# IP Concepts

## IP Overview and Key Operational Characteristics

The Internet Protocol (IP) is the core of the TCP/IP protocol suite and its main protocol at the network layer. The network layer is primarily concerned with the delivery of data, not between devices on the same physical network, but between devices that may be on different networks that are interconnected in an arbitrary manner: an *internetwork*. IP is the mechanism by which this data is sent on TCP/IP networks. (It does have help from other protocols at the network layer too, of course!)

Let's look at the TCP/IP layer model and consider what IP does from an architectural standpoint. As the layer three protocol, it provides a service to layer four in the TCP/IP stack, represented mainly by the TCP and UDP protocols. This service is to take data that has been packaged by either TCP or UDP, manipulate it as necessary, and send it out. This service is sometimes called *internetwork datagram delivery*, as shown in Figure 54. As we will see, there are many details to how exactly this service is accomplished, but in a nutshell, that's what IP does: sends data from point A to point B over an internetwork of connected networks.

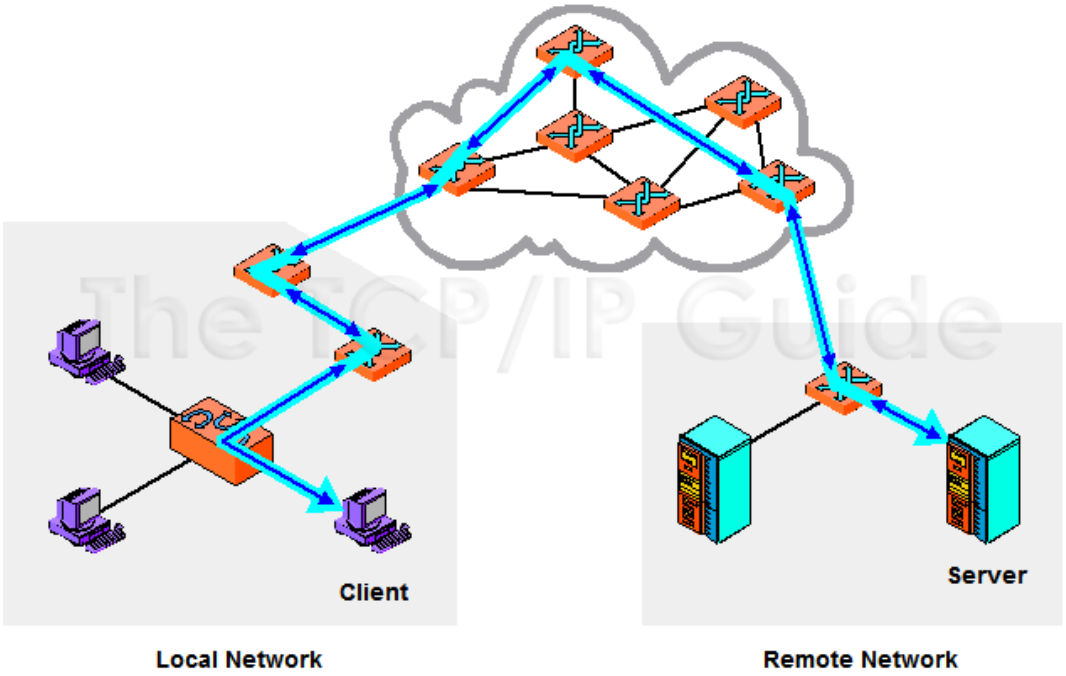


**Figure 54: The Main Function of IP: Internetwork Datagram Delivery**

The fundamental job of the Internet Protocol is the delivery of datagrams from one device to another over an internetwork. In this generic example, a distant client and server communicate with each other by passing IP datagrams over a series of interconnected networks.

**Key Concept:** While the Internet Protocol has many functions and characteristics, it can be boiled down to one primary purpose: the delivery of datagrams across an internetwork of connected networks.

### *Key IP Characteristics*

Of course there are a myriad of ways in which IP could have been implemented in order to accomplish this task. To understand how the designers of TCP/IP made IP work, let's take a look at the key characteristics used to describe IP and the general manner in which it operates. The Internet Protocol is said to be:

- **Universally-Addressed:** In order to send data from point A to point B, it is necessary to ensure that devices know how to identify which device is "point B". IP defines the addressing mechanism for the network and uses these addresses for delivery purposes.

- **Underlying-Protocol Independent:** IP is designed to allow the transmission of data across any type of underlying network that is designed to work with a TCP/IP stack. It includes provisions to allow it to adapt to the requirements of various lower-level protocols such as Ethernet or IEEE 802.11. IP can also run on the special data link protocols SLIP and PPP that were created for it. An important example is IP's ability to fragment large blocks of data into smaller ones to match the size limits of physical networks, and then have the recipient reassemble the pieces again as needed.

- **Delivered Connectionlessly:** IP is a *connectionless protocol*. This means that when A wants to send data to B, it doesn't first set up a connection to B and then send the data—it just makes the datagram and sends it. See the topic in the networking fundamentals section on connection-oriented and connectionless protocols for more information on this.

- **Delivered Unreliably:** IP is said to be an "unreliable protocol". That doesn't mean that one day your IP software will decide to go fishing rather than run your network. ☺ It does mean that when datagrams are sent from device *A* to device *B*, device *A* just sends each one and then moves on to the next. IP doesn't keep track of the ones it sent. It does not provide reliability or service quality capabilities such as error protection for the data it sends (though it does on the IP header), flow control or

retransmission of lost datagrams.

For this reason, IP is sometimes called a *best-effort* protocol. It does what it can to get data to where it needs to go, but “makes no guarantees” that the data will actually get there.

- **Delivered Without Acknowledgments:** In a similar manner to its unreliable nature, IP doesn't use acknowledgements. When device *B* gets a datagram from device *A*, it doesn't send back a “thank you note” to tell *A* that the datagram was received. It leaves device *A* “in the dark” so to speak.

### *IP's Success Despite Its Limitations*

The last three characteristics in the preceding list might be enough to make you cringe, thinking that giving your data to IP would be somewhat like trusting a new car to your sixteen-year-old son. If we are going to build our entire network around this protocol, why design it so that it works without connections, doesn't guarantee that the data will get there, and has no means of acknowledging receipt of data?

The reason is simple: establishing connections, guaranteeing delivery, error-checking and similar “insurance” functions have a cost: ***performance***. It takes time, computer resources and network bandwidth to perform these tasks, and they aren't always necessary for every application. Now, consider that IP carries pretty much ***all*** user traffic on a TCP/IP network. To build this complexity into IP would burden all traffic with this overhead whether it was needed or not.

The solution taken by the designers of TCP/IP was to exploit the power of layering. If service quality features such as connections, error-checking or guaranteed delivery are required by an application, they are provided at the transport layer (or possibly, the application layer). On the other hand, applications that don't need these features can avoid using them. This is in fact the major distinction between the two TCP/IP transport layer protocols: TCP and UDP. TCP is full-featured but a bit slower than UDP; UDP is spartan in its capabilities, but faster than TCP. This system is really the “best of both worlds”. And unlike your teenager with the shiny new license, it has been proven to work well in the real world. ☺

So how is datagram delivery accomplished by IP? In the following topic I discuss in more detail the main functions that IP performs to “get the job done”, so to speak.

# IP Datagram General Format

Data transmitted over an internet using IP is carried in messages called *IP datagrams*. Like all network protocol messages, IP uses a specific format for its datagrams. We are of course looking here at IP version 4 and so we will examine the IPv4 datagram format, which was defined in RFC 791 along with the rest of IPv4.

The IPv4 datagram is conceptually divided into two pieces: the *header* and the *payload*. The header contains addressing and control fields, while the payload carries the actual data to be sent over the internetwork. Unlike some message formats, IP datagrams do not have a footer following the payload.

Even though IP is a relatively simple, connectionless, “unreliable” protocol, the IPv4 header carries a fair bit of information, which makes it rather large. At a minimum, it is 20 bytes long, and with options can be significantly longer. The IP datagram format is described in Table 56 and illustrated in Figure 86.

**Table 56: Internet Protocol Version 4 (IPv4) Datagram Format**

| Field Name | Size (bytes) | Description |
|---|---|---|
| ***Version*** | 1/2 (4 bits) | ***Version:*** Identifies the version of IP used to generate the datagram. For IPv4, this is of course the number 4. The purpose of this field is to ensure compatibility between devices that may be running different versions of IP. In general, a device running an older version of IP will reject datagrams created by newer implementations, under the assumption that the older version may not be able to interpret the newer datagram correctly. |
| ***IHL*** | 1/2 (4 bits) | ***Internet Header Length (IHL):*** Specifies the length of the IP header, in 32-bit words. This includes the length of any options fields and padding. The normal value of this field when no options are used is 5 (5 32-bit words = 5*4 = 20 bytes). Contrast to the longer *Total Length* field below. |
| ***TOS*** | 1 | ***Type Of Service (TOS):*** A field designed to carry information to provide quality of service features, such as prioritized delivery, for IP datagrams. It was never widely used as originally defined, and its meaning has been subsequently redefined for use by a technique called *Differentiated Services (DS)*. See below for more information. |

<table>
<tr><td>TL</td><td>2</td><td>Total Length (TL): Specifies the total length of the IP datagram, in bytes. Since this field is 16 bits wide, the maximum length of an IP datagram is 65,535 bytes, though most are much smaller.</td></tr>
<tr><td>Identification</td><td>2</td><td>Identification: This field contains a 16-bit value that is common to each of the fragments belonging to a particular message; for datagrams originally sent unfragmented it is still filled in, so it can be used if the datagram must be fragmented by a router during delivery. This field is used by the recipient to reassemble messages without accidentally mixing fragments from different messages. This is needed because fragments may arrive from multiple messages mixed together, since IP datagrams can be received out of order from any device. See the discussion of IP message fragmentation.</td></tr>
<tr><td>Flags</td><td>3/8<br>(3 bits)</td><td>Flags: Three control flags, two of which are used to manage fragmentation (as described in the topic on fragmentation), and one that is reserved:
<table>
<tr><th>Subfield Name</th><th>Size (bytes)</th><th>Description</th></tr>
<tr><td>Reserved</td><td>1/8 (1 bit)</td><td>Reserved: Not used.</td></tr>
<tr><td>DF</td><td>1/8 (1 bit)</td><td>Don't Fragment: When set to 1, specifies that the datagram should not be fragmented. Since the fragmentation process is generally "invisible" to higher layers, most protocols don't care about this and don't set this flag. It is, however, used for testing the maximum transmission unit (MTU) of a link.</td></tr>
<tr><td>MF</td><td>1/8 (1 bit)</td><td>More Fragments: When set to 0, indicates the last fragment in a message; when set to 1, indicates that more fragments are yet to come in the fragmented message. If no fragmentation is used for a message, then of course there is only one "fragment" (the whole message), and this flag is 0. If fragmentation is used, all fragments but the last set this flag to 1 so the recipient knows when all fragments have been sent.</td></tr>
</table></td></tr>
<tr><td>Fragment Offset</td><td>1 5/8<br>(13 bits)</td><td>Fragment Offset: When fragmentation of a message occurs, this field specifies the offset, or position, in the overall message where the data in this fragment goes. It is specified in units of 8 bytes (64 bits). The first fragment has an offset of 0. Again, see the discussion of fragmentation for a description of how the field is used.</td></tr>
<tr><td>TTL</td><td>1</td><td>Time To Live (TTL): Short version: Specifies how long the datagram is allowed to "live" on the network, in</td></tr>
</table>

<table>
<tr><td></td><td></td><td>terms of router hops. Each router decrements the value of the TTL field (reduces it by one) prior to transmitting it. If the TTL field drops to zero, the datagram is assumed to have taken too long a route and is discarded.<br><br>See below for the longer explanation of TTL.</td></tr>
<tr><td>Protocol</td><td>1</td><td>Protocol: Identifies the higher-layer protocol (generally either a transport layer protocol or encapsulated network layer protocol) carried in the datagram. The values of this field were originally defined by the IETF "Assigned Numbers" standard, RFC 1700, and are now maintained by the Internet Assigned Numbers Authority (IANA):
<table>
<tr><th>Value (Hexadecimal)</th><th>Value (Decimal)</th><th>Protocol</th></tr>
<tr><td>00</td><td>0</td><td>Reserved</td></tr>
<tr><td>01</td><td>1</td><td>ICMP</td></tr>
<tr><td>02</td><td>2</td><td>IGMP</td></tr>
<tr><td>03</td><td>3</td><td>GGP</td></tr>
<tr><td>04</td><td>4</td><td>IP-in-IP Encapsulation</td></tr>
<tr><td>06</td><td>6</td><td>TCP</td></tr>
<tr><td>08</td><td>8</td><td>EGP</td></tr>
<tr><td>11</td><td>17</td><td>UDP</td></tr>
<tr><td>32</td><td>50</td><td>Encapsulating Security Payload (ESP) Extension Header</td></tr>
<tr><td>33</td><td>51</td><td>Authentication Header (AH) Extension Header</td></tr>
</table>
Note that the last two entries are used when IPSec inserts additional headers into the datagram: the AH or ESP headers.</td></tr>
<tr><td>Header Checksum</td><td>2</td><td>Header Checksum: A checksum computed over the header to provide basic protection against corruption in transmission. This is not the more complex CRC code typically used by data link layer technologies such as Ethernet; it's just a 16-bit checksum. It is calculated by dividing the header bytes into words (a word is two bytes) and then adding them together. The data is not checksummed, only the header. At each hop the device receiving the datagram does the same checksum calculation and on a mismatch, discards the datagram as damaged.</td></tr>
<tr><td>Source Address</td><td>4</td><td>Source Address: The 32-bit IP address of the originator of the datagram. Note that even though intermediate devices such as routers may handle the</td></tr>
</table>

| | | |
|---|---|---|
| | | datagram, they do not normally put their address into this field—it is always the device that originally sent the datagram. |
| ***Destination Address*** | 4 | ***Destination Address:*** The 32-bit IP address of the intended recipient of the datagram. Again, even though devices such as routers may be the intermediate targets of the datagram, this field is always for the ultimate destination. |
| ***Options*** | Variable | ***Options:*** One or more of several types of options may be included after the standard headers in certain IP datagrams. I discuss them in the topic that follows this one. |
| ***Padding*** | Variable | ***Padding:*** If one or more options are included, and the number of bits used for them is not a multiple of 32, enough zero bits are added to "pad out" the header to a multiple of 32 bits (4 bytes). |
| ***Data*** | Variable | ***Data:*** The data to be transmitted in the datagram, either an entire higher-layer message or a fragment of one. |

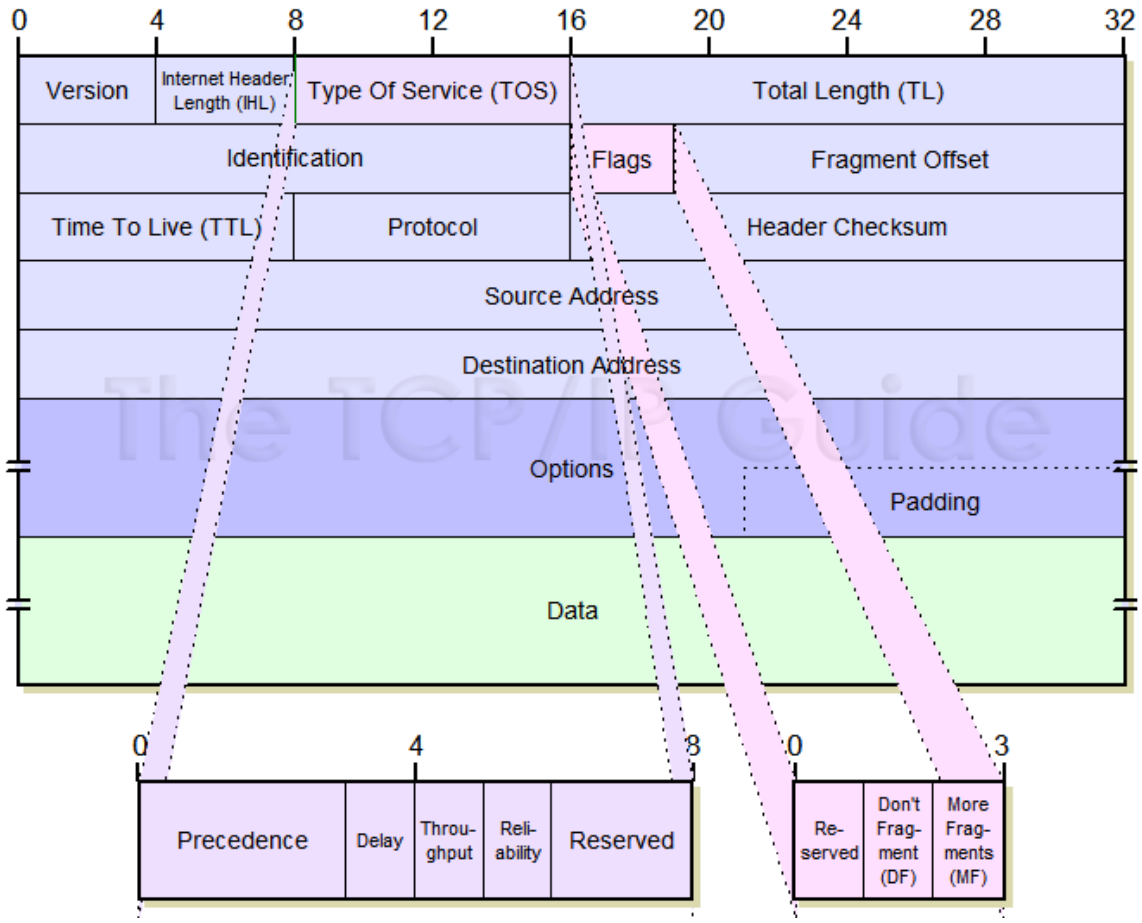


**Figure 86: Internet Protocol Version 4 (IPv4) Datagram Format**

This diagram shows graphically the all-important IPv4 datagram format. The first 20 bytes are the fixed IP header, followed by an optional *Options* section, and a variable-length *Data* area. Note that the *Type Of Service* field is shown as originally defined in the IPv4 standard.

That's a pretty big table, because the IP datagram format is pretty important and has a lot of fields that need explaining. To keep it from being even longer, I decided to move a couple of the more complex descriptions out of the table.

### *Time To Live (TTL) Field*

Since IP datagrams are sent from router to router as they travel across an internetwork, it is possible that a situation could result where a datagram gets passed from router A to router B to router C and then back to router A. Router loops are not supposed to happen, and rarely do, but are possible.

To ensure that datagrams don't circle around endlessly, the *TTL* field was intended to be filled in with a time value (in seconds) when a datagram was originally sent. Routers would decrease the time value periodically, and if it ever hit zero, the datagram would be destroyed. This was also intended to be used to ensure that time-critical datagrams wouldn't linger past the point where they would be "stale".

In practice, this field is not used in exactly this manner. Routers today are fast and usually take far less than a second to forward a datagram; measuring the time that a datagram "lives" would be impractical. Instead, this field is used as a "maximum hop count" for the datagram. Each time a router processes a datagram, it reduces the value of the *TTL* field by one. If doing this results in the field being zero, the datagram is said to have expired. It is dropped, and usually an ICMP *Time Exceeded* message is sent to inform the originator of the message that this happened.

The *TTL* field is one of the primary mechanisms by which networks are protected from router loops (see the description of ICMP Time Exceeded messages for more on how *TTL* helps IP handle router loops.)

***Type Of Service (TOS) Field***

This one-byte field was originally intended to provide certain quality of service features for IP datagram delivery. It allowed IP datagrams to be tagged with information indicating not only their precedence, but the preferred manner in which they should be delivered. It was divided into a number of subfields, as shown in Table 57 (and Figure 86).

The lack of quality of service features has been considered a weakness of IP for a long time. But as we can see in Table 57, these features were built into IP from the start. What's going on here? The answer is that even though this field was defined in the standard back in the early 1980s, it was not widely used by hardware and software. For years, it was just passed around with all zeroes in the bits and mostly ignored.

**Table 57: Original Definition Of IPv4 *Type Of Service (TOS)* Field**

| Subfield Name | Size (bytes) | Description |
|---|---|---|

<table>
<tr><td>Precedence</td><td>3/8<br>(3 bits)</td><td>Precedence: A field indicating the priority of the datagram. There were eight defined values, from lowest to highest priority:
<table>
<tr><th>Precedence Value</th><th>Priority Level</th></tr>
<tr><td>000</td><td>Routine</td></tr>
<tr><td>001</td><td>Priority</td></tr>
<tr><td>010</td><td>Immediate</td></tr>
<tr><td>011</td><td>Flash</td></tr>
<tr><td>100</td><td>Flash Override</td></tr>
<tr><td>101</td><td>CRITIC/ECP</td></tr>
<tr><td>110</td><td>Internetwork Control</td></tr>
<tr><td>111</td><td>Network Control</td></tr>
</table></td></tr>
<tr><td>D</td><td>1/8<br>(1 bit)</td><td>Delay: Set to 0 to request “normal” delay in delivery; set to 1 if low delay delivery is requested.</td></tr>
<tr><td>T</td><td>1/8<br>(1 bit)</td><td>Throughput: Set to 0 to request “normal” delivery throughput; set to 1 if higher throughput delivery is requested.</td></tr>
<tr><td>R</td><td>1/8<br>(1 bit)</td><td>Reliability: Set to 0 to request “normal” reliability in delivery; set to 1 if higher reliability delivery is requested.</td></tr>
<tr><td>Reserved</td><td>2/8<br>(2 bits)</td><td>Reserved: Not used.</td></tr>
</table>

The IETF, seeing the field unused, attempted to revive its use. In 1998, RFC 2474 redefines the first six bits of the *TOS* field to support a technique called *Differentiated Services (DS)*. Under DS, the values in the *TOS* field are called *codepoints* and are associated with different service levels. This starts to get rather complicated, so refer to RFC 2474 if you want all the details.

Understanding the IP datagram format is an important part of troubleshooting IP networks. Be sure to see the following topic on options for more information on how IP options are used in datagrams, and the topic on fragmenting for some more context on the use of fragmentation-related fields such as *Identification*, *Fragment Offset*, and *More Fragments*.

# IP Address Size, Address Space and "Dotted Decimal" Notation

Now that we have looked at the general issues and characteristics associated with IP addresses, it's time to get past the introductions and dig into the “meat” of

our IP address discussion. Let's start by looking at the physical construction and size of the IP address and how it is referred to and used.

***IP Address Size and Binary Notation***

At its simplest, the IP address is just a 32-bit binary number: a set of 32 ones or zeroes. At the lowest levels computers always work in binary and this also applies to networking hardware and software. While different meanings are ascribed to different bits in the address as we shall soon see, the address itself is just this 32-digit binary number.

Humans don't work too well with binary numbers, because they are long and complicated, and the use of only two digits makes them hard to differentiate. (Quick, which of these is larger: 11100011010100101001100110110001 or 11100011010100101001101110110001? ☺) For this reason, when we use IP addresses we don't work with them in binary except when absolutely necessary.

The first thing that humans would naturally do with a long string of bits is to split it into four eight-bit octets (or bytes, even though the two aren't technically the same), to make it more manageable. So, 11100011010100101001101110110001 would become “11100011 - 01010010 - 10011101 - 10110001”. Then, we could convert each of those octets into a more manageable two-digit hexadecimal number, to yield the following: “E3 - 52 - 9D - B1”. This is in fact the notation used for IEEE 802 MAC addresses, except that they are 48 bits long so they have six two-digit hex numbers, and they are usually separated by colons, not dashes as I used here.

***IP Address "Dotted Decimal" Notation***

Most people still find hexadecimal a bit difficult to work with. So IP addresses are normally expressed with each octet of 8 bits converted to a decimal number and the octets separated by a period (a “dot”). Thus, the example above would become *227.82.157.177*, as shown in Figure 56. This is usually called *dotted decimal notation* for rather obvious reasons. Each of the octets in an IP address can take on the values from 0 to 255 (not 1 to 256, note!) so the lowest value is theoretically 0.0.0.0 and the highest is 255.255.255.255.

**Key Concept:** IP addresses are 32-bit binary numbers, which can be expressed in binary, hexadecimal or decimal form. Most commonly, they are expressed by dividing the 32 bits into four bytes and converting each to decimal, then separating these numbers with dots to create *dotted decimal notation*.

Dotted decimal notation provides a convenient way to work with IP addresses when communicating amongst humans. Never forget that to the computers, the

IP address is always a 32-bit binary number; the importance of this will come in when we look at how the IP address is logically divided into components in the next topic, as well as when we examine techniques that manipulate IP addresses, such as subnetting.

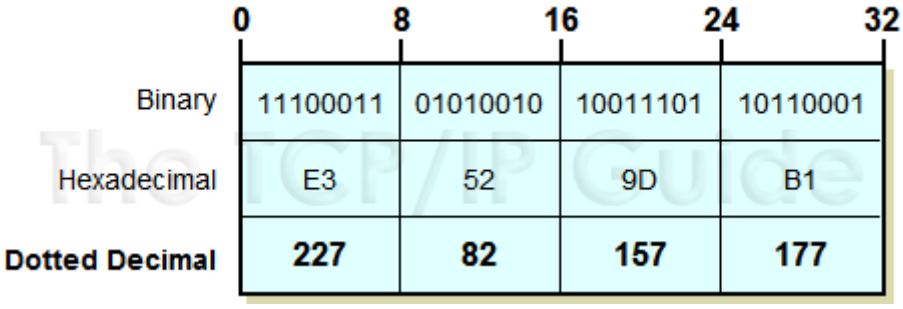


**Figure 56: IP Address Binary, Hexadecimal and Dotted Decimal Representations**

The binary, hexadecimal and decimal representations of an IP address are all equivalent.

### *IP Address Space*

Since the IP address is 32 bits wide, this provides us with a theoretical *address space* of $2^{32}$, or 4,294,967,296 addresses. This seems like quite a lot of addresses! And in some ways it is. However, as we will see, due to how IP addresses are structured and allocated, not every one of those addresses can actually be used. One of the unfortunate legacies of the fact that IP was originally created on a rather small internetwork is that decisions were made that "wasted" much of the address space. For example, all IP addresses starting with "127" in the first octet are reserved for the loopback function. Just this one decision makes 1/256th of the total number of addresses, or 16,277,216 addresses, no longer available. There are also other ways that the IP address space was not "conserved", which caused difficulty as the Internet grew in size. We'll see more about this in the section on "classful" addressing.

**Key Concept:** Since IP addresses are 32 bits long, the total *address space* of IPv4 is $2^{32}$ or 4,294,967,296 addresses. However, not all of these addresses can be used, for a variety of reasons.

This IP address space dictates the limit on the number of addressable interfaces in *each* IP internetwork. So, if you have a private network you can in theory have 4 billion plus addresses. However, in a public network such as the Internet, all devices must share the available address space. Techniques such as CIDR

("supernetting") and Network Address Translation (NAT) were designed in part to more efficiently utilize the existing Internet IP address space. Of course, IP version 6 expands the IP address size from 32 bits all the way up to 128, which increases the address space to a ridiculously large number and makes the entire matter of address space size moot.

(Incidentally, the second binary number is the larger one.)

# IP Basic Address Structure and Main Components: Network ID and Host ID

As I mentioned in the IP addressing overview, one of the ways that IP addresses are used is to facilitate the routing of datagrams in an IP internet. This is made possible because of the way that IP addresses are structured, and how that structure is interpreted by network routers.

***Internet IP Address Structure***

As we just saw, each version 4 IP address is 32 bits long. When we refer to the IP address we use a dotted-decimal notation, while the computer converts this into binary. However, even though these sets of 32 bits are considered a single "entity", they have an internal structure containing two components:

- **Network Identifier (Network ID):** A certain number of bits, starting from the left-most bit, is used to identify the network where the host or other network interface is located. This is also sometimes called the *network prefix* or even just the *prefix*.
- **Host Identifier (Host ID):** The remainder of the bits are used to identify the host on the network.

**Note:** By convention, IP devices are often called *hosts* for simplicity, as I do throughout this Guide. Even though each host ***usually*** has a single IP address, remember that IP addresses are strictly associated with network-layer network interfaces, not physical devices, and a device may therefore have more than one IP address.

As you can see in Figure 57, this really is a fairly simple concept; it's the same idea as the structure used for phone numbers in North America. The telephone number (401) 555-7777 is a ten-digit number usually referred to as a single "phone number". However, it has a structure. In particular, it has an area code ("401") and a local number ("555-7777").

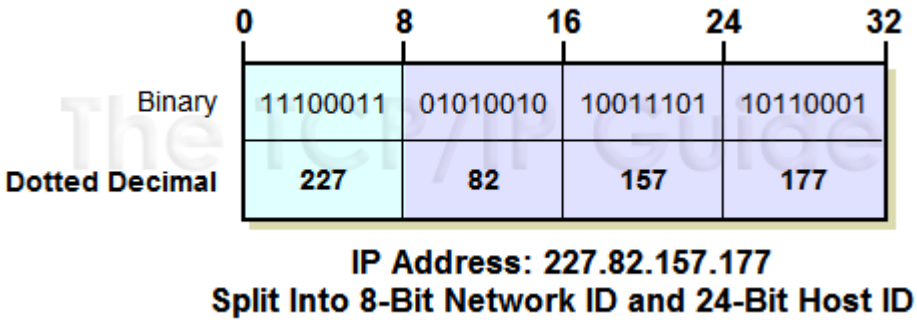


**Figure 57: Basic IP Address Division: Network ID and Host ID**

The fundamental division of the bits of an IP address is into a network ID and host ID. Here, the network ID is 8 bits long, shown in cyan, and the host ID is 24 bits in length.

### *Implications of Including the Network ID in IP Addresses*

The fact that the network identifier is contained in the IP address is what partially facilitates the routing of IP datagrams when the address is known. Routers look at the network portion of the IP address to determine first of all if the destination IP address is on the same network as the host IP address. Then routing decisions are made based on information the routers keep about where various networks are located. Again, this is conceptually similar to how the area code is used by the equivalent of “routers” in the phone network to switch telephone calls. The host portion of the address is used by devices on the local portion of the network.

Since the IP address can be split into network ID and host ID components, it is also possible to use either one or the other by itself, depending on context. These addresses are assigned special meanings. For example, if the network ID is used with all ones as the host ID, this indicates a broadcast to the entire network. Similarly, if the host ID is used by itself with all zeroes for the network ID, this implies an IP address sent to the host of that ID on “the local network”, whatever that might be.

It is the inclusion of the network identifier in the IP address of each host on the network that causes the IP addresses to be network-specific. If you move a device from one network to a different one the network ID must change to that of the new network. Therefore, the IP address must change as well. This is an unfortunate drawback that shows up most commonly when dealing with mobile devices.

### *Location of the Division Between Network ID and Host ID*

One difference between IP addresses and phone numbers is that the dividing point between the bits used to identify the network and those that identify the host isn't fixed. It depends on the nature of the address, the type of addressing being used, and other factors. Let's take the example from the last topic, 227.82.157.177. It is possible to divide this into a network identifier of “227.82” and a host identifier of “157.177”. Alternately, the network identifier might be “227” and the host identifier “82.157.177” within that network.

To express the network and host identifiers as 32-bit addresses, we add zeroes to replace the missing “pieces”. In the latter example just above, the address of the network becomes “227.0.0.0” and the address of the host “0.82.157.177”. (In practice, network addresses of this sort are routinely seen with the added zeroes; network IDs are not as often seen in 32-bit form this way.)

Lest you think from these examples that the division must always be between whole octets of the address, it's also possible to divide it in the middle of an octet. For example, we could split the IP address 227.82.157.177 so there were 20 bits for the network ID and 12 bits for the host ID. The process is the same, but determining the dotted decimal ID values is more tricky because here, the “157” is “split” into two binary numbers. The results are “227.82.144.0” for the network ID and “0.0.0.13.177” for the host ID, as shown in Figure 58.

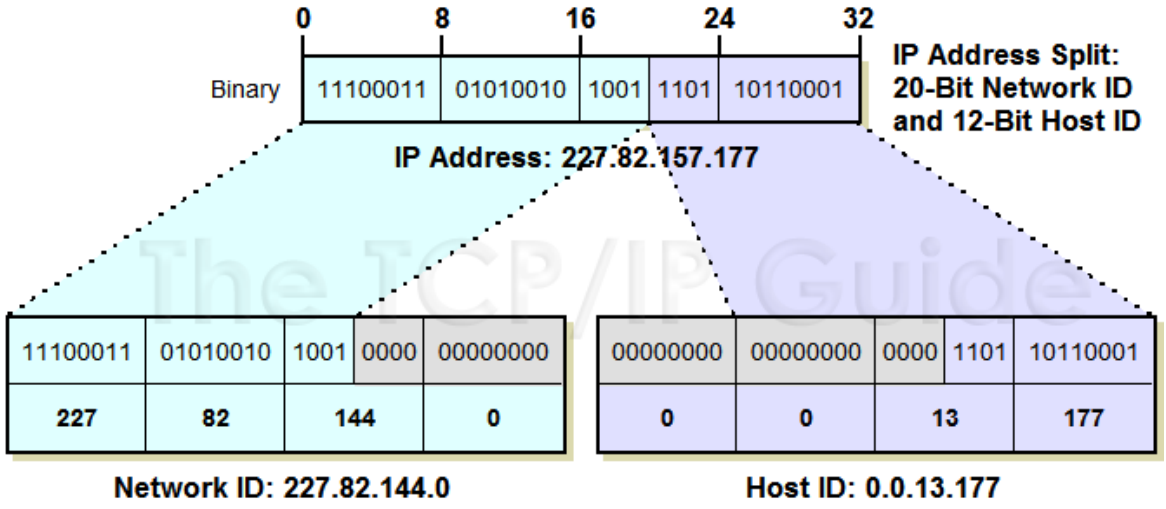


**Figure 58: Mid-Octet IP Address Division**

Since IP addresses are normally expressed as four dotted-decimal numbers, educational resources often show the division between the Network ID and Host ID occurring on an octet boundary. However, it’s essential to remember that the dividing point often appears in the middle of one of these eight-bit numbers. In this example, the Network ID is 20 bits long and the Host ID 12 bits long. This

results in the third number of the original IP address, 157, being split into 144 and 13.

The place where the “line is drawn” between the network ID and the host ID must be known in order for devices such as routers to know how to interpret the address. This information is conveyed either implicitly or explicitly depending on the type of IP addressing in use. I describe this in the following topic.

**Key Concept:** The basic structure of an IP address consists of two components: the *network ID* and *host ID*. The dividing point of the 32-bit address is not fixed, but rather, depends on a number of factors, and can occur in a variety of places, including in the middle of a dotted-decimal octet.

# IP Addressing Categories (Classful, Subnetted and Classless) and IP Address Adjuncts (Subnet Mask and Default Gateway)

The preceding topic illustrated how the fundamental division of the 32 bits in an IP address is into the network identifier (network ID) and host identifier (host ID). The network ID is used for routing purposes while the host ID uniquely identifies each network interface on the network. In order for devices to know how to use IP addresses on the network they must be able to tell which bits are used for each ID. However, the “dividing line” is not predefined. It depends on the type of addressing used in the network.

***IP Addressing Scheme Categories***

Understanding how these IDs are determined leads us into a larger discussion of the three main categories of IP addressing schemes. Each of these uses a slightly different system of indicating where in the IP address the host ID is found.

**Conventional (“Classful”) Addressing**

The original IP addressing scheme is set up so that the dividing line occurs only in one of a few locations: on octet boundaries. Three main classes of addresses, A, B and C are differentiated based on how many octets are used for the network ID and how many for the host ID. For example, class C addresses devote 24 bits to the network ID and 8 to the host ID. This type of addressing is now often referred to by the made-up word “classful” to differentiate it from newer “classless” scheme.

This most basic addressing type uses the simplest method to divide the network and host identifiers: the class, and therefore the dividing point, are encoded into the first few bits of each address. Routers can tell from these bits which octets belong to which identifier.

**Subnetted "Classful" Addressing**

In the subnet addressing system, the two-tier network/host division of the IP address is made into a three-tier system by taking some number of bits from a class A, B or C host ID and using them for a *subnet identifier*. The network ID is unchanged. The *subnet ID* is used for routing within the different subnetworks that constitute a complete network, providing extra flexibility for administrators. For example, consider a class C address that normally uses the first 24 bits for the network ID and remaining 8 bits for the host ID. The host ID can be split into, say, 3 bits for a subnet ID and 5 for the host ID.

This system is based on the original "classful" scheme, so the dividing line between the network ID and "full" host ID is based on the first few bits of the address as before. The dividing line between the subnet ID and the "sub-host" ID is indicated by a 32-bit number called a *subnet mask*. In the example above, the subnet mask would be 27 ones followed by 5 zeroes—the zeroes indicate what part of the address is the host. In dotted decimal notation, this would be 255.255.255.224.

**Classless Addressing**

In the classless system, the classes of the original IP addressing scheme are tossed out the window. The division between the network ID and host ID can occur at an arbitrary point, not just on octet boundaries like in the "classful" scheme.

The dividing point is indicated by putting the number of bits used for the network ID, called the *prefix length*, after the address (recall that the network ID bits are also sometimes called the *network prefix*, so the network ID size is the prefix length). For example, if 227.82.157.177 is part of a network where the first 27 bits are used for the network ID, that network would be specified as 227.82.157.160/27. The "/27" is conceptually the same as the 255.255.255.224 subnet mask, since it has 27 one bits followed by 5 zeroes.

**Key Concept:** An essential factor in determining how an IP address is interpreted is the addressing scheme in which it is used. The three methods, arranged in increasing order of age, complexity and flexibility, are *"classful" addressing*, *subnetted "classful" addressing*, and *classless addressing*.

Did I just confuse the heck out of you? Sorry—and don't worry. I'm simply introducing the concepts of “classful”, subnetted and classless addressing and showing you how they impact the way the IP address is interpreted. This means of necessity that I have greatly summarized important concepts here. All three methods are explained in their own sections in full detail.

### *IP Address Adjuncts: Subnet Mask and Default Gateway*

As you can see, in the original “classful” scheme the division between network ID and host ID is implied. However, if either subnetting or classless addressing is used, then the subnet mask or “slash number” are required to fully qualify the address. These numbers are considered adjuncts to the IP address and usually mentioned “in the same breath” as the address itself, because without them, it is not possible to know where the network ID ends and the host ID begins.

One other number that is often specified along with the IP address for a device is the *default gateway* identifier. In simplest terms, this is the IP address of the router that provides default routing functions for a particular device. When a device on an IP network wants to send a datagram to a device it can't see on its local IP network, it sends it to the default gateway which takes care of routing functions. Without this, each IP device would also have to have knowledge of routing functions and routes, which would be inefficient. See the sections on routing concepts and TCP/IP routing protocols for more information.

## IP "Classful" Addressing Network and Host Identification and Address Ranges

The “classful” IP addressing scheme divides the total IP address space into five classes, A through E. One of the benefits of the relatively simple “classful” scheme is that information about the classes is encoded directly into the IP address. This means we can determine in advance which address ranges belong to each class. It also means the opposite is possible: we can identify which class is associated with any address by examining just a few bits of the address.

This latter benefit was one of the main motivators for the initial creation of the “classful” system, as we saw in the previous topic.

### *"Classful" Addressing Class Determination Algorithm*

When TCP/IP was first created computer technology was still in its infancy, compared to its current state. Routers needed to be able to quickly make decisions about how to move IP datagrams around. The IP address space was split into classes in a way that looking at only the first few bits of any IP address

would tell the router where to “draw the line” between the network ID and host ID, and thus what to do with the datagram.

The number of bits the router needs to look at may be as few as one or as many as four, depending on what it finds when it starts looking. The algorithm used corresponds to the system used to divide the address space; it involves four very basic steps (see Figure 61):

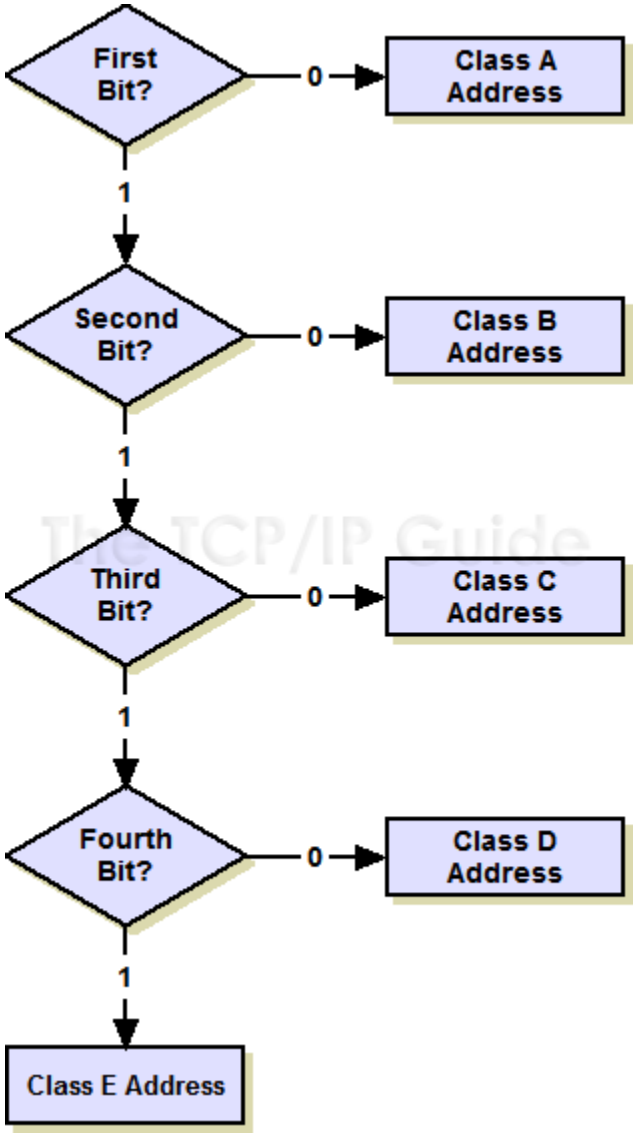


**Figure 61: Class Determination Algorithm for “Classful” IP Addresses**

The simplicity of the “classful” IP addressing can be seen in the very

uncomplicated algorithm used to determine the class of an address.

1. If the first bit is a “0”, it's a class A address and we're done. (Half the address space has a “0” for the first bit, so this is why class A takes up half the address space.) If it's a “1”, continue to step two.

2. If the second bit is a “0”, it's a class B address and we're done. (Half of the remaining non-class-A addresses, or one quarter of the total.) If it's a “1”, continue to step three.

3. If the third bit is a “0”, it's a class C address and we're done. (Half again of what's left, or one eighth of the total.) If it's a “1”, continue to step four.

4. If the fourth bit is a “0”, it's a class D address. (Half the remainder, or one sixteenth of the address space.) If it's a “1”, it's a class E address. (The other half, one sixteenth.)

And that's pretty much it.

***Determining Address Class From the First Octet Bit Pattern***

As humans, of course, we generally work with addresses in dotted decimal notation and not in binary, but it's pretty easy to see the ranges that correspond to the classes. For example, consider class B. The first two bits of the first octet are “10”. The remaining bits can be any combination of ones and zeroes. This is normally represented as “10xx xxxx” (shown as two groups of four for readability.) Thus, the binary range for the first octet can be from “**10**00 0000” to “**10**11 1111”. This is 128 to 191 in decimal. So, in the “classful” scheme, any IP address whose first octet is from 128 to 191 (inclusive) is a class B address.

In Table 44 I have shown the bit patterns of each of the five classes, and the way that the first octet ranges can be calculated. In the first column is the format for the first octet of the IP address, where the “x”s can be either a zero or a one. Then I show the lowest and highest value for each class in binary (the “fixed” few bits are highlighted so you can see that they do not change while the others do.) I then also show the corresponding range for the first octet in decimal.

**Table 44: IP Address Class Bit Patterns, First-Octet Ranges and Address Ranges**

| IP Address | First Octet of | Lowest Value | Highest Value | Range of First | Octets in | Theoretical IP Address Range |
|---|---|---|---|---|---|---|

| Class | IP Address | of First Octet (binary) | of First Octet (binary) | Octet Values (decimal) | Network ID / Host ID | |
|---|---|---|---|---|---|---|
| **Class A** | **0**xxx xxxx | **0**000 0001 | **0**111 1110 | 1 to 126 | 1 / 3 | **1**.0.0.0 to **126**.255.255.255 |
| **Class B** | **10**xx xxxx | **10**00 0000 | **10**11 1111 | 128 to 191 | 2 / 2 | **128.0**.0.0 to **191.255**.255.255 |
| **Class C** | **110**x xxxx | **110**0 0000 | **110**1 1111 | 192 to 223 | 3 / 1 | **192.0.0**.0 to **223.255.255**.255 |
| **Class D** | **1110** xxxx | **1110** 0000 | **1110** 1111 | 224 to 239 | — | 224.0.0.0 to 239.255.255.255 |
| **Class E** | **1111** xxxx | **1111** 0000 | **1111** 1111 | 240 to 255 | — | 240.0.0.0 to 255.255.255.255 |

**Key Concept:** In the “classful” IP addressing scheme, the class of an IP address is identified by looking at the first one, two, three or four bits of the address. This can be done both by humans working with these addresses and routers making routing decisions. The use of these bit patterns means that IP addresses in different classes fall into particular address ranges that allow an address’s class to be determined by looking at the first byte of its dotted-decimal address.

### *Address Ranges for Address Classes*

I have also shown in Table 44 the *theoretical* lowest and highest IP address ranges for each of the classes. This means that the address ranges shown are just a result of taking the full span of binary numbers possible in each class. In reality, some of the values are not available for normal use. For example, even though 192.0.0.0 to 192.0.0.255 is technically in class C, it is reserved and not actually used by hosts on the Internet.

Also, there are IP addresses that can't be used because they have special meaning. For example, you can't use an IP address of 255.255.255.255, as this is a reserved “all ones” broadcast address. In a similar vein, note that the range for Class A is from 1 to 126 and not 0 to 127 like you might have expected. This is because class A networks 0 and 127 are reserved; 127 is the network containing the IP loopback address. These special and reserved addresses are discussed later in this section.

Now, recall that classes A, B and C differ in where the dividing line is between the network ID and the host ID: 1 for network and 3 for host for class A, 2 for each for class B, and 3 for network and 1 for host for class C. Based on this division, I have highlighted the network ID portion of the IP address ranges for each of classes A, B and C. The plain text corresponds to the range of host IDs for each allowable network ID. Figure 62 shows graphically how bits are used in each of the five classes.

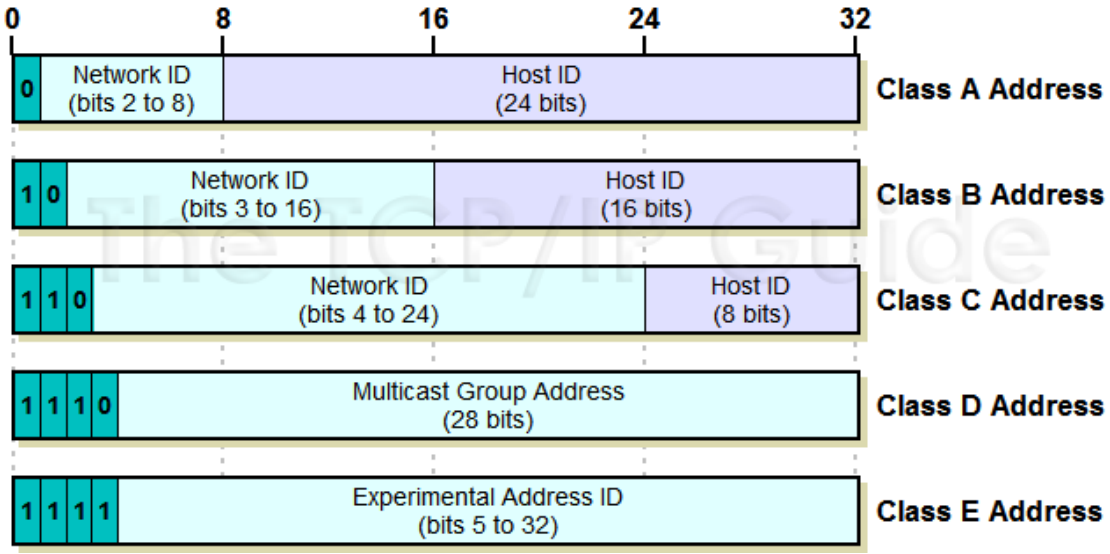


**Figure 62: IP Address Class Bit Assignments and Network/Host ID Sizes**

This illustration shows how the 32 bits of IP address are assigned for each of the five IP address classes. Classes A, B and C are the “normal” classes used for regular unicast addresses; each has a different dividing point between the Network ID and Host ID. Classes D and E are special and are not divided in this manner.

Phew, time for another example methinks. Let's look at class C. The lowest IP address is **192.0.0.**0 and the highest is **223.255.255.**255. The first three octets are the network ID, and can range from **192.0.0** to **223.255.255**. For each network ID in that range, the host ID can range from 0 to 255.

**Note:** It is common to see resources refer to the network ID of a “classful” address as including only the “significant” bits, that is, only the ones that are not common to all networks of that class. For example, you may see a Class B network ID shown in a diagram as having 14 bits, with the “**10**” that starts all such networks shown separately, as if it were not part of the network ID. Remember that the network ID ***does*** include those bits as well; it is 8 full bits for

Class A, 16 for Class B and 24 for Class C. In the case of Class D addresses, all 32 bits are part of the address, but only the lower 28 bits are part of the multicast group address; see the topic on multicast addressing for more.

# IP Address Class A, B and C Network and Host Capacities

In the preceding topics I introduced the concepts of IP address classes and showed how the classes related to ranges of IP addresses. Of the five classes, D and E are dedicated to special purposes, so I will leave those alone for now. Classes A, B and C are the ones actually assigned for normal (unicast) addressing purposes on IP internetworks, and therefore the primary focus of our continued attention.

As we've seen, they differ in the number of bits (and octets) used for the network ID compared to the host ID. The number of different networks possible in each class is a function of the number of bits assigned to the network ID, and likewise, the number of hosts possible in each network depends on the number of bits provided for the host ID. We must also take into account the fact that one, two or three of the bits in the IP address is used to indicate the class itself, so it is effectively "excluded" from use in determining the number of networks (though again, it is still part of the network ID).

Based on this information, we can calculate the number of networks in each class, and for each class, the number of host IDs per network. Table 45 shows the calculations.

**Table 45: IP Address Class Network and Host Capacities**

| IP Address Class | Total # Of Bits For Network ID / Host ID | First Octet of IP Address | # Of Network ID Bits Used To Identify Class | Usable # Of Network ID Bits | Number of Possible Network IDs | # Of Host IDs Per Network ID |
|---|---|---|---|---|---|---|
| **Class A** | 8 / 24 | **0**xxx xxxx | 1 | 8-1 = 7 | $2^7$-2 = 126 | $2^{24}$-2 = 16,277,214 |
| **Class B** | 16 / 16 | **10**xx xxxx | 2 | 16-2 = 14 | $2^{14}$ = 16,384 | $2^{16}$-2 = 65,534 |
| **Class C** | 24 / 8 | **110**x xxxx | 3 | 24-3 = 21 | $2^{21}$ = 2,097,152 | $2^8$-2 = 254 |

Let's walk through one line of this table so we can see how it works. I'll stick with class B since it's "in the middle". The basic division is into 16 bits for network ID and 16 bits for host ID. However, the first two bits of all class B addresses must be "10", so that leaves only 14 bits to uniquely identify the network ID. This gives us a total of $2^{14}$ or 16,384 class B network IDs. For each of these, we have $2^{16}$ host IDs, ***less two***, for a total of 65,534.

Why less two? For each network ID, two host IDs cannot be used: the host ID with all zeroes and the ID with all ones. These are addresses with "special meanings" as described in the topic that follows. You will also notice that 2 has been subtracted from the number of network IDs for class A. This is because two of the class A network IDs (0 and 127) are reserved. There are actually several other address ranges that are set aside in all three of the classes that I haven't shown here. They are listed in the topic on reserved, private and loopback addresses. (The exclusion of 0 and 127 from class A is probably the best-known address range reservation which is why I am explicit with that one in the table above.)

**Key Concept:** In the "classful" IP addressing scheme, a Class A network contains addresses for about 16 million network interfaces; a Class B about 65,000; and a Class C, 254.

As you can see, there is quite a disparity in the number of hosts available for each network in each of these classes. What happens if an organization needs 1,000 IP addresses? They have to either use four class Cs or use one class B (and in so doing waste over 90% of the possible addresses in the class B network.) Bear in mind that there are only about 16,000 class B network IDs available worldwide and you begin to understand one of the big problems with "classful" addressing.