitly scoping multicast addresses is important. Globally scoped multicast addresses must be unique across the entire Internet, but locally scoped addresses are unique only within the organization. This provides tremendous flexibility, as every type of multicast address actually comes in several versions: one that multicasts only within a node, one that multicasts on the local link (local network), one that multicasts on the local site, and so on. The scope also allows routers to immediately determine how broadly they should propagate multicast datagrams in order to improve efficiency and eliminate problems with traffic being sent outside the area for which it is intended. Figure 25-9 illustrates the notion of multicast scope graphically.

KEY CONCEPT Multicast addresses are used to send data to a number of devices on an internetwork simultaneously. In IPv6, each multicast address can be specified for a variety of different *scopes*, thereby allowing a transmission to be targeted to either a wide or a narrow audience of recipient devices.

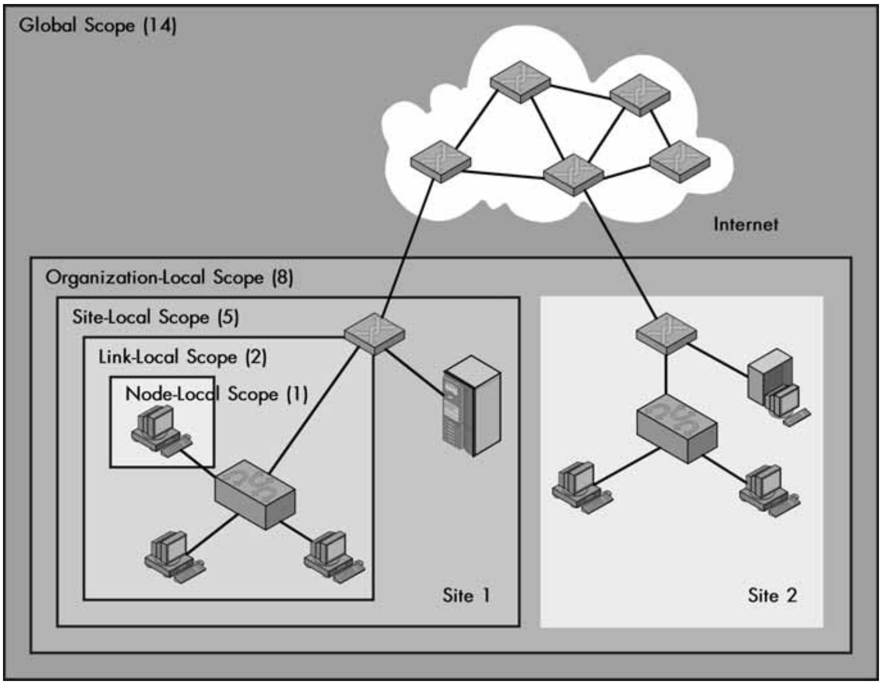

Figure 25-9: IPv6 multicast scope *This diagram shows how the notion of scope allows IPv6 multicasts to be limited to specific spheres of influence. The tightest scope is node-local scope, with a scope ID value of 1. As the scope ID value increases, the scope expands to cover the local network, site, organization, and finally, entire Internet.*

Well-Known Multicast Addresses

The Transient flag allows for the explicit determination of which multicast addresses are available for normal use compared to which ones are set aside as well known. Several well-known multicast addresses are defined by setting aside certain group IDs that are used for a number of different scope ID values. Table 25-7 shows these values; the *x* in the multicast address pattern is the hexadecimal digit corresponding to the four-bit scope ID field.

The all-nodes and all-routers multicast addresses enable the equivalent function of what broadcast used to perform in IPv4. Again, the concept of scope is important in a multicast of this type, because we don't want to try to send a message to all nodes on the global Internet, for example. So when the all-routers address is used with a scope value of 2, it means "all routers on the local link." If it is used with a value of 5, it means "all routers in this site."

Solicited-Node Multicast Addresses

In addition to the regular multicast addresses, each unicast address has a special multicast address called its *solicited-node address*. This address is created through a special mapping from the device's unicast address. Solicited-node addresses are used by the IPv6 ND protocol (see Chapter 36) to provide more efficient address resolution than the Address Resolution Protocol (ARP; see Chapter 13) technique used in IPv4.

Table 25-7: Important IPv6 Well-Known Multicast Addresses

Multicast Address Pattern	Valid Scope Values (Decimal)	Designation	Description
FF0x:0:0:0:0:0:0	0 to 15	Reserved	All multicast addresses where the 112-bit group ID is zero are reserved.
FF0x:0:0:0:0:0:1	1, 2	All Nodes	When the group ID is equal to exactly 1, this is a multicast to all nodes. Both node-local (FF01:0:0:0:0:0:1) and link-local (FF02:0:0:0:0:0:1) all-nodes multicast addresses are possible.
FF0x:0:0:0:0:0:2	1, 2, 5	All Routers	When the group ID is equal to exactly 2, this designates all routers within a specific scope as the recipients. Valid scope values are node-local, link-local, and site-local.

All solicited-node addresses have their T flag set to zero and a scope ID of 2, so they start with FF02. The 112-bit group ID is broken down as follows (see Figure 25-10):

- Eighty bits consisting of 79 zeros followed by a single one. This means that the next five hexadecimal values are 0000:0000:0000:0000:0001 in colon hexadecimal notation, or more succinctly, 0:0:0:0:1.
- Eight ones: FF.
- Twenty-four bits taken from the bottom 24 bits of its unicast address.

So, these addresses start with FF02:0:0:0:0:1:FF, followed by the bottom 24 bits of the unicast address. Thus, the node with IP address 805B:2D9D:DC28:0:0:FC57:D4C8:1FFF would have a solicited-node address of FF02:0:0:0:0:1:FFC8:1FFF (or FF02::1:FFC8:1FFF).

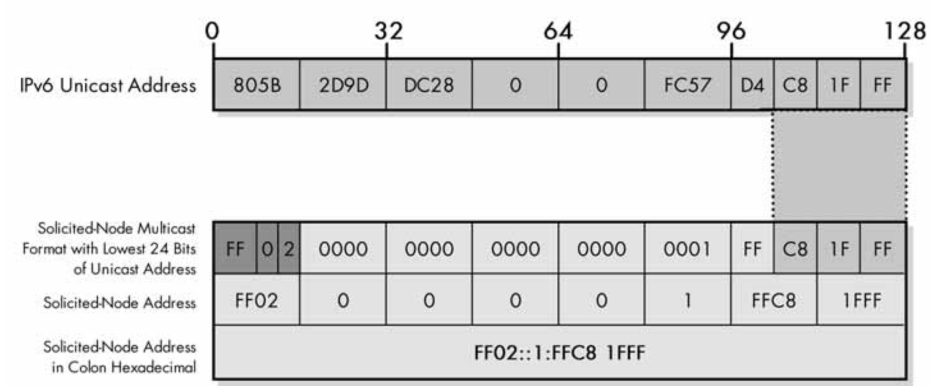

Figure 25-10: IPv6 solicited-node address calculation *The solicited-node multicast address is calculated from a unicast address by taking the last 24 bits of the address and prepending them with the IPv6 partial address FF02:0:0:0:0:1:FF. This shows the example address from Figure 25-2 converted to its solicited-node address, FF02::1:FFC8:1FFF.*

KEY CONCEPT Each unicast address has an equivalent *solicited-node multicast address* that is created from the unicast address and used when other devices need to reach it on the local network.

IPv6 Anycast Addresses

Anycast addresses are a unique type of address that is new to IP in IPv6. The IPv6 implementation is based on the material in RFC 1546, "Host Anycasting Service." Anycast addresses can be considered a conceptual cross between unicast and multicast addressing. Where unicast says, "Send this to one address," and multicast says, "Send this to every member of this group," anycast says, "Send this to any one member of this group." Naturally, in choosing which member to send to, we would, for efficiency, normally send to the closest one—that is, the closest in routing terms. So, we can normally also consider *anycast* to mean, "Send this to the closest member of this group."

The idea behind anycast is to enable functionality that was previously difficult to implement in TCP/IP. Anycast was specifically intended to provide flexibility in situations where we need a service that is provided by a number of different servers or routers but don't really care which one provides it. In routing, anycast allows datagrams to be sent to whichever router in a group of equivalent routers is closest, and to allow load sharing among routers and dynamic flexibility if certain routers go out of service. Datagrams sent to the anycast address will automatically be delivered to the device that is easiest to reach.

Perhaps surprisingly, there is no special anycast-addressing scheme. Anycast addresses are the same as unicast addresses. An anycast address is created automatically when a unicast address is assigned to more than one interface.

KEY CONCEPT Anycast addresses are new in IPv6 and can be used to set up a group of devices, any one of which can respond to a request sent to a single IP address.

Like multicast, anycast creates more work for routers, because it is more complicated than unicast addressing. In particular, the further apart the devices that share the anycast address are, the more complexity. Anycasting across the global Internet would be potentially difficult to implement, and IPv6 anycasting was designed for devices that are proximate to each other, generally within the same network. Also, at present, due to the Internet community's relative inexperience with anycast, only routers, not individual hosts, use anycast addresses.

IPv6 Autoconfiguration and Renumbering

One of the most interesting and potentially valuable addressing features implemented in IPv6 is a facility that allows devices on an IPv6 network to actually configure themselves independently. In IPv4, hosts were originally configured manually. Later, host configuration protocols like the Dynamic Host Configuration Protocol (DHCP; see Chapter 61) enabled servers to allocate IP addresses to hosts that joined the network. IPv6 takes this a step further by defining a method for some devices to automatically configure their IP address and other parameters without the need for a server. It also defines a method whereby the IP addresses on a network can be renumbered (changed en masse). These are the sorts of features that make TCP/IP network administrators drool.

The IPv6 autoconfiguration and renumbering feature is defined in RFC 2462, “IPv6 Stateless Address Autoconfiguration.” The word *stateless* contrasts this method to the server-based method using something like DHCPv6, which is called *stateful*. (This word, like *classful*, makes me cringe.) This method is called stateless because it begins with no information (or *state*) at all for the host to work with. It has no need for a DHCP server.

IPv6 Stateless Autoconfiguration

Stateless autoconfiguration exploits several other new features in IPv6, including link-local addresses, multicasting, the ND protocol, and the ability to generate the interface ID of an address from the underlying data link layer address. The general idea is to have a device generate a temporary address until it can determine the characteristics of the network it is on, and then create a permanent address it can use based on that information. In the case of multihomed devices, autoconfiguration is performed for each interface separately.

The following is a summary of the steps a device takes when using stateless autoconfiguration:

1. **Link-Local Address Generation** The device generates a link-local address. You’ll recall that this is one of the two types of local-use IPv6 addresses. Link-local addresses have 1111 1110 10 for the first 10 bits. The generated address uses those 10 bits, followed by 54 zeros and then the 64-bit interface ID. Typically, this will be derived from the data link layer (MAC) address as explained in the “IPv6 Interface Identifiers and Physical Address Mapping” section earlier in this chapter, or it may be a “token” generated in some other manner.
2. **Link-Local Address Uniqueness Test** The node tests to ensure that the address it generated isn’t already in use on the local network. (This is very unlikely to be an issue if the link-local address came from a MAC address; it is more likely that the address is already in use if it was based on a generated token.) It sends a Neighbor Solicitation message using the ND protocol. In response, it listens for a Neighbor Advertisement, which indicates that another device is already using its link-local address. If so, either a new address must be generated or autoconfiguration fails, and another method must be employed.
3. **Link-Local Address Assignment** Assuming the uniqueness test passes, the device assigns the link-local address to its IP interface. This address can be used for communication on the local network, but not on the wider Internet (since link-local addresses are not routed).
4. **Router Contact** The node next attempts to contact a local router for more information on continuing the configuration. This is done either by listening for Router Advertisement messages sent periodically by routers or by sending a specific Router Solicitation message to ask a router for information on what to do next. This process is described in the section on the IPv6 ND protocol, in Chapter 36.

5. **Router Direction** The router provides direction to the node about how to proceed with the autoconfiguration. It may tell the node that on this network stateful autoconfiguration is in use, and it may give it the address of a DHCP server to use. Alternatively, it will tell the host how to determine its global Internet address.
6. **Global Address Configuration** Assuming that stateless autoconfiguration is in use on the network, the host will configure itself with its globally unique Internet address. This address is generally formed from a network prefix provided to the host by the router. The prefix is combined with the device's identifier, as generated in step 1.

Clearly, this method has numerous advantages over both manual and server-based configuration. It is particularly helpful in supporting the mobility of IP devices, because they can move to new networks and get a valid address without any knowledge of local servers or network prefixes. At the same time, it still allows for the management of IP addresses using the (IPv6-compatible) version of DHCP, if that is desired. Routers on the local network will typically tell hosts which type of autoconfiguration is supported using special flags in Internet Control Message Protocol version 6 (ICMPv6) Router Advertisement messages (see Chapter 35).

KEY CONCEPT IPv6 includes an interesting feature called *stateless address autoconfiguration*, which allows a host to actually determine its own IPv6 address from its layer 2 address by following a special procedure.

IPv6 Device Renumbering

The renumbering of devices is a method related to autoconfiguration. Like host configuration, it can be implemented using protocols like DHCP through the use of IP address leases that expire after a period of time. Under IPv6, networks can be renumbered by having routers specify an expiration interval for network prefixes when autoconfiguration is done. Later, they can send a new prefix to tell devices to regenerate their IP addresses. Devices can actually maintain the old deprecated address for a while, and then move over to the new address.

RFC 2894 defined a similar technique for renumbering router addresses. It uses special ICMPv6 messages and is described in Chapter 35.