Chapter 2. IP, ARP, ICMP

2.1 Overview

The best place to start looking at TCP/IP is probably the name itself. TCP/IP in fact consists of dozens of different protocols, but only a few are the “main” protocols that define the core operation of the suite. Of these key protocols, two are usually considered the most important. The Internet Protocol (IP) is the primary OSI network layer (layer three) protocol that provides addressing, datagram routing and other functions in an internetwork. The Transmission Control Protocol (TCP) is the primary Transport Layer (layer four) protocol, and is responsible for connection establishment and management and reliable data transport between software processes on devices. The User Datagram Protocol (UDP) is also located at Transport Layer being one of the core members of the Internet protocol suite (the set of network protocols used for the Internet). With UDP, computer applications can send messages, in this case referred to as datagrams, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. UDP is suitable for purposes where error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level.

The main design goal of TCP/IP was to build an interconnection of networks, referred to as an internetwork, or internet, that provided universal communication services over heterogeneous physical networks. The clear benefit of such an internetwork is the enabling of communication between hosts on different networks, perhaps separated by a large geographical area.

The words internetwork and internet are simply a contraction of the phrase interconnected network. However, when written with a capital “I”, the Internet refers to the worldwide set of interconnected networks. Therefore, the Internet is an internet, but the reverse does not apply. The Internet is sometimes called the connected Internet.

The Internet consists of the following groups of networks:

- Backbones: Large networks that exist primarily to interconnect other networks. Also known as network access points (NAPs) or Internet Exchange Points (IXPs). Currently, the backbones consist of commercial entities.
- Regional networks connecting, for example, universities and colleges.

- Commercial networks providing access to the backbones to subscribers, and networks owned by commercial organizations for internal use that also have connections to the Internet.
- Local networks, such as campus-wide university networks.

In most cases, networks are limited in size by the number of users that can belong to the network, by the maximum geographical distance that the network can span, or by the applicability of the network to certain environments. For example, an Ethernet network is inherently limited in terms of geographical size. Therefore, the ability to interconnect a large number of networks in some hierarchical and organized fashion enables the communication of any two hosts belonging to this internetwork.

Another important aspect of TCP/IP internetworking is the creation of a standardized abstraction of the communication mechanisms provided by each type of network. Each physical network has its own technology-dependent communication interface, in the form of a programming interface that provides basic communication functions (primitives). TCP/IP provides communication services that run between the programming interface of a physical network and user applications. It enables a common interface for these applications, independent of the underlying physical network. The architecture of the physical network is therefore hidden from the user and from the developer of the application. The application need only code to the standardized communication abstraction to be able to function under any type of physical network and operating platform.

As is evident in Figure 2.1, to be able to interconnect two networks, we need a computer that is attached to both networks and can forward data packets from one network to the other; such a machine is called a router. The term IP router is also used because the routing function is part of the Internet Protocol portion of the TCP/IP protocol suite.

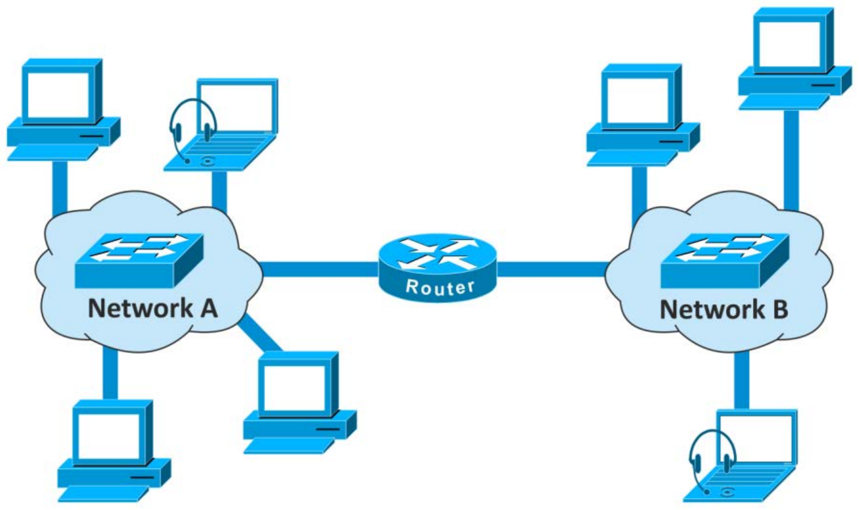


Fig. 2.1 Networks interconnected by a router

2.2 TCP/IP Four Layers Architecture Model

TCP/IP architecture does not exactly follow the OSI model. Unfortunately, there is no universal agreement regarding how to describe TCP/IP with a layered model. It is generally agreed that TCP/IP has fewer levels (from three to five layers) than the seven layers of the OSI model. Hereby it is presented a four layers model for the TCP/IP architecture.
TCP/IP architecture omits some features found under the OSI model, combines the features of some adjacent OSI layers and splits other layers apart. The 4-layer structure of TCP/IP is built as information is passed down from applications to the physical network layer. When data is sent, each layer treats all of the information it receives from the upper layer as data, adds control information (header) to the front of that data and then pass it to the lower layer. When data is received, the opposite procedure takes place as each layer processes and removes its header before passing the data to the upper layer. The TCP/IP 4-layer model and the key functions of each layer is described below (Fig. 2.2).

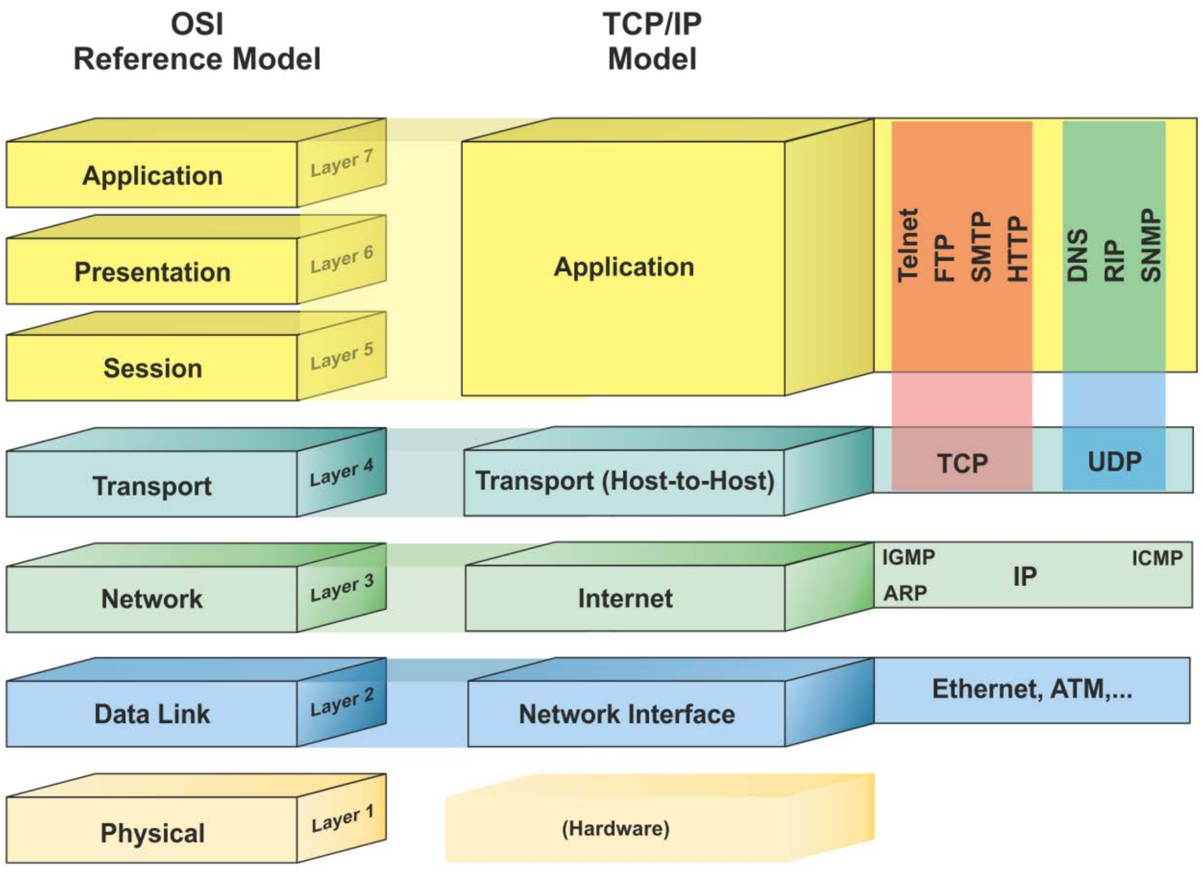


Fig. 2.2. TCP/IP Four Layers Architecture Model

Application Layer

The Application Layer in TCP/IP groups the functions of OSI Application, Presentation Layer and Session Layer. Therefore any process above the transport layer is called an Application in the TCP/IP architecture. In TCP/IP socket and port are used to describe the path over which applications communicate. Most application level protocols are associated with one or more port number. The application layer is provided by the program that uses TCP/IP for communication. An application is a user process cooperating with another process usually on a different host (there is also a benefit to application communication within a single host). Examples of applications include Telnet, File Transfer Protocol (FTP), e-mail (SMTP), World Wide Web (http) and the Domain Name System. The interface between the application and transport layers is defined by port numbers and sockets.

Transport Layer
The transport layer provides the end-to-end data transfer by delivering data from an application to its remote peer. Multiple applications can be supported simultaneously. The most-used transport layer protocol is the Transmission Control Protocol (TCP), which provides connection-oriented reliable data delivery, duplicate data suppression, congestion control, and flow control. Another transport layer protocol is the User Datagram Protocol. It provides connectionless, unreliable, best-effort service. As a result, applications using UDP as the transport protocol have to provide their own end-to-end integrity, flow control, and congestion control, if desired. Usually, UDP is used by applications that need a fast transport mechanism and can tolerate the loss of some data.

Network Layer
The network layer, also called the internetwork layer or the internet layer, provides the "virtual network" image of an internet (this layer shields the higher levels from the physical network architecture below it). Internet Protocol (IP) is the most important protocol in this layer. It is a connectionless protocol that does not assume reliability from lower layers. IP does not provide reliability, flow control, or error recovery. These functions must be provided at a higher level.

IP provides a routing function that attempts to deliver transmitted messages to their destination. A message unit in an IP network is called an IP datagram. This is the basic unit of information transmitted across TCP/IP networks. Other internetwork-layer protocols are IP, ICMP, IGMP, ARP, and RARP.

Network Interface Layer
The network interface layer, also called the link layer or the data-link layer, is the interface to the actual network hardware. This interface may or may not provide reliable delivery, and may be packet or stream oriented. In fact, TCP/IP does not specify any protocol here, but can use almost any network interface available, which illustrates the flexibility of the IP layer. Examples are Ethernet, Token Ring, IEEE 802.2, X.25 (which is reliable in itself), HSSI, ATM, FDDI, and even SNA.

2.3 Internet Protocol (IP)

IP is the protocol that hides the underlying physical network by creating a virtual network view. It is an unreliable, best-effort, and connectionless packet delivery protocol. Note that best-effort means that the packets sent by IP might be lost, arrive out of order, or even be duplicated. IP assumes higher layer protocols will address these anomalies.

One of the reasons for using a connectionless network protocol was to minimize the dependency on specific computing centers that used hierarchical connection-oriented networks. The United States Department of Defense intended to deploy a network that would still be operational if parts of the country were destroyed. This has been proven to be true for the Internet.

IP has been around for more than three decades. Perhaps the easiest and best place to start is with RFC 791, titled "Internet Protocol DARPA Internet Protocol Specification." This RFC was written in 1981 and is based on six earlier editions of the ARPA Internet Protocol Specification. This early document also describes IP as the protocol that "provides for transmitting blocks of data called datagrams from sources to destinations." Today, we use the terms packets and datagrams interchangeably, but the goal is the same. Every time a node connected to a network tries to communicate with another node, the transmission is broken up into these datagrams or packets. For example, the request to see a webpage and the delivery of the web content returned to the desktop are accomplished via IP packets. The size and number of the packets depends on the amount of information. The device primarily responsible for getting these packets to the correct destination is a router. This also means that every single IP packet must have all of the information necessary to be routed independently from all other packets.

Internet Protocol version 6 (IPv6) is the new generation of the basic protocol of the Internet. IP is the common language of the Internet, every device connected to the Internet must support it. The current version of IP (IP version 4) has several shortcomings which complicate, and in some cases present a barrier to, the further development of the Internet. The coming IPv6 revolution should remove these barriers and provide a feature-rich environment for the future of global networking.

2.3.1 IP version 4

2.3.1.1 IP addressing

This paragraph deals with the structure of network-layer addresses used in the Internet, also known as IP addresses. It is presented how addresses are allocated and assigned to devices on the Internet, the way hierarchy in address assignment aids routing scalability, and the use of special-purpose addresses, including broadcast, multicast, and anycast addresses. It is also presented how the structure and use of IPv4 and IPv6 addresses differ.

Every device connected to the Internet has at least one IP address. Devices used in private networks based on the TCP/IP protocols also require IP addresses. In either case, the forwarding procedures implemented by IP routers use IP addresses to identify where traffic is going. IP addresses also indicate where traffic has come from. IP addresses are similar in some ways to telephone numbers, but whereas telephone numbers are often known and used directly by end users, IP addresses are often shielded from a user's view by the Internet's DNS, which allows most users to use names instead of numbers. Users are confronted with manipulating IP addresses when they are required to set up networks themselves or when the DNS has failed for some reason. To understand how the Internet identifies hosts and routers and delivers traffic between them, we must understand the role of IP addresses. We are therefore interested in their administration, structure, and uses.

When devices are attached to the global Internet, they are assigned addresses that must be coordinated so as to not duplicate other addresses in use on the network. For private networks, the IP addresses being used must be coordinated to avoid similar overlaps within the private networks. Groups of IP addresses are allocated to users and organizations. The recipients of the allocated addresses then assign addresses to devices, usually according to some network “numbering plan.” For global Internet addresses, a hierarchical system of administrative entities helps in allocating addresses to users and service providers.

Individual users typically receive address allocations from Internet service providers (ISPs) that provide both the addresses and the promise of routing traffic in exchange for a fee.

2.3.1.2 Expressing IP Addresses

IPv4 addresses are represented by a 32-bit unsigned binary value. The vast majority of Internet users who are familiar with IP addresses understand the most popular type: IPv4 addresses. Such addresses are often represented in so-called dotted-quad or dotted-decimal notation, for example, 165.195.130.107. The dotted-quad notation consists of four decimal numbers separated by periods. Each such number is a nonnegative integer in the range [0, 255] and represents one quarter of the entire IP address. The dotted-quad notation is simply a way of writing the whole IPv4 address—a 32-bit nonnegative integer used throughout the Internet system—using convenient decimal numbers. In many circumstances we will be concerned with the binary structure of the address. A number of Internet sites now contain calculators for converting between formats of IP addresses and related information. Table 2-1 gives a few examples of IPv4 addresses and their corresponding binary representations, to get started.

Dotted-Decimal Representation	Binary Representation
0.0.0.0	00000000 00000000 00000000 00000000
10.0.0.255	00001010 00000000 00000000 11111111
193.226.37.1	11000001 11100010 00100101 00000001
255.255.255.240	11111111 11111111 11111111 11110000

Table 2-1. Example IPv4 addresses

2.3.1.3 Basic IP Address Structure

IPv4 has more than 4×10^9 possible addresses in its address space. Because of the large number of addresses, it is convenient to divide the address space into chunks. IP addresses are grouped by type and size. Most of the IPv4 address chunks are eventually subdivided down to a single address and used to identify a single network interface of a computer attached to the Internet or to some private intranet. These addresses are called unicast addresses. Most of the IPv4 address space is unicast address space. Most of the IPv6 address space is not currently being used. Beyond unicast addresses, other types of addresses include broadcast, multicast, and anycast, which may refer to more than one interface, plus some special-purpose addresses we will discuss later. Before we begin with the details of the current address structure, it is useful to understand the historical evolution of IP addresses.

2.3.1.4 Class-based IP addresses

When the Internet's address structure was originally defined, every unicast IP address had a network portion, to identify the network on which the interface using the IP address was to be found, and a host portion, used to identify the particular host on the network given in the network portion. Thus, some number of contiguous bits in the address became known as the net number, and remaining bits were known as the host number. At the time, most hosts had only a single network interface, so the terms interface address and host address were used somewhat interchangeably.

With the realization that different networks might have different numbers of hosts, and that each host requires a unique IP address, a partitioning was devised wherein different-size allocation units of IP address space could be given out to different sites, based on their current and projected number of hosts. The partitioning of the address space involved five classes. Each class represented a different trade-off in the number of bits of a 32-bit IPv4 address devoted to the network number versus the number of bits devoted to the host number. Figure 2.3 shows the basic idea.

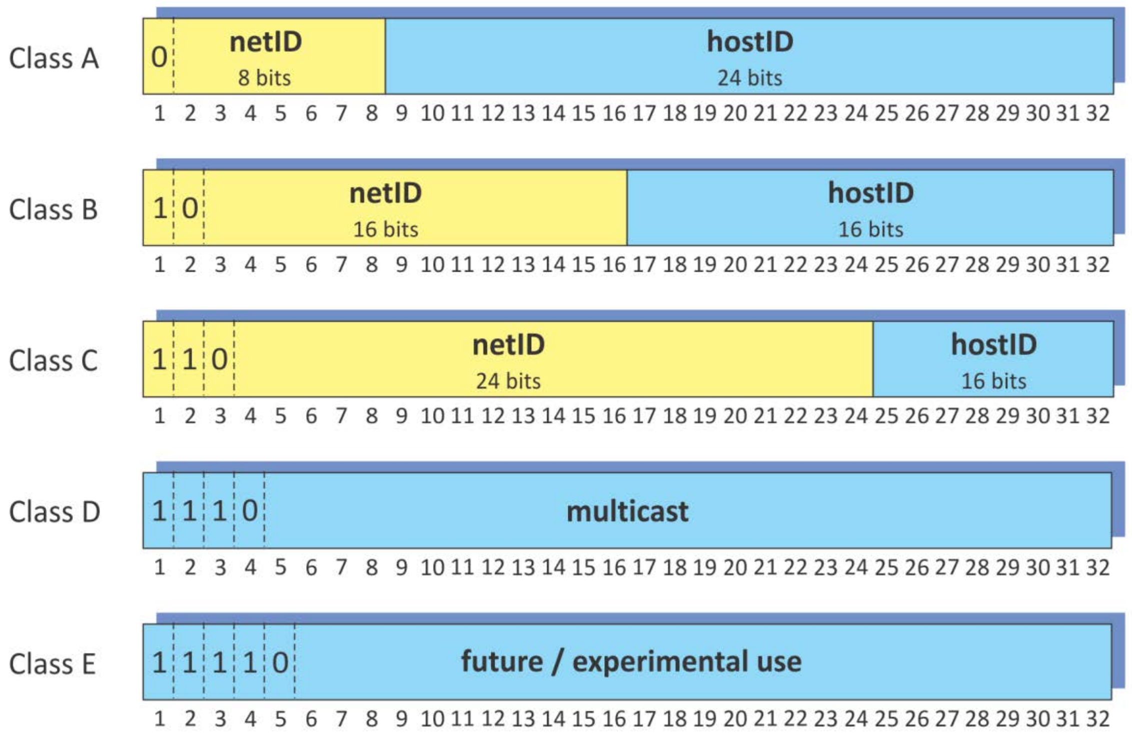


Fig. 2.3 Assigned classes of IP addresses

Here we see that the five classes are named A, B, C, D, and E. The A, B, and C class spaces were used for unicast addresses. If we look more carefully at this addressing structure, we can see how the relative sizes of the different classes and their corresponding address ranges really work. Table 2-3 gives this class structure (sometimes called classful addressing structure).

Class	Address range	High-order bits	Use	Fraction of total	Number of networks	Number of hosts
A	0.0.0.0-127.255.255.255	0	Unicast/special	1/2	2^{7}	2^{24}
B	128.0.0.0-191.255.255.255	10	Unicast/special	1/4	2^{14}	2^{16}
C	192.0.0.0-223.255.255.255	110	Unicast/special	1/8	2^{21}	2^{8}
D	224.0.0.0-239.255.255.255	1110	Multicast	1/16	N/A	N/A
E	240.0.0.0-255.255.255.255	1111	Reserved	1/16	N/A	N/A

Table 2.2. The original IPv4 address space partitioning

The table indicates how the class based addressing structure was used primarily to have a way of allocating unicast address blocks of different sizes to users. The partitioning into classes induces a trade-off between the number of available network numbers of a given size and the number of hosts that can be assigned to the given network. For example, a site allocated the class A network number 18.0.0.0 (MIT) has 224 possible addresses to assign as host addresses (i.e., using IPv4 addresses in the range 18.0.0.0–18.255.255.255), but there are only 128 class A networks available for the entire Internet. A site allocated a class C network number, say, 193.226.37.0, would be able to assign only 256 hosts (i.e., those in the range 193.226.37.0–193.226.37.255), but there are more than two million class C network numbers available.

Note:
These numbers are not exact. Several addresses are not generally available for use as unicast addresses (host addresses). In particular, the first and last addresses of the range are not generally available: the first (lowest) address of the range is used as “network address” while the last (highest) address in the range is used as “broadcast address”. In our example, the site assigned address range 18.0.0.0 would really be able to assign as many as 224 - 2 = 16,777,214 unicast IP addresses.

The class based approach to Internet addressing lasted mostly intact for the first decade of the Internet’s growth (to about the early 1980s). After that, it began to show its first signs of scaling problems—it was becoming too inconvenient to centrally coordinate the allocation of a new class A, B, or C network number every time a new network segment was added to the Internet. In addition, assigning class A and B network numbers tended to waste too many host numbers, whereas class C network numbers could not provide enough host numbers to many new sites.

2.3.1.5 Subnet Addressing

One of the earliest difficulties encountered when the Internet began to grow was the inconvenience of having to allocate a new network number for any new network segment that was to be attached to the Internet. This became especially cumbersome with the development and increasing use of local area networks (LANs) in the early 1980s. To address the problem, it was natural to consider a way that a site attached to the Internet could be allocated a network number centrally that could then be subdivided locally by site administrators. But that would be preferably accomplished without altering the rest of the Internet's core routing infrastructure.

Implementing this idea would require the ability to alter the line between the network portion of an IP address and the host portion, but only for local purposes at a site; the rest of the Internet would "see" only the traditional class A, B, and C partitions. The approach adopted to support this capability is called ***subnet addressing***. Using subnet addressing, a site is allocated a class A, B, or C network number, leaving some number of remaining host bits to be further allocated and assigned within a site. The site may further divide the host portion of its base address allocation into a subnetwork (subnet) number and a host number. Essentially, subnet addressing adds one additional field to the IP address structure, but without adding any bits to its length. As a result, a site administrator is able to trade off the number of subnetworks versus the number of hosts expected to be on each subnetwork without having to coordinate with other sites.

In exchange for the additional flexibility provided by subnet addressing, a new cost is imposed. Because the definition of the Subnet and Host fields is now site-specific (not dictated by the class of the network number), all routers and hosts at a site require a new way to determine where the Subnet field of the address and the Host field of the address are located within the address. Before subnets, this information could be derived directly by knowing whether a network number was from class A, B, or C (as indicated by the first few bits in the address). As an example, using subnet addressing, an IPv4 address might have the form shown in Figure 2.4.

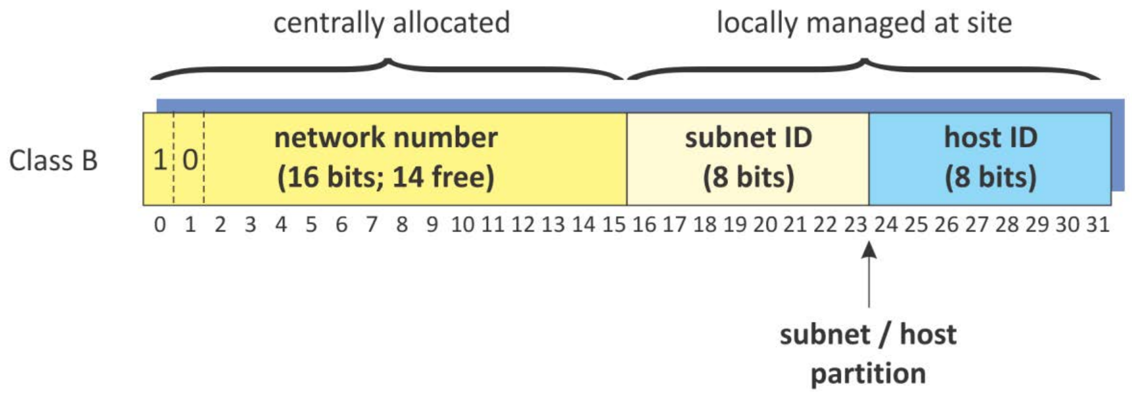


Fig. 2.4. Example of a subnetted class B address

Figure 2.4 is an example of how a class B address might be “subnetted.” Assume that some site in the Internet has been allocated a class B network number. The first 16 bits of every address the site will use are fixed at some particular number because these bits have been allocated by a central authority. The last 16 bits (which would have been used only to create host numbers in the class B network without subnets) can now be divided by the site network administrator as needs may dictate. In this example, 8 bits have been chosen for the subnet number, leaving 8 bits for host numbers. This particular configuration allows the site to support 256 subnetworks, and each subnetwork may contain up to 254 hosts (now the first address of each subnetwork is used as “subnetwork address” and last addresses for each subnetwork is used as “broadcast” address for that specific subnetwork), as opposed to losing only the first and last addresses of the entire allocated range). Recall that the subnetwork structure is known only by hosts and routers where the subnetting is taking place. The remainder of the Internet still treats any address associated with the site just as it did prior to the advent of subnet addressing. Figure 2.5 shows how this works.

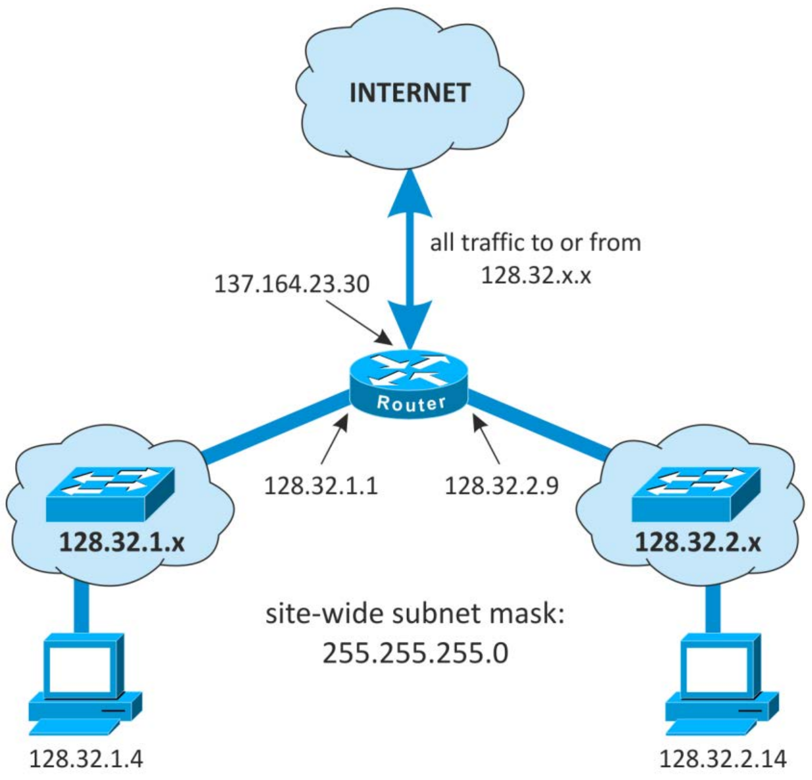


Fig. 2.5 Subnetworks

This figure shows a hypothetical site attached to the Internet with one border router (i.e., one attachment point to the Internet) and two internal local area networks. The value of x could be anything in the range [0, 255]. Each of the Ethernet networks is an IPv4 subnetwork of the overall network number 128.32, a class B address allocation. For other sites on the Internet to reach this site, all traffic with destination addresses starting with 128.32 is directed by the Internet routing system to the border router (specifically, its interface with IPv4 address 137.164.23.30). At this point, the border router must distinguish among different subnetworks within the 128.32 network. In particular, it must be able to distinguish and separate traffic destined for addresses of the form 128.32.1.x from those destined for addresses of the form 128.32.2.x. These represent subnetwork numbers 1 and 2, respectively, of the 128.32 class B network number. In order to do this, the router must be aware of where the subnet ID is to be found within the addresses.

2.3.1.6 Subnet Masks

The subnet mask is an assignment of bits used by a host or router to determine how the network and subnetwork information is partitioned from the host information in a corresponding IP address. Subnet masks for IP are the same length as the corresponding IP addresses (32 bits for IPv4 and 128 bits for IPv6). They are typically configured into a host or router in the same way as IP addresses—either statically (typical for routers) or using some dynamic system such as the Dynamic Host Configuration Protocol (DHCP). For IPv4, subnet masks may be written in the same way an IPv4 address is written (i.e., dotted-decimal). Although not originally required to be arranged in this manner, today subnet masks are structured as some number of 1 bits followed by some number of 0 bits. Because of this arrangement, it is possible to use a shorthand format for expressing masks that simply gives the number of contiguous 1 bits in the mask (starting from the left). This format is now the most common format and is sometimes called the prefix length. Table 2.3 presents some examples for IPv4.

Dotted-Decimal Representation	Shorthand (Prefix Length)	Binary Representation
128.0.0.0	/1	10000000 00000000 00000000 00000000
255.0.0.0	/8	11111111 00000000 00000000 00000000
255.192.0.0	/10	11111111 11000000 00000000 00000000
255.255.254.0	/23	11111111 11111111 11111110 00000000
255.255.255.0	/24	11111111 11111111 11111111 00000000
255.255.255.192	/27	11111111 11111111 11111111 11100000
255.255.255.255	/32	11111111 11111111 11111111 11111111

Table 2.3. IPv4 subnet mask examples in various formats

Masks are used by routers and hosts to determine where the network/subnetwork portion of an IP address ends and the host part begins. A bit set to 1 in the subnet mask means the corresponding bit position in an IP address should be considered part of a combined network/subnetwork portion of an address, which is used as the basis for forwarding datagrams. Conversely, a bit set to 0 in the subnet mask means the corresponding bit position in an IP address should be considered part of the host portion. For example, in Figure 2.6 we can see how the IPv4 address 128.32.1.14 is treated when a subnet mask of 255.255.255.0 is applied to it.

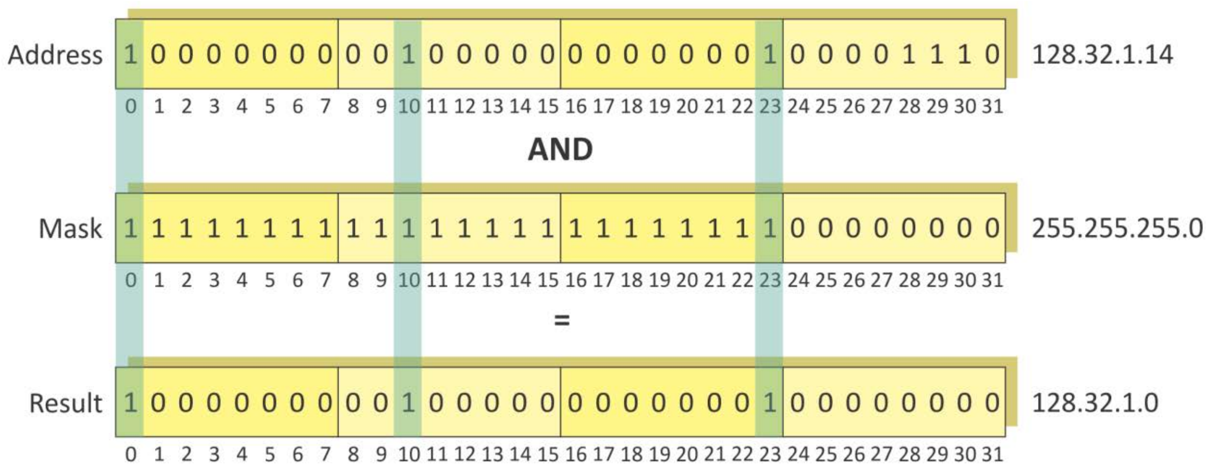


Fig. 2.6. Obtaining the subnetwork address

Recalling the bitwise AND operation, a result bit is only ever a 1 if the corresponding bits in both the mask and the address are 1. In this example, we see that the address 128.32.1.14 belongs to the subnet 128.32.1.0/24. In Figure 2-3, this is precisely the information required by the border router to determine to which subnetwork a datagram destined for the system with address 128.32.1.14 should be forwarded. Note again that the rest of the Internet routing system does not require knowledge of the subnet mask because routers outside the site make routing decisions based only on the network number portion of an address and not the combined network/subnetwork or host portions. Consequently, subnet masks are purely a local matter at the site.

Variable-Length Subnet Masks (VLSM)

So far we have discussed how a network number allocated to a site can be subdivided into ranges assigned to multiple subnetworks, each of the same size and therefore able to support the same number of hosts, based on the operational expectations of the network administrator. We now observe that it is possible to use a different-length subnet mask applied to the same network number in different portions of the same site. Although doing this complicates address configuration management, it adds flexibility to the subnet structure because different subnetworks may be set up with different numbers of hosts. Variablelength subnet masks (VLSM) are now supported by most hosts, routers, and routing protocols.

Recall that the number of hosts is constrained by the number of bits remaining in the IP address that are not used by the network/subnet number. For IPv4 and a /24 prefix, this allows for 32 – 24 = 8 bits (256 hosts); for /25, half as many (128 hosts); and for /26, half further still (64 hosts).

Although it may not seem obvious, there is a common case where a subnetwork contains only two hosts. When routers are connected together by a point-to-point link requiring an IP address to be assigned at each end, it is common practice to use a /30 network prefix with

IPv4 (that means a 4 IPs subnetwork: 2 IPs for the routers, one IP for the subnetwork address and one IP for the broadcast address).

2.3.1.7 Broadcast Addresses

In each IPv4 subnetwork, a special address is reserved to be the subnet broadcast address. The subnet broadcast address is formed by setting the network/subnetwork portion of an IPv4 address to the appropriate value and all the bits in the Host field to 1. The subnet broadcast address is constructed by inverting the subnet mask (i.e., changing all the 0 bits to 1 and vice versa) and performing a bitwise OR operation with the address of any of the computers on the subnet (or, equivalently, the network/subnetwork prefix). Recall that the result of a bitwise OR operation is 1 if either input bit is 1. Using the IPv4 address 128.32.1.14, this computation can be written as shown in Figure 2.7.

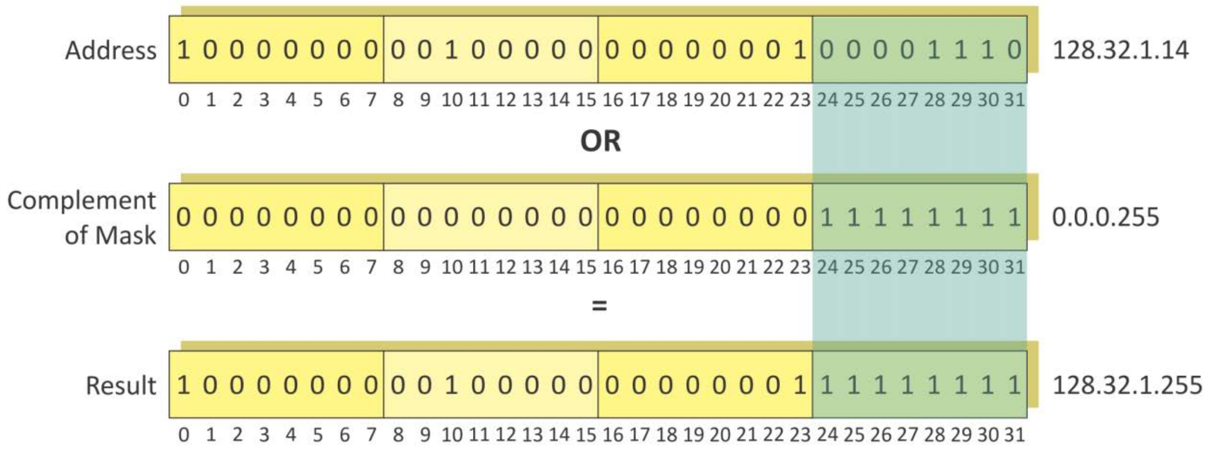


Fig. 2.7. Finding the broadcast address

As shown in the figure, the subnet broadcast address for the subnet 128.32.1.0/24 is 128.32.1.255. Historically, a datagram using this type of address as its destination has also been known as a directed broadcast. Such a broadcast can, at least theoretically, be routed through the Internet as a single datagram until reaching the target subnetwork, at which point it becomes a collection of broadcast datagrams that are delivered to all hosts on the subnetwork. Generalizing this idea further, we could form a datagram with the destination IPv4 address 128.32.255.255 and launch it into the Internet. This would address all hosts at the target site.

Note
Directed broadcasts were found to be such a big problem from a security point of view that they are effectively disabled on the Internet today. By default routers must now disable the forwarding of directed broadcasts and are even free to omit support for the capability altogether.

In addition to the subnet broadcast address, the special-use address 255.255.255.255 is reserved as the local net broadcast (also called limited broadcast), which is never forwarded by routers. Note that although routers may not forward broadcasts, subnet broadcasts and local net broadcasts destined for the same network to which a computer is attached should be expected to work unless explicitly disabled by end hosts. Such broadcasts do not require action by a router; data link-layer broadcast mechanisms, if available, are used for supporting them. Broadcast addresses are typically used with protocols such as UDP/IP or ICMP because these protocols do not involve two-party conversations as in TCP/IP. IPv6 lacks any broadcast addresses; for places where broadcast addresses might be used in IPv4, IPv6 instead uses exclusively multicast addresses.

2.3.1.8 Multicast Addresses

Multicast addressing is supported by IPv4 and IPv6. An IP multicast address (also called group or group address) identifies a group of host interfaces, rather than a single one. Generally speaking, the group could span the entire Internet. The portion of the network that a single group covers is known as the group's scope. Common scopes include node-local (same computer), link-local (same subnet), site-local (applicable to some site), global (entire Internet), and administrative. Administrative scoped addresses may be used in an area of the network that has been manually configured into routers. A site administrator may configure routers as admin-scope boundaries, meaning that multicast traffic of the associated group is not forwarded past the router. Note that the site-local and administrative scopes are available for use only with multicast addressing.

Under software control, the protocol stack in each Internet host is able to join or leave a multicast group. When a host sends something to a group, it creates a datagram using one of its own (unicast) IP addresses as the source address and a multicast IP address as the destination. All hosts in scope that have joined the group should receive any datagrams sent to the group. The sender is not generally aware of the hosts receiving the datagram unless they explicitly reply. Indeed, the sender does not even know in general how many hosts are receiving its datagrams.

The original multicast service model, described so far, has become known as any-source multicast (ASM). In this model, any sender may send to any group; a receiver joins the group by specifying only the group address. A newer approach, called source-specific multicast (SSM), uses only a single sender per group. In this case, when joining a group, a host specifies the address of a channel, which comprises both a group address and a source IP address. SSM was developed to avoid some of the complexities in deploying the ASM model. Although neither form of multicast is widely available throughout the Internet, it seems that SSM is now the more likely candidate for adoption.

Understanding and implementing wide area multicasting has been an ongoing effort within the Internet community for more than a decade, and a large number of protocols have been developed to support it.

For IPv4, the class D space (224.0.0.0–239.255.255.255) has been reserved for supporting multicast. With 28 bits free, this provides for the possibility of 228 = 268,435,456 host groups (each host group is an IP address). This address space is divided into major sections based on the way they are allocated and handled with respect to routing.

The blocks of addresses up to 224.255.255.255 are allocated for the exclusive use of certain application protocols or organizations. The local network control block is limited to the local network of the sender; datagrams sent to those addresses are never forwarded by multicast routers. The All Hosts group (224.0.0.1) is one group in this block. The internetwork control block is similar to the local network control range but is intended for control traffic that needs to be routed off the local link. An example from this block is the Network Time Protocol (NTP) multicast group (224.0.1.1)

The first ad hoc block was constructed to hold addresses that did not fall into either the local or internetwork control blocks. Most of the allocations in this range are for commercial services, some of which do not (or never will) require global address allocations;

The other major address blocks were created somewhat later in the evolution of IP multicast. The SSM block is used by applications employing SSM in combination with their own unicast source IP address in forming SSM channels, as described previously.

The most recent of the IPv4 multicast address allocation mechanisms associates a number of multicast addresses with an IPv4 unicast address prefix. This is called unicast-prefix-based multicast addressing (UBM). The UBM IPv4 address range is 234.0.0.0 through 234.255.255.255. A unicast address allocation with a /24 or shorter prefix may make use of UBM addresses. Allocations with fewer addresses (i.e., a /25 or longer prefix) must use some other mechanism. UBM addresses are constructed as a concatenation of the 234/8 prefix, the allocated unicast prefix, and the multicast group ID. Figure 2.8 shows the format.

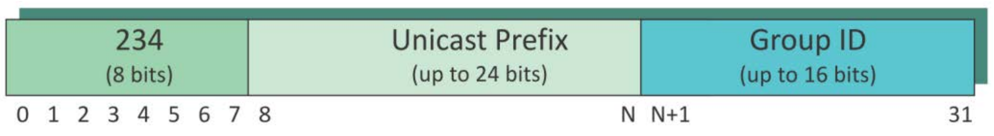


Fig. 2.8. The IPv4 UBM address format

To determine the set of UBM addresses associated with a unicast allocation, the allocated prefix is simply prepended with the 234/8 prefix. For example, the unicast IPv4 address prefix 192.0.2.0/24 has a single associated UBM address, 234.192.0.2. It is also possible to determine the owner of a multicast address by simply “left-shifting” the multicast address by 8 bit positions. We know that the multicast address range 234.128.32.0/24 is allocated to UC Berkeley, for example, because the corresponding unicast IPv4 address space 128.32.0.0/16 (the “left-shifted” version of 234.128.32.0) is owned by UC Berkeley.

UBM addresses may offer advantages over the other types of multicast address allocations. For example, they do not carry the 16-bit restriction for AS numbers used by GLOP addressing. In addition, they are allocated as a consequence of already-existing unicast address space allocations. Thus, sites wishing to use multicast addresses already know which addresses they can use without further coordination.

The administratively scoped address block can be used to limit the distribution of multicast traffic to a particular collection of routers and hosts. These are the multicast analogs of private unicast IP addresses. Such addresses should not be used for distributing multicast into the Internet, as most of them are blocked at enterprise boundaries. Large sites sometimes subdivide administratively scoped multicast addresses to cover specific useful scopes (e.g., work group, division, and geographical area).

2.3.1.9 Anycast Addresses

An anycast address is a unicast IPv4 or IPv6 address that identifies a different host depending on where in the network it is used. This is accomplished by configuring Internet routers to advertise the same unicast routes from multiple locations in the Internet. Thus, an anycast address refers not to a single host in the Internet, but to the "most appropriate" or "closest" single host that is responding to the anycast address. Anycast addressing is used most frequently for finding a computer that provides a common service. For example, a datagram sent to an anycast address could be used to find a DNS server, a 6to4 gateway that encapsulates IPv6 traffic in IPv4 tunnels etc.

2.3.1.10 Classless Inter-Domain Routing (CIDR) and Aggregation

In the early 1990s, after the adoption of subnet addressing to ease one form of growing pains, the Internet started facing a serious set of scaling problems. Three particular issues were considered so important as to require immediate attention:

1. By 1994, over half of all class B addresses had already been allocated. It was expected that the class B address space would be exhausted by about 1995.
2. The 32-bit IPv4 address was thought to be inadequate to handle the size of the Internet anticipated by the early 2000s.
3. The number of entries in the global routing table (one per network number), about 65,000 in 1995, was growing. As more and more class A, B, and C routing entries appeared, routing performance would suffer.

These three issues were attacked by a group in the Internet Engineering Task Force (IETF) called ROAD (for ROuting and ADdressing), starting in 1992. They considered problems 1 and 3 to be of immediate concern, and problem 2 as requiring a long-term solution. The short-term solution they proposed was to effectively remove the class breakdown of IP addresses and also promote the ability to aggregate hierarchically assigned IP addresses. These measures would help problems 1 and 3. IPv6 was envisioned to deal with problem 2.

2.3.1.11 Prefixes

In order to help relieve the pressure on the availability of IPv4 addresses (especially class B addresses), the class based addressing scheme was generalized using a scheme similar to VLSM, and the Internet routing system was extended to support Classless Inter-Domain Routing (CIDR). This provided a way to conveniently allocate contiguous address ranges that contained more than 255 hosts but fewer than 65,536. That is, something other than single class B or multiple class C network numbers could be allocated to sites. Using CIDR, any address range is not predefined as being part of a class but instead requires a mask similar to a subnet mask, sometimes called a CIDR mask. CIDR masks are not limited to a site but are instead visible to the global routing system. Thus, the core Internet routers must be able to interpret and process masks in addition to network numbers. This combination of numbers, called a network prefix, is used for both IPv4 and IPv6 address management.

Eliminating the predefined separation of network and host number within an IP address makes finer-grain allocation of IP address ranges possible. As with class based addressing, dividing the address spaces into chunks is most easily achieved by grouping numerically contiguous addresses for use as a type or for some particular special purpose. Such groupings are now commonly expressed using a prefix of the address space. An n-bit prefix is a predefined value for the first n bits of an address. The value of n (the length of the prefix) is typically expressed as an integer in the range 0–32 for IPv4 and 0–128 for IPv6. It is generally appended to the base IP address following a / character. Table 2.4. gives some examples of prefixes and their corresponding IPv4 or IPv6 address ranges.

Prefix	Prefix (Binary)	Address Range
0.0.0.0/0	00000000 00000000 00000000 00000000	0.0.0.0 - 255.255.255.255
128.0.0.0/1	**1**0000000 00000000 00000000 00000000	128.0.0.0 - 255.255.255.255
128.0.0.0/24	**10000000 00000000 00000000** 00000000	128.0.0.0 - 128.0.0.255
193.64.32.192/27	**11000001 01000000 00100000 110**00000	193.64.32.192 - 193.64.32.223
193.64.32.17/32	**11000001 01000000 00100000 00010001**	193.64.32.17

Table 2.4. Examples of prefixes and their corresponding IPv4 or IPv6 address range

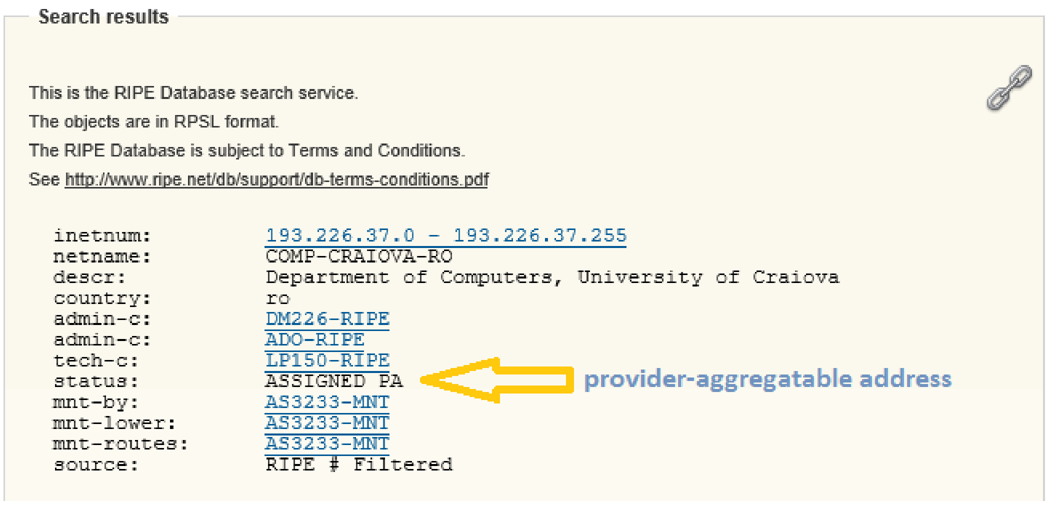
In the table, the bits defined and fixed by the prefix are highlighted in blue. The remaining bits may be set to any combination of 0s and 1s, thereby covering the possible address range. Clearly, a smaller prefix length corresponds to a larger number of possible addresses. In addition, the earlier class based addressing approach is easily generalized by this scheme. For example, the class C network number 193.64.32.0 can be written as the prefix 193.64.32.0/24 or 193.64.32/24. Class based A and B network numbers can be expressed using /8 and /16 prefix lengths, respectively.

2.3.1.12 Allocation

IP address space is allocated, usually in large chunks, by a collection of hierarchically organized authorities. The authorities are generally organizations that allocate address space to various owners— usually ISPs or other smaller authorities. Authorities are most often involved in allocating portions of the global unicast address space, but other types of addresses (multicast and special-use) are also sometimes allocated. The authorities can make allocations to users for an undetermined amount of time, or for a limited time (e.g., for running experiments). The top of the hierarchy is the Internet Assigned Numbers Authority (IANA - https://www.iana.org), which has wide-ranging responsibility for allocating IP addresses and other types of numbers used in the Internet protocols.

For unicast IPv4 and IPv6 address space, the IANA delegates much of its allocation authority to a few Regional Internet Registries (RIRs). The RIRs coordinate with each other through an organization formed in 2003 called the Number Resource Organization (NRO). Note in addition that, as of early 2011, all the remaining unicast IPv4 address space held by IANA for allocation had been handed over to these RIRs. The RIR for Europe, Middle East, and Central Asia is Réseaux IP Européens (RIPE - http://www.ripe.net/)

These entities typically deal with relatively large address blocks. They allocate address space to smaller registries operating in countries (e.g., for Romania: Institutul de Cercetari in Informatica ICI - http://www.rnc.ro) and to large Internet Service Providers (ISPs). ISPs, in turn, provide address space to their customers and themselves. When users sign up for Internet service, they are ordinarily provided a (typically small) fraction or range of their ISP's address space in the form of an address prefix. These address ranges are owned and managed by the customer's ISP and are called provider-aggregatable (PA) addresses because they consist of one or more prefixes that can be aggregated with other prefixes the ISP owns (see Fig. 2.9.). Such addresses are also sometimes called non-portable addresses. Switching providers typically requires customers to change the IP prefixes on all computers and routers they have that are attached to the Internet (an often unpleasant operation called renumbering).


Search results

This is the RIPE Database search service.
The objects are in RPSL format.
The RIPE Database is subject to Terms and Conditions.
See http://www.ripe.net/db/support/db-terms-conditions.pdf

```
inetnum:        193.226.37.0 - 193.226.37.255
netname:        COMP-CRAIOVA-RO
descr:          Department of Computers, University of Craiova
country:        ro
admin-c:        DM226-RIPE
admin-c:        ADO-RIPE
tech-c:         LP150-RIPE
status:         ASSIGNED PA
mnt-by:         AS3233-MNT
mnt-lower:      AS3233-MNT
mnt-routes:     AS3233-MNT
source:         RIPE # Filtered
```

Fig. 2.9. RIPE record

An alternative type of address space is called provider-independent (PI) address space. Addresses allocated from PI space are allocated to the user directly and may be used with any ISP. However, because such addresses are owned by the customer, they are not numerically adjacent to the ISP's own addresses and are therefore not aggregatable. An ISP being asked to provide routing for a customer's PI addresses may require additional payment for service or simply not agree to support such a configuration. In some sense, an ISP that agrees to provide routing for a customer's PI addresses is taking on an extra cost relative to other customers by having to increase the size of its routing tables. On the other hand, many sites prefer to use PI addresses, and might be willing to pay extra for them, because it helps to avoid the need to renumber when switching ISPs (avoiding what has become known as provider lock).

Many Internet users who own more than one computer find that having only a single computer attached to the Internet is not an ideal situation. As a result, they have home LAN or WLAN networks and use either a router or a computer acting as a router to provide connectivity to the Internet. Such configurations are very similar to the single-computer case, except that the router forwards packets from the home network to the ISP and also performs Network Address Translation - NAT (also called Internet Connection Sharing (ICS) in Windows) by rewriting the IP addresses in packets being exchanged with the customer's ISP. From the ISP's point of view, only a single IP address has been used. Today, much of this activity is automated, so the need for manual address configuration is minimal. The routers provide automatic address assignment to the home clients using DHCP. They also handle address assignment for the link set up with the ISP if necessary.

2.3.1.13 Attacks Involving IP Addresses

Given that IP addresses are essentially numbers, few network attacks involve only them. Generally, attacks can be carried out when sending "spoofed" datagrams or with other related activities. That said, IP addresses are now being used to help identify individuals suspected of undesirable activities (e.g., copyright infringement in peer-to-peer networks or distribution of illegal materials). Doing this can be misleading for several reasons. For example, in many circumstances IP addresses are only temporary and are reassigned to different users at different times. Therefore, any errors in accurate timekeeping can easily cause databases that map IP addresses to users to be incorrect.

Furthermore, access controls are not widely and securely deployed; it is often possible to attach to the Internet through some public access point or some unintentionally open wireless router in someone's home or office. In such circumstances, the unsuspecting home or business owner may be targeted based on IP address even though that person was not the originator of traffic on the network. This can also happen when compromised hosts are used to form botnets. Such collections of computers (and routers) can now be leased on what has effectively become an Internet-based black market for carrying out attacks, serving illicit content and other misdeeds.

2.3.1.14 IPv4 datagram format

IP is the basis of the TCP/IP protocol suite. All TCP, UDP, ICMP, and IGMP data gets transmitted as IP datagrams. IP provides a best-effort, connectionless datagram delivery service. By "best-effort" we mean there are no guarantees that an IP datagram gets to its destination successfully. Although IP does not simply drop all traffic unnecessarily, it provides no guarantees as to the fate of the packets it attempts to deliver. When something goes wrong, such as a router temporarily running out of buffers, IP has a simple error-handling algorithm: throw away some data (usually the last datagram that arrived). Any required reliability must be provided by the upper layers (e.g., TCP). IPv4 and IPv6 both use this basic best-effort delivery model.

The term connectionless means that IP does not maintain any connection state information about related datagrams within the network elements (i.e., within the routers); each datagram is handled independently from all other others. This also means that IP datagrams can be delivered out of order. If a source sends two consecutive datagrams (first A, then B) to the same destination, each is routed independently and can take different paths, and B may arrive before A. Other things can happen to IP datagrams as well: they may be duplicated in transit, and they may have their data altered as the result of errors. Again, some protocol above IP (usually TCP) has to handle all of these potential problems in order to provide an error-free delivery abstraction for applications.

Fig. 2.10 shows the format of an IPv4 datagram. The normal size of the IPv4 header is 20 bytes, unless options are present (which is rare). In the following pictures of headers and datagrams, the most significant bit is numbered 0 at the left, and the least significant bit of a 32-bit value is numbered 31 on the right. The 4 bytes in a 32-bit value are transmitted in the following order: bits 0–7 first, then bits 8–15, then 16–23, and bits 24–31 last. This is called big endian byte ordering, which is the byte ordering required for all binary integers in the TCP/IP headers as they traverse a network. It is also called network byte order.

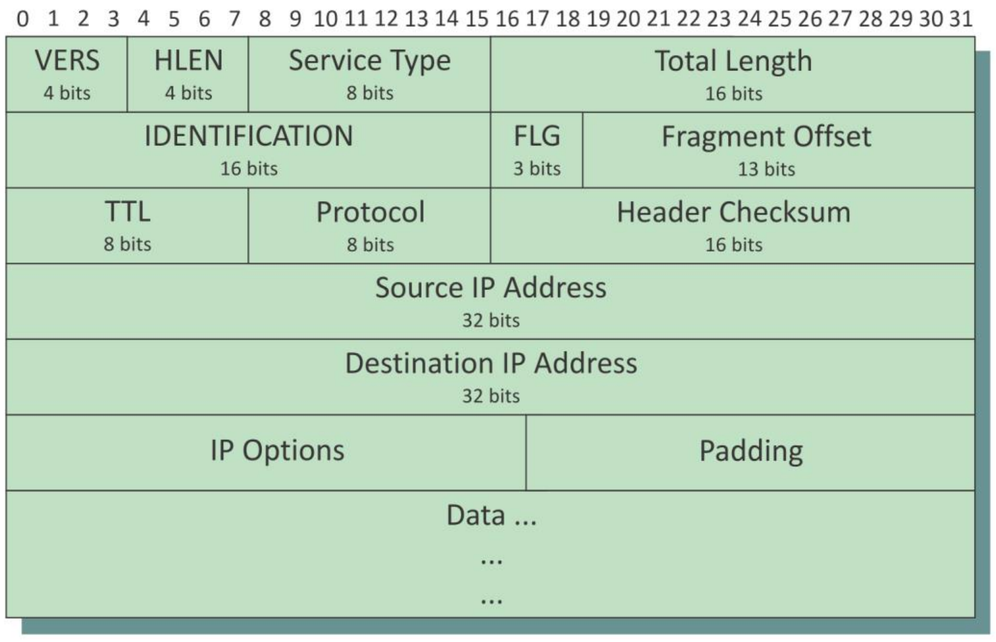


Fig. 2.10. Format of an IP datagram header

Computer CPUs that store binary integers in other formats, such as the little endian format used by most PCs, must convert the header values into network byte order for transmission and back again for reception.

The first field (only 4 bits or one nibble wide) is the Version field. It contains the version number of the IP datagram: 4 for IPv4 and 6 for IPv6. The headers for both IPv4 and IPv6 share the location of the Version field but no others. Thus, the two protocols are not directly interoperable—a host or router must handle either IPv4 or IPv6 (or both, called dual stack) separately. Although other versions of IP have been proposed and developed, only versions 4 and 6 have any significant amount of use. The IANA keeps an official registry of these version numbers.

The Internet Header Length (IHL) field is the number of 32-bit words in the IPv4 header, including any options. Because this is also a 4-bit field, the IPv4 header is limited to a maximum of fifteen 32-bit words or 60 bytes. Later we shall see how this limitation makes some of the options, such as the Record Route option, nearly useless today. The normal value of this field (when no options are present) is 5. There is no such field in IPv6 because the header length is fixed at 40 bytes.

Following the header length, the original specification of IPv4 specified a Type of Service (ToS) byte, and IPv6 specified the equivalent Traffic Class byte. Use of these never became widespread, so eventually this 8-bit field was split into two smaller parts. The first 6 bits are now called the Differentiated Services Field (DS Field), and the last 2 bits are the Explicit Congestion Notification (ECN) field or indicator bits. These fields are used for special processing of the datagram when it is forwarded.

The Total Length field is the total length of the IPv4 datagram in bytes. Using this field and the IHL field, we know where the data portion of the datagram starts, and its length. Because this is a 16-bit field, the maximum size of an IPv4 datagram (including header) is 65,535 bytes. The Total Length field is required in the header because some lower-layer protocols that carry IPv4 datagrams do not (accurately) convey the size of encapsulated datagrams on their own. Ethernet, for example, pads small frames to be a minimum length (64 bytes). Even though the minimum Ethernet payload size is 46 bytes, an IPv4 datagram can be smaller (as few as 20 bytes). If the Total Length field were not provided, the IPv4 implementation would not know how much of a 46-byte Ethernet frame was really an IP datagram, as opposed to padding, leading to possible confusion.

Although it is possible to send a 65,535-byte IP datagram, most link layers (such as Ethernet) are not able to carry one this large without fragmenting it (chopping it up) into smaller pieces. Furthermore, a host is not required to be able to receive an IPv4 datagram larger than 576 bytes. (In IPv6 a host must be able to process a datagram at least as large as the MTU of the link to which it is attached, and the minimum link MTU is 1280 bytes.) Many applications that use the UDP protocol for data transport (e.g., DNS, DHCP, etc.) use a limited data size of 512 bytes to avoid the 576-byte IPv4 limit. TCP chooses its own datagram size based on additional information.

When an IPv4 datagram is fragmented into multiple smaller fragments, each of which itself is an independent IP datagram, the Total Length field reflects the length of the particular fragment. As with IPv4, the 16-bit size of the field limits its maximum value to 65,535.

The Identification field helps identify each datagram sent by an IPv4 host. To ensure that the fragments of one datagram are not confused with those of another, the sending host normally increments an internal counter by 1 each time a datagram is sent (from one of its IP addresses) and copies the value of the counter into the IPv4 Identification field. This field is most important for implementing fragmentation.

The Time-to-Live field, or TTL, sets an upper limit on the number of routers through which a datagram can pass. It is initialized by the sender to some value (64 is recommended, although 128 or 255 is not uncommon) and decremented by 1 by every router that forwards the datagram. When this field reaches 0, the datagram is thrown away, and the sender is notified with an ICMP message. This prevents packets from getting caught in the network forever should an unwanted routing loop occur.

Note
The TTL field was originally specified to be the maximum lifetime of an IP datagram in seconds, but routers were also always required to decrement the value by at least 1. Because virtually no routers today hold on to a datagram longer than 1s under normal operation, the earlier rule is now ignored or forgotten, and in IPv6 the field has been renamed to its de facto use: Hop Limit.

The Protocol field in the IPv4 header contains a number indicating the type of data found in the payload portion of the datagram. The most common values are 17 (for UDP) and 6 (for TCP). This provides a demultiplexing feature so that the IP protocol can be used to carry payloads of more than one protocol type. Although this field originally specified the transport-layer protocol the datagram is encapsulating, it is now understood to identify the encapsulated protocol, which may or not be a transport protocol. For example, other encapsulations are possible, such as IPv4-in-IPv4 (value 4). The official list of the possible values of the Protocol field is given in the assigned numbers page. The Next Header field in the IPv6 header generalizes the Protocol field from IPv4. It is used to indicate the type of header following the IPv6 header. This field may contain any values defined for the IPv4 Protocol field, or any of the values associated with the IPv6 extension headers.

The Header Checksum field is calculated over the IPv4 header only. This is important to understand because it means that the payload of the IPv4 datagram (e.g., TCP or UDP data) is not checked for correctness by the IP protocol. To help ensure that the payload portion of an IP datagram has been correctly delivered, other protocols must cover any important data that follows the header with their own data-integrity-checking mechanisms. We shall see that almost all protocols encapsulated in IP (ICMP, IGMP, UDP, and TCP) have a checksum in their own headers to cover their header and data and also to cover certain parts of the IP header they deem important (a form of “layering violation”). Perhaps surprisingly, the IPv6 header does not have any checksum field.

Note
Omitting the checksum field from the IPv6 header was a somewhat controversial decision. The reasoning behind this action is roughly as follows: Higher-layer protocols requiring correctness in the IP header are required to compute their own checksums over the data they believe to be important. A consequence of errors in the IP header is that the data is delivered to the wrong destination, is indicated to have come from the wrong source, or is otherwise mangled during delivery. Because bit errors are relatively rare (thanks to fiber-optic delivery of Internet traffic) and stronger mechanisms are available to ensure correctness of the other fields (higher-layer checksums or other checks), it was decided to eliminate the field from the IPv6 header.

The algorithm used in computing a checksum is also used by most of the other Internet-related protocols that use checksums and is sometimes known as the Internet checksum. Note that when an IPv4 datagram passes through a router, its header checksum must change as a result of decrementing the TTL field.

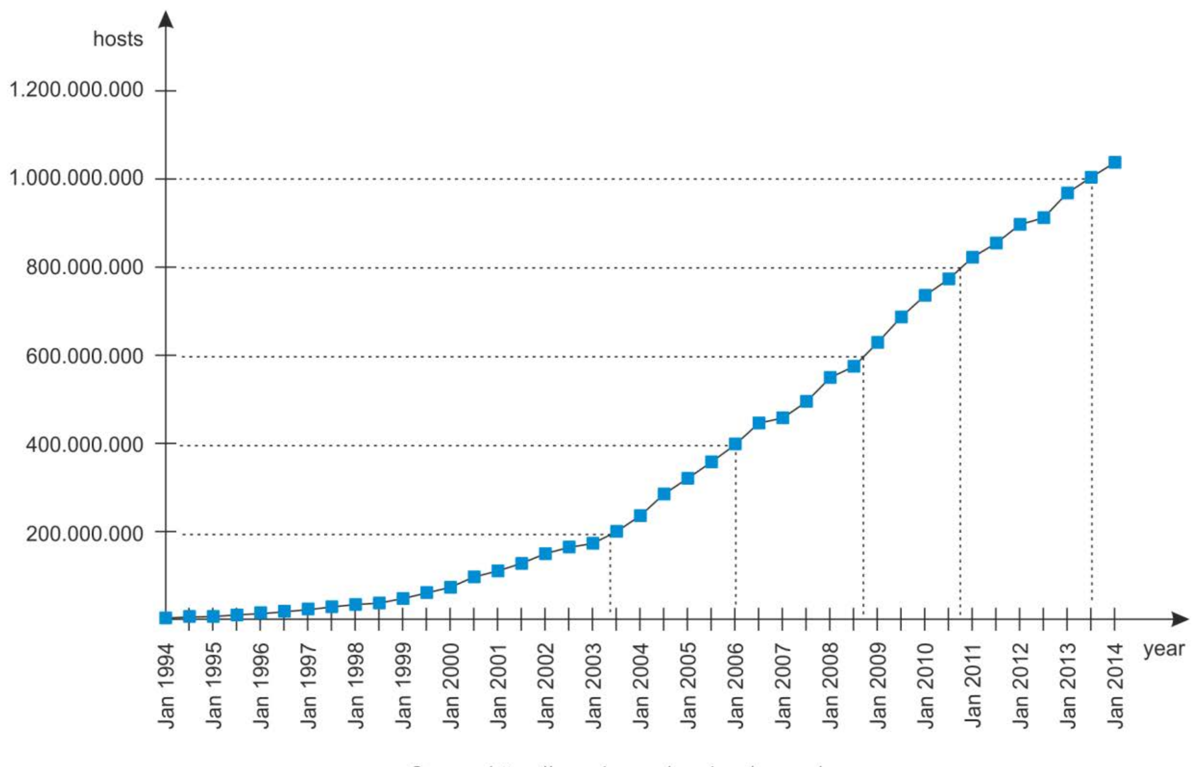
Every IP datagram contains the Source IP Address of the sender of the datagram and the Destination IP Address of where the datagram is destined. These are 32-bit values for IPv4 and 128-bit values for IPv6, and they usually identify a single interface on a computer, although multicast and broadcast addresses violate this rule. While a 32-bit address can accommodate a seemingly large number of Internet entities (4.5 billion), there is widespread agreement that this number is inadequate, a primary motivation for moving to IPv6. The 128-bit address of IPv6 can accommodate a huge number of Internet entities. IPv6 has $2^{128} = 3{,}4 \times 10^{38}$ (340 undecillion) addresses. As the surface of the Earth is about 510.072.000 km^2, this implies that there are about $6{,}67 \times 10^{23}$ IPv6 addresses per square meter on Earth. Compared to IPv4, which offers only 8 addresses per square kilometer, this is a significant improvement on paper. It seems as if this should last a very, very long time indeed.

2.3.2 IP version 6

Due to the increasing number of hosts connected to Internet (see Fig. 2.11.), we are currently crossing a period of switching to IPv6 addresses.

The IPv6 story began in the early nineties when it was discovered that the address space available in IPv4 was vanishing quite rapidly. Contemporary studies indicated that it may be depleted within the next ten years – around 2005! These findings challenged the Internet community to start looking for a solution. Two possible approaches were at hand:

1. Minimal: Keep the protocol intact, just increase the address length. This was the easier way promising less pain in the deployment phase.
2. Maximal: Develop an entirely new version of the protocol. Taking this approach would enable incorporating new features and enhancements in IP.



Source: https://www.isc.org/services/survey/

Fig. 2.11. Increasing number of hosts connected to Internet

Because there was no urgent need for a quick solution, the development of a new protocol was chosen. Its original name IP Next Generation (IPng) was soon replaced by IP version 6 which is now the definitive name. The main architects of this new protocol were Steven Deering and Robert Hinden.

The first set of RFCs specifying the IPv6 were released at the end of 1995, namely, RFC 1883: Internet Protocol, Version 6 (IPv6) Specification [RFC1883] and its relatives. Once the definition was available, implementations were eagerly awaited. But they did not come. The second half of the nineties was a period of significant Internet boom. Companies on the market had to solve a tricky business problem: while an investment in IPv6 can bring some benefits in the future, an investment in the blossoming IPv4 Internet earns money now. For a vast majority of them it was essentially a no-brainer: they decided to prefer the rapid and easy return of investments and developed IPv4-based products.

Another factor complicating IPv6 deployment was the change of rules in the IPv4 domain. Methods to conserve the address space were developed and put into operation. The most important of these was Classless Inter-Domain Routing (CIDR). The old address classes were removed and address assignment rules hardened. As a consequence, newly connected sites obtained significantly less addresses than in previous years.

The use of CIDR may well have delayed the need for IPv6 in the eyes of many people, but not in all. Somewhat perversely, the use of CIDR accelerated the perception of a lack of address space in the eyes of those who were relatively new to the Internet. Since the Internet was booming, requests for new address blocks were increasing, yet the size of assigned address blocks were being reduced. Thus, new or expanding sites had to develop methods to spare this scarce resource. One of these approaches is network address translation (NAT) which allows a network to use an arbitrary number of non-public addresses, which are translated to public ones when the packets leave the site (and vice versa). Thus, NATs provided a mechanism for hosts to share public addresses. Furthermore, mechanisms such as PPP and DHCP provide a means for hosts to lease addresses for some period of time.

Probably the most important period for IPv6 so far has been the first years of the 21st century, when IPv6 finally gained some momentum. The increasing number of implementations forced remaining hardware/software vendors to react and to enhance their products with IPv6 capabilities.

The deployment of IPv6 in Europe has been boosted by the largest European networking project - the academic backbone GÉANT – which would include IPv6 support once sufficient confidence and experience had been gained. University of Craiova is one of the main nodes in RoEduNet (Romanian Education Network) which is part of GÉANT.

The main focus of the GÉANT project was to interconnect national research and education networks in European countries. As these networks were interested in IPv6, its support in the backbone was one of the natural consequences. After a period of experiments, IPv6 has been officially provided by GÉANT since January 2004.

Between 2000 and 2004, the vast majority of operating system and router vendors implemented IPv6. These days it is hard to find a platform without at least some IPv6 support. Although the implementations are not perfect yet (advanced features like security or mobility are missing in many of them), they provide a solid ground for basic usage. Moreover, in most cases one can see dramatic improvements from one release to the next.

All in all, after some years of hesitation IPv6 finally leaves the status of a high-tech extravagance and starts to be a usable tool.

IPv6 addressing architecture

IPv6 supports unicast, multicast and anycast addresses. As with IPv4, an IPv6 unicast address is used to identify one datalink-layer interface on a host. If a host has several datalink layer interfaces (e.g. an Ethernet interface and a WiFi interface), then it needs several IPv6 addresses. In general, an IPv6 unicast address is structured as shown in the figure below.
An IPv6 unicast address is composed of three parts :

1. A global routing prefix that is assigned to the Internet Service Provider that owns this block of addresses
2. A subnet identifier that identifies a customer of the ISP
3. An interface identifier that identifies a particular interface on an endsystem

Addresses are written using 32 hexadecimal digits. The digits are arranged into 8 groups of four to improve the readability. Groups are separated by colons. So the written form of IPv6 address looks like this:

2001:0718:1c01:0016:020d:56ff:fe77:52a3

As you can imagine DNS plays an important role in the IPv6 world, because the manual typing of IPv6 addresses is not an easy thing. Some abbreviations are allowed to lighten this task at least a little. Namely: leading zeroes in every group can be omitted. So the example address can be shortened to

2001:718:1c01:16:20d:56ff:fe77:52a3

Secondly, a sequence of all-zero groups can be replaced by pair of colons. Only one such abbreviation may occur in any address, otherwise the address would be ambiguous. This is especially handy for special-purpose addresses or address prefixes containing long sequences of zeroes. For example the loopback address

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

may be written as

::1

which is not only much shorter but also more evident. Address prefixes are usually written in the form:

prefix::/length

Where prefix defines the value of bits in the address beginning and length contains the number of important bits from the start. Because the rest of the prefix is not important, zeroes are used in this part of the address, and the "::" abbreviation is deployed. So for example prefix dedicated to the 6to4 transition mechanism is 2002::/16 which means that the starting 16 bits (two bytes, corresponding to one group in the written address) have to contain value 2002 (hexadecimal), the rest is unimportant.

Not all addresses are handled equally. IPv6 supports three different address types for which the delivery process varies:

- Unicast (individual) address: identifies one single network interface (typically a computer or similar device). The packet is delivered to this individual interface.
- Multicast (group) address: identifies group of interfaces. Data must be delivered to all group members.
- Anycast (selective) address: also identifies a group of network interfaces. But this time the packet is delivered just to one single member of the group (to the nearest one).

Broadcast as an address category is missing in IPv6, because broadcast is just a special kind of multicast. Instead of including a separate address category, IPv6 defines some standard multicast addresses corresponding to the commonly used IPv4 broadcast addresses. For example ff02::1 is the multicast address for all nodes connected to given link.

Let's look at the features of different address types in more detail.

2.3.2.1 Unicast Addresses

This is the most important address type because unicast addresses are the "normal" addresses identifying the common computers, printers and other devices connected to the network. Let's look at the most important subtype of unicast addresses first - at the global unicast addresses. The global unicast address structure is quite simple. It contains just three parts as depicted in Figure 2.12: global routing prefix, subnet ID, and interface ID.

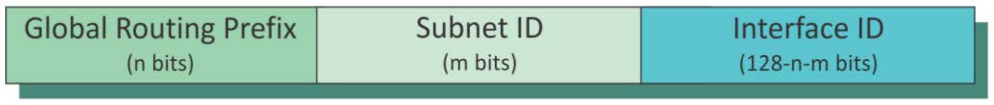


Fig. 2.12. Global unicast address format

Global Routing Prefix:
A value assigned to a site for a cluster of subnets/links. The global routing prefix is designed to be structured hierarchically by the RIRs and ISPs. It can be used to identify the ISP responsible for that IP

Subnet ID
is the identifier of a subnet. The end-network may be partitioned into to subnets (for example every building of some institution may hold separate subnet). This part of the address serves to identify individual subnets (i.e. a customer of the ISP).

Interface ID:
identifies a particular interface on an endsystem. Interface identifiers are unique inside the same subnet only, there may be devices holding the same interface ID in different subnets. Internet standards request the modified EUI-64 (described below) to play the role of interface ID.

In reality the address structure is even simpler because all used addressing schemes have the common length of the global prefix (48 bits) and subnet identifier (16 bits). In consequence the typical unicast address has the structure showed in Figure 2.13.

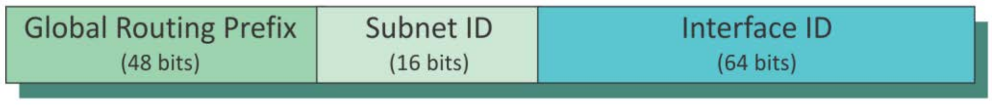


Fig. 2.13. Real-world Structure of the Global Unicast Address

Not all unicast addresses are global. Some of them are limited just to a single physical (layer 2) network. These link-local addresses are distinguished by prefix fe80::/10. They can be used for communication within a local network only – both the sender and recipient of the datagram must be connected to the same local network. A router must not forward any datagram having such a destination address.

Theses addresses are used in some mechanisms, such as the autoconfiguration of network parameters. Actually there are defined two scoping levels: link-local (prefix fe80::/10) and site-local (fec0::/10) addresses. But due to a long-term lack of consensus on the definition of "site" it was deprecated the usage of site-local addresses and prohibits new IPv6 implementations to handle the fec0::/10 prefix. The given prefix is now reserved for potential future usage.

Note:
Although, in theory, site-local addresses have been deprecated, many IPv6 implementations and IPv6 applications still use site-local prefixes and this will probably remain true for some time. Consequently, unicast addresses have just two scope levels: link-local (starting with fe80::/10) or global (all the others).

Interface Identifier – Modified EUI-64

Providing 64 bits to identify the interface in the scope of a single subnet seems to be a huge extravagance. Subnets for which 16 bits would not suffice to identify all the nodes (2^{16}) are hard to imagine. On the other hand this 64-bit long interface identifier simplifies significantly some autoconfiguration mechanisms.

EUI-64 is a network interface identifier defined by the IEEE. The modification deployed in IPv6 is related to 7th bit of the 64-bit identifier. This bit distinguishes global identifiers (world-wide unique) from the local ones (unique only in the scope of single link). The value of this bit is inverted in IPv6 addresses. Hence the value 0 of this bit means a local identifier, while a value of 1 indicates a global ID.

How do we determine the value of this final part of the IPv6 address? The answer depends on the lower-layer address which the corresponding interface has. Basic rules are following: Interface has an EUI-64 identifier This is the simplest case. The 7th bit of the existing EUI-64 identifier is inverted and the resulting value is used.

Interface has a MAC (Ethernet) address
There is a simple algorithm converting the MAC address into a modified EUI-64: the global flag (7th bit) of the MAC address is inverted and the value fffe is inserted between the 3rd and 4th byte of the MAC address. For example the MAC address 00:8c:a0:c2:71:35 is converted to interface ID 028c:a0ff:fec2:7135 (the conversion is illustrated in Figure 2.14.).

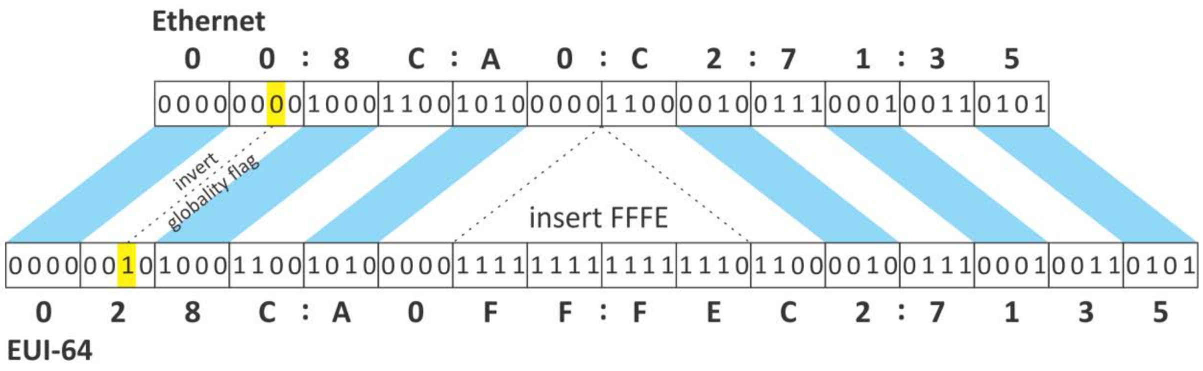


Fig. 2.14. Conversion of MAC Address to Interface Identifier

Otherwise
In other cases the network manager simply assigns some identifier to the interface. Typically some simple identifiers (like 0:0:0:1 and 0:0:0:2) are used. Such artificial identifiers are used for example for serial lines, which do not provide any values usable as a ground for the identification.

From a technical point of view this is a perfect working mechanism. But there is a hidden drawback – a threat to privacy. Since a common computer is equipped by some MAC-addressed network card, the second rule is used for the vast majority of computers. But this means that even if the user is travelling and changing the networks used to connect to the Internet, the interface identifier of his/her computer remains constant. In other words the computer can be tracked.

This issue can be solved by using several identifiers for the same interface. One of them is a fixed, EUI-64 based identifier. This is used in the "official" (DNS registered) address and is used mainly for incoming connections. The additional identifiers are randomly generated and their lifetime can be limited to a few hours or days. These identifiers are used for outgoing connections, initiated by the computer itself. Thanks to these short-lived identifiers the systematic long-term tracking of computer activities is much more difficult.

2.3.2.2 Anycast Addresses

The essential idea behind anycast is that there is a group of IPv6 nodes providing the same service. If you use an anycast address to identify this group, the request will be delivered to its nearest member using standard network mechanisms.

An anycast address is hard to distinguish. There is no separate part of the address space dedicated for these addresses; they are living in the unicast space. The local configuration is responsible for identification of anycast addresses.

Common routing methods should be used to maintain the information about the nearest anycast group member. It means that routers inside the network part containing the whole group should have a dedicated entry for this anycast address in their routing tables. This is a serious drawback for large-scale anycast deployment, where the anycast group members are spread around the global Internet. Every such anycast address involves a separate entry in global routing tables. It contradicts one of the essential IPv6 addressing design principles: hold the global routing tables as small as possible.

But this is not the sole problem of anycast addresses. Another problem is found in the heart of the anycasting mechanism. Selection of a particular receiver of an anycast-addressed datagram is left to IP, which means it is stateless. In consequence the receiver can change during running communication which can be truly confusing for the transport and application layers.

Yet another problem is related to security. How do we protect anycast groups from intruders falsely declaring themselves to be holders of given anycast addresses and stealing the data or sending false responses. It is extremely inadvisable to use IPSec to secure anycast addressed traffic (it would require all group members to use the same security parameters).

There is a research focused on solving the anycast problem. Until it succeeds, there are serious limitations to the anycast usage. Especially:

- Anycast addresses must not be used as a sender address in the IP datagram.
- Anycast addresses may be assigned to routers only. Anycast-addressed hosts are prohibited.

In summary: anycast addresses represent an experimental area where many aspects are still researched. They are already deployed in limited scope - for example anycast address is used when a mobile node looks for home agent. The scope of this address is limited to its home network only, which makes anycast perfectly usable for such application.

2.3.2.3 Multicast Addresses

Compared to anycast, multicast is a well-known entity. It is used in the contemporary IPv4 Internet, mainly to transport video/audio data in real time (e.g., videoconferencing, TV/radio broadcast). Multicast in IPv6 is just an evolution of the mechanisms already in use.

There is a separate part of the IPv6 address space dedicated to multicast. It is identified by the prefix ff00::/8. So every multicast address starts with “ff” which makes them easy to distinguish. The internal structure of the remaining 120 bits is shown in Figure 2.15.

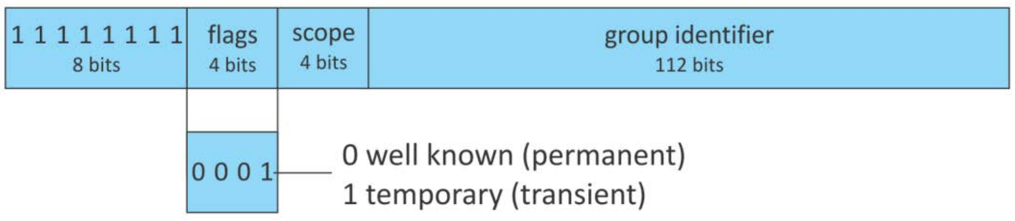


Fig. 2.15. Structure of the IPv6 Multicast Address

2.3.2.4 Required Addresses and Address Selection

There is a serious difference between IPv4 and IPv6. Every interface has just a single address in IPv4. If you want to assign more addresses to the same interface, you have to use various hacks (i.e., virtual sub-interfaces) or vendor specific implementations that do not adhere to open standards such as DHCP.

IPv6 is different. Not only does it allow you to assign more addresses to the same interface, it even urges you to do so because multiple addresses are needed for the full complement of IPv6 functionality. Required addresses for a common host (computer, printer or any other device which does not forward the datagrams) are as follows:

- link-local address for every interface
- assigned (configured) unicast and multicast addresses for the interfaces
- loopback address (::1)
- all-nodes multicast addresses (ff01::1, ff02::1)
- solicited node multicast address
- assigned multicast addresses (identifying groups to which the node belongs)

For example a PC equipped with a single Ethernet network card having a MAC address of 00:2a:0f:32:5e:d1 sitting in two subnets (2001:a:b:c::/64 and 2001:a:b:1::/64) and participating in the group ff15::1:2:3 must receive data on all these addresses:

- fe80::22a:fff:fe32:5ed1 (link-local)
- 2001:a:b:c:22a:fff:fe32:5ed1 (configured unicast)
- 2001:a:b:1:22a:fff:fe32:5ed1 (another configured unicast)
- ::1 (loopback)
- ff01::1 (all nodes on the interface)
- ff02::1 (all nodes on the link)
- ff02::1:ff32:5ed1 (solicited node multicast)
- ff15::1:2:3 (configured multicast)

A router has even more required addresses. It must support all the addresses obligatory for a node plus:

- anycast address for all routers in the subnet for every interface on which it acts as a router
- all assigned anycast addresses
- all-routers multicast (ff01::2, ff02::2, ff05::2)

Having more addresses on the same interface brings a new problem: Which one should be used? Suppose for example that you typed www.somewehere.com into your WWW browser and the request to DNS returns four different IP addresses. The selection of particular address influences datagram routing and the behaviour of the network in general. So it is quite important.

Some rules were established for this situation. It is defined an imperative algorithm to select the sender and receiver addresses for an IP datagram. The general idea behind the address selection is following. The application willing to communicate will call some system service (typically the getaddrinfo() function) to obtain a list of available addresses for given target host. It takes the first of these addresses, selects some appropriate source address and tries to establish the communication. In the case of failure the next potential destination address from the list is used. The exact rules to judge between addresses are not covered here because they are important for the implementers of IPv6, but not for its users.

2.3.2.5 Real-world Addresses

Leaving aside the addressing 'theory', in reality the IPv6 address space has been partitioned into a few areas which have a fixed meaning. You can see the allocation in Table 3-1.

Address space	Meaning
: : 0 / 128	Unspecified address
: : 1 / 128	Loopback address
ff00 : : / 8	Multicast addresses
fe80 : : / 10	Link-local addresses
fec0 : : 10	Deprecated (former site-local addresses)
other	Global unicast addresses

Table 2.5. IPv6 Address Allocation

The most important address prefix is the 2001::/16 from which all the regular global unicast addresses in the contemporary Internet originate. Allocation of addresses from this prefix is managed by Regional Internet Registries (RIRs) – the same organisations which also manage the IPv4 address space.

Every RIR has obtained some part of the 2001::/16 prefix from which it allocates smaller parts. Regions covered by a single RIR are quite large (e.g. a continent), so the allocation is not made directly. It is made in collaboration with Local Internet Registries (LIRs), which are typically the Internet providers. The mechanism is similar – every LIR obtains some part of the address space from which it allocates prefixes to its customers.

All the RIRs have agreed on common allocation rules and the address structure displayed in Figure 2.16. This clear and well-arranged structure is the result of a few years of evolution. One of the really nice features of this contemporary addressing structure is that the borders between different authorities are situated at the colons in the address written form.

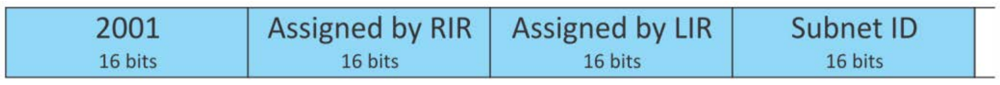


Fig. 2.16. Structure of the Real-world Global Unicast Address Prefix

The first four hexadecimal digits are fixed, they are 2001. The next four are assigned by the RIRs. It means that if some LIR asks for an address space to manage it obtains a 32-bit prefix from which 48- bit prefixes are allocated to its customers (end-sites). So the specification of the third quadruple is up to the LIR. Finally, the last foursome of hexadecimal digits in the first half of the address contains the subnet ID. This is assigned by the local network manager.

And finally we should answer the Big Principal Question: “How to get IPv6 addresses for my network?” The answer is that it depends on your position in the addressing “food chain”. If you are an endcustomer, simply ask your IPv6 connectivity provider. You should get a 48-bit prefix. If the provider tries to assign you some longer prefix (smaller address space), argue and fight for /48, because RFC 3177 clearly states that /48 should be the standard network prefix length for almost all networks.

If you are an ISP and would like to provide IPv6 addresses to your customers, your situation is more complicated. You must ask your RIR to assign you some part of the address space. It is done the usual way – by filling in an application form. But there are some requirements which you have to meet to qualify for getting some address space. In the time of writing this text the requirements are:

- You have to be an ISP.
- You must not be an end site.
- You must develop a plan to provide IPv6 connectivity to connected sites. You have to assign /48 prefixes to these sites and aggregate all these prefixes to a single routing table entry used to advertise your network to the rest of world.
- You have to have a plan for assign at least 200 prefixes to other organisations within two years.

If you meet these criteria and ask for it, you should obtain a /32 prefix to manage and use for your assignments.

2.3.2.6 IPv6 Datagram header

The format of the IPv6 packet header has been simplified from its counterpart in IPv4. The length of the IPv6 header increases to 40 bytes (from 20 bytes) and contains two 16-byte addresses (source and destination), preceded by 8 bytes of control information, as shown in Figure 2.17. The IPv4 header has two 4-byte addresses preceded by 12 bytes of control information and possibly followed by option data. The reduction of the control information and the elimination of options in the header for most IP packets are intended to optimize the processing time per packet in a router. The infrequently used fields that have been removed from the header are moved to optional extension headers when they are required.

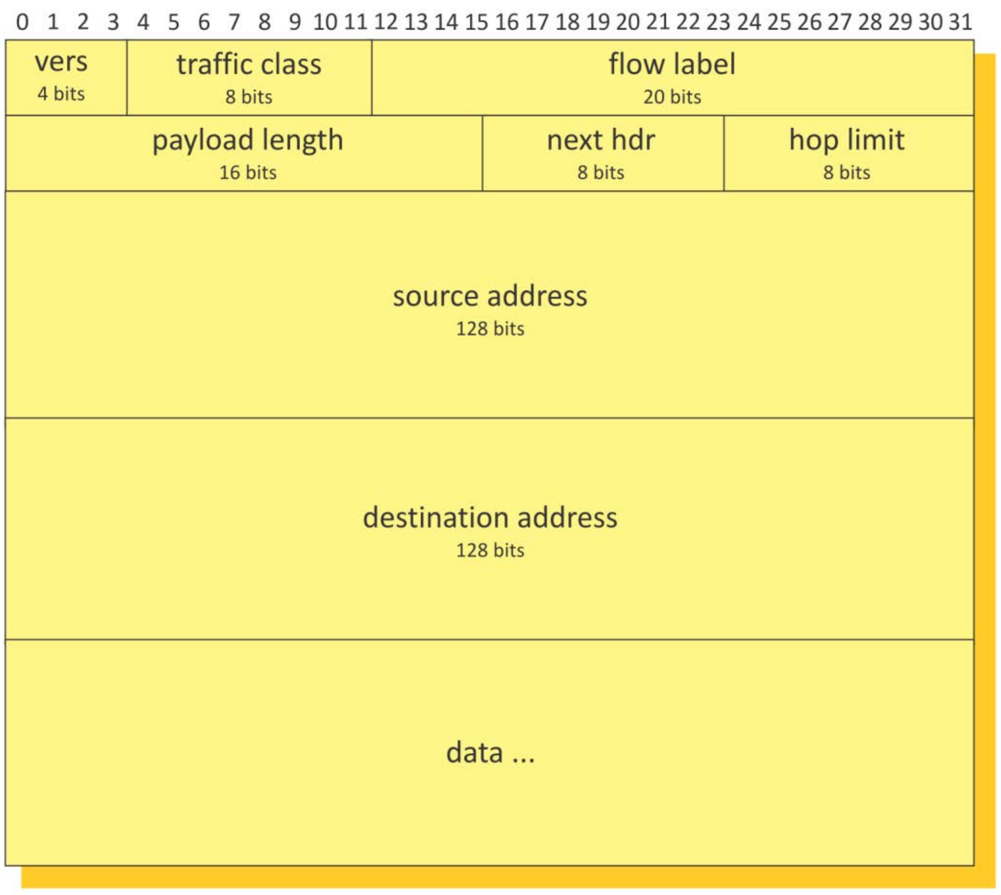


Fig. 2.17. IPv6 header format

Where:

Vers	4-bit Internet Protocol version number: 6.
Traffic class	8-bit traffic class value. allows applications to specify a certain priority for the traffic they generate, thus introducing the concept of class of service. This enables the prioritization of packets, as in Differentiated Services.
Flow label	20-bit field. IPv6 introduces the concept of a flow, which is a series of related packets from a source to a destination that requires a particular type of handling by the intervening routers, for example, real-time service.
Payload length	The length of the packet in bytes (excluding this header) encoded as a 16-bit unsigned integer. If length is greater than 64 KB, this field is 0 and an option header (Jumbo Payload) gives the true length.
Next header	Indicates the type of header immediately following the basic IP header. It can indicate an IP option header or an upper layer protocol. The protocol numbers used are the same as those used in IPv4. The next header field is also used to indicate the presence of extension headers, which provide the mechanism for appending optional information to the IPv6 packet. The following values appear in IPv6 packets, in addition to those mentioned for IPv4:

	41	Header
	45	Interdomain Routing Protocol
	46	Resource Reservation Protocol
	58	IPv6 ICMP Packet
	The following values are all extension headers:	
	0	Hop-by-Hop Options Header
	43	IPv6 Routing Header
	44	IPv6 Fragment Header
	50	Encapsulating Security Payload
	51	IPv6 Authentication Header
	59	No Next Header
	60	Destination Options Header

Hop limit This is similar to the IPv4 TTL field but it is now measured in hops and not seconds. It was changed for two reasons:

- IP normally forwards datagrams faster than one hop per second and the TTL field is always decremented on each hop, so, in practice, it is measured in hops and not seconds.
- Many IP implementations do not expire outstanding datagrams on the basis of elapsed time.

The packet is discarded after the hop limit is decremented to zero.

Source address A 128-bit address of the source host/interface.

Destination address A 128-bit address of the destination host/interface.

A comparison of the IPv4 and IPv6 header formats shows that a number of IPv4 header fields have no direct equivalents in the IPv6 header:

- Type of service
  Type of service issues in IPv6 are handled using the flow concept, “Flow labels”.
- Identification, fragmentation flags, and fragment offset
  Fragmented packets have an extension header rather than fragmentation information in the IPv6 header. This reduces the size of the basic IPv6 header, because higher-level protocols, particularly TCP, tend to avoid fragmentation of datagrams (this reduces the IPv6 header processing costs for the normal case). As noted later, IPv6 does not fragment packets en route to their destinations, only at the source.
- Header checksum
  Because transport protocols implement checksums, and because IPv6 includes an optional authentication header that can also be used to ensure integrity, IPv6 does not provide checksum monitoring of IP packets. Both TCP and UDP include a pseudo IP header in the checksums they use, so in these cases, the IP header in IPv4 is being checked twice.
  TCP and UDP, and any other protocols using the same checksum mechanisms running over IPv6, will continue to use a pseudo IP header although, obviously, the format of the pseudo IPv6 header will be different from the pseudo IPv4 header. ICMP, IGMP, and any other protocols that do not use a pseudo IP header over IPv4 use a pseudo IPv6 header in their checksums.

- Options
  All optional values associated with IPv6 packets are contained in extension headers, ensuring that the basic IP header is always the same size.

2.3.2.7 Extension headers

Every IPv6 packet starts with the basic header. In most cases, this will be the only header necessary to deliver the packet. Sometimes, however, it is necessary for additional information to be conveyed along with the packet to the destination or to intermediate systems on route (information that would previously have been carried in the Options field in an IPv4 datagram). Extension headers are used for this purpose.

Extension headers are placed immediately after the IPv6 basic packet header and are counted as part of the payload length. Each extension header (with the exception of 59) has its own 8-bit Next Header field as the first byte of the header that identifies the type of the following header. This structure allows IPv6 to chain multiple extension headers together. Figure 2.18. shows an example packet with multiple extension headers.

The length of each header varies, depending on type, but is always a multiple of 8 bytes. There are a limited number of IPv6 extension headers, any one of which can be present only once in the IPv6 packet (with the exception of the Destination Options Header, 60, which can appear more than once). IPv6 nodes that originate packets are required to place extension headers in a specific order (numeric order, with the exception of 60), although IPv6 nodes that receive packets are not required to verify that this is the case. The order is important for efficient processing at intermediate routers. Routers will generally only be interested in the hop-by-hop options and the routing header. After the router has read this far, it does not need to read further in the packet and can immediately forward. When the Next Header field contains a value other than one for an extension header, this indicates the end of the IPv6 headers and the start of the higher-level protocol data.

IPv6 allows for encapsulation of IPv6 within IPv6 (“tunnelling”). This is done with a Next Header value of 41 (IPv6). The encapsulated IPv6 packet can have its own extension headers. Because the size of a packet is calculated by the originating node to match the path MTU, IPv6 routers should not add extension headers to a packet, but instead encapsulate the received packet within an IPv6 packet of their own making (which can be fragmented if necessary).

With the exception of the hop-by-hop header (which must immediately follow the IP header if present), extension headers are not processed by any router on the packet's path except the final one.

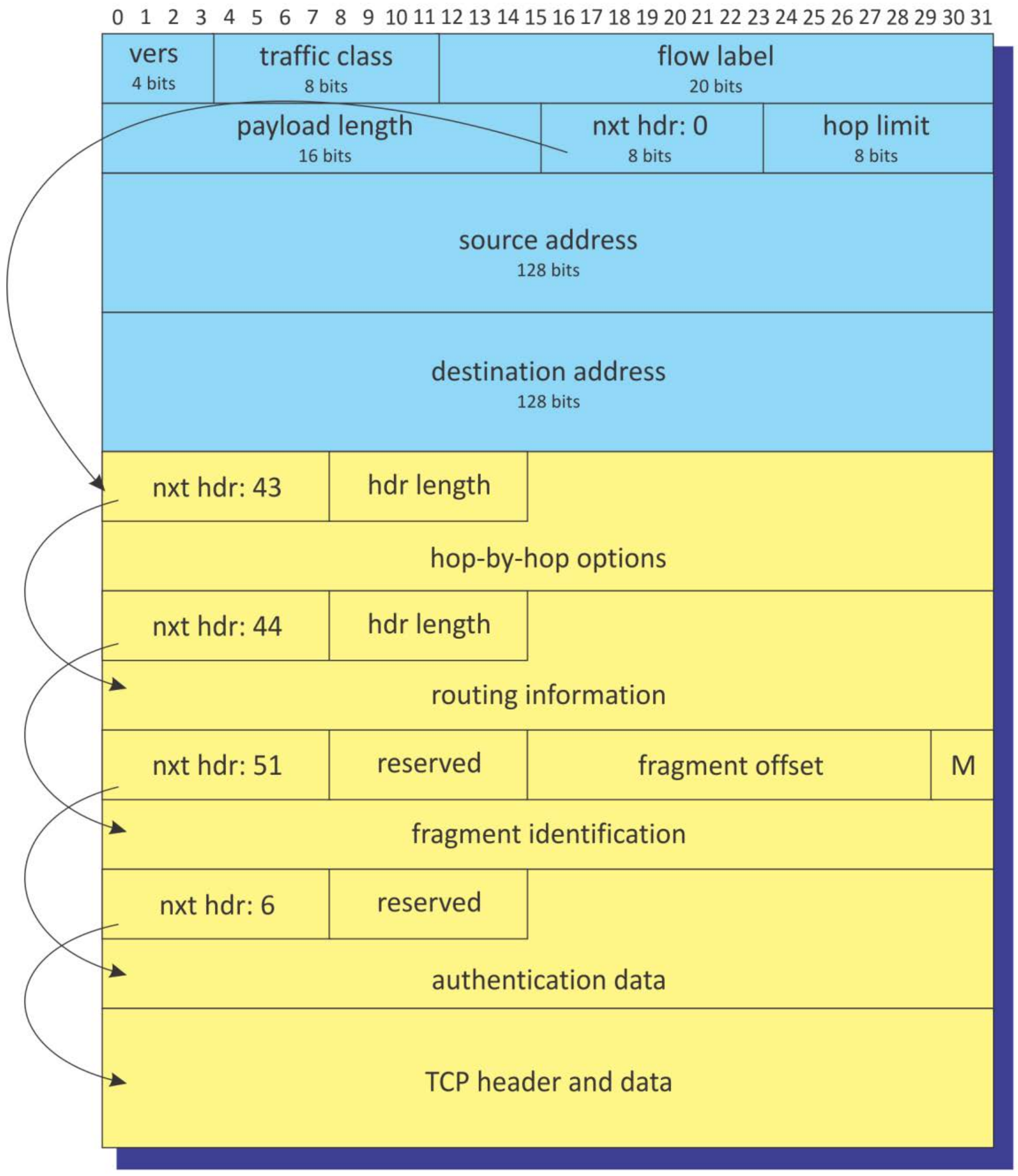


Fig. 2.18. IPv6 packet containing multiple extension headers

2.3.2.7.1 Hop-by-hop header

A hop-by-hop header contains options that must be examined by every node the packet traverses, as well as the destination node. It must immediately follow the IPv6 header (if present) and is identified by the special value 0 in the Next Header field of the IPv6 basic header. (This value is not actually a protocol number but a special case to identify this unique type of extension header).

Hop-by-hop headers contain variable length options of the format shown in Fig. 2.19. (commonly known as the Type-Length-Value (TLV) format).

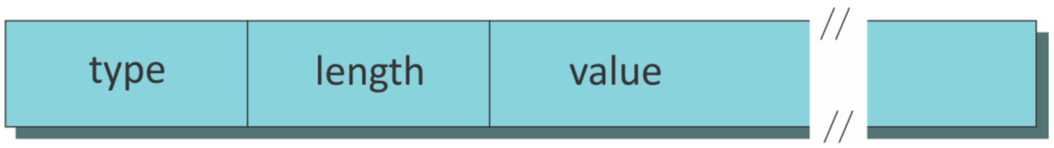


Fig. 2.19. IPv6 Type-Length-Value (TLV) option format

2.3.2.7.2 Routing header

The path that a packet takes through the network is normally determined by the network itself. Sometimes, however, the source wants more control over the route taken by the packet. It might want, for example, for certain data to take a slower but more secure route than would normally be taken. The routing header allows a path through the network to be predefined.

2.3.2.7.3 Fragment header

The source node determines the maximum transmission unit or MTU for a path before sending a packet. If the packet to be sent is larger than the MTU, the packet is divided into pieces, each of which is a multiple of 8 bytes and carries a fragment header.

2.3.2.7.4 Authentication header

The authentication header is used to ensure that a received packet has not been altered in transit and that it really came from the claimed sender. The authentication header is identified by the value 51 in the preceding Next Header field.

2.3.2.7.5 Encapsulating Security Payload

The Encapsulated Security Payload (ESP) is a special extension header, in that it can appear anywhere in a packet between the basic header and the upper layer protocol. All data following the ESP header is encrypted.

2.3.2.7.6 Destination options header

This has the same format as the hop-by-hop header, but it is only examined by the destination node or nodes. Normally, the destination options are only intended for the final destination only and the destination options header will be immediately before the upper-layer header.

2.4 Address Resolution Protocol (ARP)

Address Resolution Protocol (ARP) is a network-specific standard protocol. The address resolution protocol is responsible for converting the higher-level protocol addresses (IP addresses) to physical network addresses.

2.4.1 ARP overview

On a single physical network, individual hosts are known in the network by their physical hardware (MAC) address. Higher-level protocols address destination hosts in the form of a symbolic address (IP address in this case). When such a protocol wants to send a datagram to destination IP address w.x.y.z, the device driver does not understand this address.

Therefore, a module (ARP) is provided that will translate the IP address to the physical address of the destination host. It uses a lookup table (sometimes referred to as the ARP cache or ARP table) to perform this translation.

When the address is not found in the ARP cache, a broadcast is sent out in the network with a special format called the ARP request. If one of the machines in the network recognizes its own IP address in the request, it will send an ARP reply back to the requesting host. The reply will contain the physical hardware address of the host and source route information (if the packet has crossed bridges on its path). Both this address and the source route information are stored in the ARP cache of the requesting host. All subsequent datagrams to this destination IP address can now be translated to a physical address, which is used by the device driver to send out the datagram in the network.

An exception to the rule is the asynchronous transfer mode (ATM) technology, where ARP cannot be implemented in the physical layer as described previously. Therefore, every host, upon initialization, must register with an ARP server in order to be able to resolve IP addresses to hardware addresses.

ARP was designed to be used on networks that support hardware broadcast. This means, for example, that ARP will not work on an X.25 network.

It is important to note here that the network-layer and link-layer addresses are assigned by different authorities. For network hardware, the primary address is defined by the manufacturer of the device and is stored in permanent memory within the device, so it does not change. Thus, any protocol suite designed to operate with that particular hardware technology must make use of its particular types of addresses. This allows network-layer protocols of different protocol suites to operate at the same time. On the other hand, the IP address assigned to a network interface is installed by the user or network administrator and selected by that person to meet his or her needs. The IP addresses assigned to a portable device may, for example, be changed when it is moved. IP addresses are typically derived from a pool of addresses maintained near the network attachment point and are installed when systems are turned on or configured. When an Ethernet frame containing an IP datagram is sent from one host on a LAN to another host in the same LAN, it is the 48-bit Ethernet address that determines to which interface(s) the frame is destined.

ARP provides a dynamic mapping from a network-layer address to a corresponding hardware address. We use the term dynamic because it happens automatically and adapts to changes over time without requiring reconfiguration by a system administrator. That is, if a host were to have its network interface card changed, thereby changing its hardware address (but retaining its assigned IP address), ARP would continue to operate properly after some delay. ARP operation is normally not a concern of either the application user or the system administrator.

2.4.2 ARP detailed concept

ARP is used on IEEE 802 networks as well as on the older DIX Ethernet networks to map IP addresses to physical hardware addresses. To do this, it is closely related to the device driver for that network. In fact, the ARP specifications only describe its functionality, not its implementation. The implementation depends to a large extent on the device driver for a network type and they are usually coded together in the adapter microcode.

ARP packet generation
If an application wants to send data to a certain IP destination address, the IP routing mechanism first determines the IP address of the next hop of the packet (it can be the destination host itself, or a router) and the hardware device on which it should be sent. If it is an IEEE 802.3/4/5 network, the ARP module must be consulted to map the <protocol type, target protocol address> to a physical address.

The ARP module tries to find the address in this ARP cache. If it finds the matching pair, it gives the corresponding 48-bit physical address back to the caller (the device driver), which then transmits the packet. If it does not find the pair in its table, it discards the packet (the assumption is that a higher-level protocol will retransmit) and generates a network broadcast of an ARP request. See Fig. 2.20. for more details.

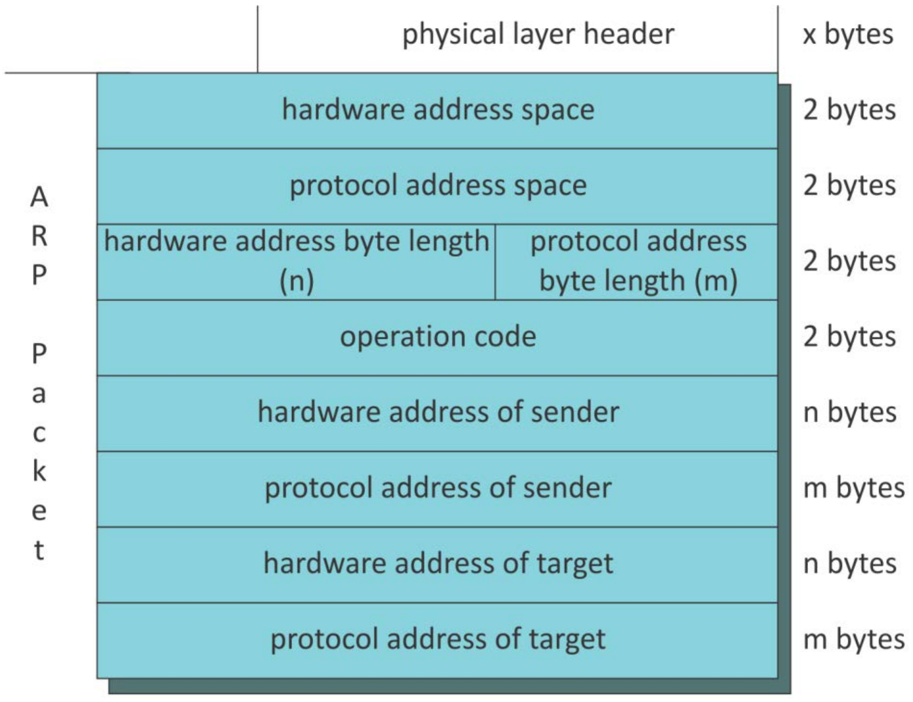

physical layer header
x bytes
hardware address space
2 bytes
protocol address space
2 bytes
hardware address byte length (n)
protocol address byte length (m)
2 bytes
operation code
2 bytes
hardware address of sender
n bytes
protocol address of sender
m bytes
hardware address of target
n bytes
protocol address of target
m bytes
ARP Packet

Fig. 2.20. ARP: Request/reply packet

Where:

- **Hardware address space**: Specifies the type of hardware; examples are Ethernet or Packet Radio Net.
- **Protocol address space**: Specifies the type of protocol, same as the EtherType field in the IEEE 802 header (IP or ARP).
- **Hardware address length**: Specifies the length (in bytes) of the hardware addresses in this packet. For IEEE 802.3 and IEEE 802.5, this is 6.
- **Protocol address length**: Specifies the length (in bytes) of the protocol addresses in this packet. For IP, this is 4.
- **Operation code**: Specifies whether this is an ARP request (1) or reply (2).
- **Source/target hardware address**: Contains the physical network hardware addresses. For IEEE 802.3, these are 48-bit addresses.
- **Source/target protocol address**: Contains the protocol addresses. For TCP/IP, these are the 32-bit IP addresses. For the ARP request packet, the target hardware address is the only undefined field in the packet.

2.4.3. ARP Cache

Essential to the efficient operation of ARP is the maintenance of an ARP cache (or table) on each host and router. This cache maintains the recent mappings from network-layer addresses to hardware addresses for each interface that uses address resolution. When IPv4 addresses are mapped to hardware addresses, the normal expiration time of an entry in the cache is 20 minutes from the time the entry was created.

We can examine the ARP cache with the arp command on Linux or in Windows. The –a option displays all entries in the cache for either system. Running arp on Linux yields the following type of output:

```
root@mail:~# arp
Address                    HWtype    HWaddress            Flags Mask    Iface
multiprez.ucv.ro           ether     00:04:61:6e:11:4d    C
10.1.2.250                 ether     00:0e:a6:81:80:ef    C             eth0
labs7bis.cs.ucv.ro         ether     00:18:74:9e:95:00    C             eth0
vl.edu.ro                  ether     00:18:74:4e:95:00    C             eth0
cab3401.cs.ucv.ro          ether     00:80:77:33:6c:ab    C             eth0
deathstar.cs.ucv.ro        ether     00:0e:a6:21:80:ef    C             eth0
gig-1-7-ucv-gw.ucv.ro      ether     00:18:74:6e:95:00    C             eth0
tdc.cs.ucv.ro              ether     00:02:b3:70:c6:42    C             eth0
ip-gate.craiova.roedu.net  ether     00:18:74:5e:95:00    C             eth0
dan.comp-craiova.ro        ether     00:1e:4f:a4:5d:48    C             eth0
193.226.37.20              ether     00:c0:df:28:f5:72    C             eth0
any.cs.ucv.ro              ether     00:15:f2:60:27:5e    C             eth0
```

Running arp on Windows provides output similar to the following:

```
C:\Users\Dan>arp -a

Interface: 193.226.37.18 --- 0x9
Internet Address       Physical Address       Type
169.254.12.97          00-18-74-2e-95-00      dynamic
169.254.59.53          00-18-74-2e-95-00      dynamic
193.226.37.1           00-18-74-2e-95-00      dynamic
193.226.37.40          00-16-76-cc-6f-c0      dynamic
193.226.37.50          00-18-74-2e-95-00      dynamic
193.226.37.78          60-eb-69-b9-5e-14      dynamic
193.226.37.125         00-02-b3-40-c6-42      dynamic
193.226.37.163         00-18-74-2e-95-00      dynamic
193.226.37.167         00-18-74-2e-95-00      dynamic
193.226.37.255         ff-ff-ff-ff-ff-ff      static
224.0.0.22             01-00-5e-00-00-16      static
224.0.0.251            01-00-5e-00-00-fb      static
224.0.0.252            01-00-5e-00-00-fc      static
239.255.255.250        01-00-5e-7f-ff-fa      static
```

2.4.4. ARP Examples

Whenever we use Internet services, such as opening a Web page with a browser, our local computer must determine how to contact the server in which we are interested. The most basic decision it makes is whether that service is local (part of the same IP subnetwork) or remote. If it is remote, a router is required to reach the destination. ARP operates only when reaching those systems on the same IP subnet.

For this example, then, let us assume that we use a Web browser to contact the following URL:

http://193.231.39.22

and let’s also assume that our PC has the IP address 193.231.39.10 and netmask 255.255.255.0. That means that our PC (193.231.39.10) and the web server (193.231.39.22) are located within the same LAN segment (share the same IPv4 prefix – 193.231.39).

Note that this URL contains an IPv4 address rather than the more common domain or host name. The reason for using the address here is to underscore the fact that our demonstration of ARP is most relevant to systems sharing the same IPv4 prefix. Here, we use a URL containing an address identifying a local Web server and explore how direct delivery operates. Such local servers are becoming more common as embedded devices such as printers and VoIP adapters include built-in Web servers for configuration.

Direct delivery takes place when an IP datagram is sent to an IP address with the same IP prefix as the sender’s. It plays an important role in the general method of forwarding of IP datagrams. The following list captures the basic operation of direct delivery with IPv4, using the previous example:

1. The application, in this case a Web browser, calls a special function to parse the URL to see if it contains a host name. Here it does not, so the application uses the 32-bit IPv4 address 193.231.39.22.
2. The application asks the TCP protocol to establish a connection with 193.231.39.22.
3. TCP attempts to send a connection request segment to the remote host by sending an IPv4 datagram to 193.231.39.22.
4. Because we are assuming that the address 193.231.39.22 is using the same network prefix as our PC, the datagram can be sent directly to that address without going through a router.
5. Assuming that Ethernet-compatible addressing is being used on the IPv4 subnet, the sending host must convert the 32-bit IPv4 destination address into a 48-bit Ethernet-style address. Generally speaking a translation is required from the logical Internet address to its corresponding physical hardware address. This is the function of ARP. ARP works in its normal form only for broadcast networks, where the link layer is able to deliver a single message to all attached network devices. This is an important requirement imposed by the operation of ARP. On non-broadcast networks (sometimes called NBMA for non-broadcast multiple access), other, more complex mapping protocols may be required.
6. ARP sends an Ethernet frame called an ARP request to every host on the shared link-layer segment. This is called a link-layer broadcast. We show the broadcast domain in Fig. 2.21. with a crosshatched box. The ARP request contains the IPv4 address of the destination host (193.231.39.22) and seeks an answer to the following question: "If you are configured with IPv4 address 193.231.39.22 as one of your own, please respond to me with your MAC address."
7. With ARP, all systems in the same broadcast domain receive ARP requests. This includes systems that may not be running the IPv4 or IPv6 protocols at all but does not include systems on different VLANs, if they are supported. Provided there exists an attached system using the IPv4 address specified in the request, it alone responds with an ARP reply. This reply contains the IPv4 address (for matching with the request) and the corresponding MAC address. The reply does not ordinarily use broadcast but is directed only to the sender. The host receiving the ARP request also learns of the sender's IPv4-to-MAC address mapping at this time and records it in memory for later use.
8. The ARP reply is then received by the original sender of the request, and the datagram that forced the ARP request/reply to be exchanged can now be sent.
9. The sender now sends the datagram directly to the destination host by encapsulating it in an Ethernet frame and using the Ethernet address learned by the ARP exchange as the destination Ethernet address. Because the Ethernet address refers only to the correct destination host, no other hosts or routers receive the datagram. Thus, when only direct delivery is used, no router is required.

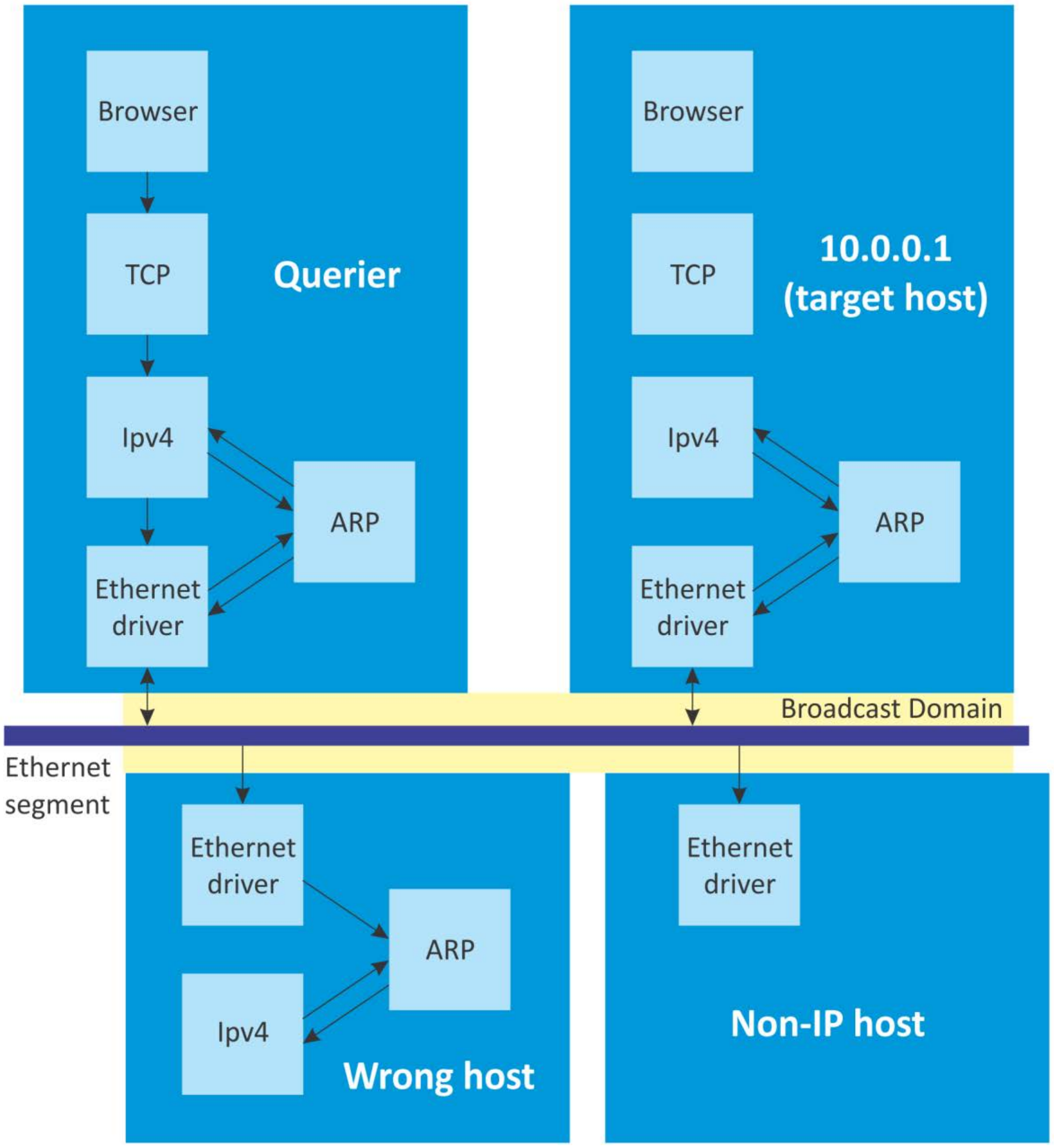


Fig. 2.21. ARP example

ARP is used in multi-access link-layer networks running IPv4, where each host has its own primary hardware address. Point-to-point links such as PPP do not use ARP. When these links are established (normally by action of the user or a system boot), the system is told of the addresses in use at each end of the link. Because hardware addresses are not involved, there is no need for address resolution or ARP.

2.4.5. Proxy ARP

Proxy ARP lets a system (generally a specially configured router) answer ARP requests for a different host. This fools the sender of the ARP request into thinking that the responding system is the destination host, when in fact the destination host may be elsewhere (or may not exist). Proxy ARP is not commonly used and is generally to be avoided if possible.

Proxy ARP has also been called promiscuous ARP or the ARP hack. These names are from a historical use of proxy ARP: to hide two physical networks from each other. In this case both physical networks can use the same IP prefix as long as a router in the middle is configured as a proxy ARP agent to respond to ARP requests on one network for a host on the other network. This technique can be used to “hide” one group of hosts from another. In the past, there were two common reasons for doing this: some systems were unable to handle subnetting, and some used an older broadcast address (a host ID of all 0 bits, instead of the current standard of a host ID with all 1 bits).

Linux supports a feature called auto-proxy ARP. It can be enabled by writing the character 1 into the file /proc/sys/net/ipv4/conf/*/proxy_arp, or by using the sysctl command. This supports the ability of using proxy ARP without having to manually enter ARP entries for every possible IPv4 address that is being proxied. Doing so allows a range of addresses, instead of each individual address, to be automatically proxied.

2.4.6. Gratuitous ARP and Address Conflict Detection (ACD)

Another feature of ARP is called gratuitous ARP. It occurs when a host sends an ARP request looking for its own address. This is usually done when the interface is configured “up” at bootstrap time.

Gratuitous ARP achieves two goals:

1. It lets a host determine if another host is already configured with the same IPv4 address. The host sending the gratuitous ARP is not expecting a reply to its request. If a reply is received, however, the error message “Duplicate IP address sent from Ethernet address . . .” is usually displayed. This is a warning to the system administrator and user that one of the systems in the same broadcast domain (e.g., LAN or VLAN) is misconfigured.
2. If the host sending the gratuitous ARP has just changed its hardware address (perhaps the host was shut down, the interface card was replaced, and then the host was rebooted), this frame causes any other host receiving the broadcast that has an entry in its cache for the old hardware address to update its ARP cache entry accordingly. As mentioned before, if a host receives an ARP request from an IPv4 address that is already in the receiver’s cache, that cache entry is updated with the sender’s hardware address from the ARP request. This is done for any ARP request received by the host; gratuitous ARP happens to take advantage of this behavior.

Although gratuitous ARP provides some indication that multiple stations may be attempting to use the same IPv4 address, it really provides no mechanism to react to the situation (other than by printing a message that is ideally acted upon by a system administrator). To deal with this issue, it was elaborated IPv4 Address Conflict Detection (ACD). ACD defines ARP probe and ARP announcement packets. An ARP probe is an ARP request packet in which the Sender's Protocol (IPv4) Address field is set to 0. Probes are used to see if a candidate IPv4 address is being used by any other systems in the broadcast domain. Setting the Sender's Protocol Address field to 0 avoids cache pollution should the candidate IPv4 address already be in use by another host, a difference from the way gratuitous ARP works. An ARP announcement is identical to an ARP probe, except both the Sender's Protocol Address and the Target Protocol Address fields are filled in with the candidate IPv4 address. It is used to announce the sender's intention to use the candidate IPv4 address as its own. To perform ACD, a host sends an ARP probe when an interface is brought up or out of sleep, or when a new link is established (e.g., when an association with a new wireless network is made). It first waits a random amount of time (in the range 0–1s, distributed uniformly) before sending up to three probe packets. The delay is used to avoid power-on congestion when multiple systems powered on simultaneously would otherwise attempt to perform ACD at once, leading to a network traffic spike. The probes are spaced randomly, with between 1 and 2s of delay (distributed uniformly) placed between. While sending its probes, a requesting station may receive ARP requests or replies. A reply to its probe indicates that a different station is already using the candidate IP address. A request containing the same candidate IPv4 address in the Target Protocol Address field sent from a different system indicates that the other system is simultaneously attempting to acquire the candidate IPv4 address. In either case, the system should indicate an address conflict message and pursue some alternative address. For example, this is the recommended behavior when being assigned an address using DHCP. It was established a limit of ten conflicts when trying to acquire an address before the requesting host enters a rate-limiting phase when it is permitted to perform ACD only once every 60s until successful.

If a requesting host does not discover a conflict according to the procedure just described, it sends two ARP announcements spaced 2s apart to indicate to systems in the broadcast domain the IPv4 address it is now using. In the announcements, both the Sender's Protocol Address and the Target Protocol Address fields are set to the address being claimed. The purpose of sending these announcements is to ensure that any preexisting cached address mappings are updated to reflect the sender's current use of the address.

ACD is considered to be an ongoing process, and in this way it differs from gratuitous ARP. Once a host has announced an address it is using, it continues inspecting incoming ARP traffic (requests and replies) to see if its address appears in the Sender's Protocol Address field. If so, some other system believes it is rightfully using the same address. In this case, there were defined three possible resolution mechanisms:

1. cease using the address,
2. keep the address but send a "defensive" ARP announcement and cease using it if the conflict continues, or
3. continue to use the address despite the conflict.

The last option is recommended only for systems that truly require a fixed, stable address (e.g., an embedded device such as a printer or router).

Although this has not traditionally been the way ARP works, there can be some benefit in doing so, at the expense of requiring all stations on the same segment to process all ARP traffic. Broadcast replies allow ACD to occur more quickly because all stations will notice the reply and invalidate their caches during a conflict.

2.5. ICMPv4 and ICMPv6: Internet Control Message Protocol

2.5.1. Introduction

The IP protocol alone provides no direct way for an end system to learn the fate of IP packets that fail to make it to their destinations. In addition, IP provides no direct way of obtaining diagnostic information (e.g., which routers are used along a path or a method to estimate the round-trip time). To address these deficiencies, a special protocol called the Internet Control Message Protocol (ICMP) is used in conjunction with IP to provide diagnostics and control information related to the configuration of the IP protocol layer and the disposition of IP packets. ICMP is often considered part of the IP layer itself, and it is required to be present with any IP implementation. It uses the IP protocol for transport. So, precisely, it is neither a network nor a transport protocol but lies somewhere between the two.

ICMP provides for the delivery of error and control messages that may require attention. ICMP messages are usually acted on by the IP layer itself, by higher-layer transport protocols (e.g., TCP or UDP), and in some cases by user applications. Note that ICMP does not provide reliability for IP. Rather, it indicates certain classes of failures and configuration information. The most common cause of packet drops (buffer overrun at a router) does not elicit any ICMP information. Other protocols, such as TCP, handle such situations.

Because of the ability of ICMP to affect the operation of important system functions and obtain configuration information, hackers have used ICMP messages in a large number of attacks. As a result of concerns about such attacks, network administrators often arrange to block ICMP messages with firewalls, especially at border routers. If ICMP is blocked, however, a number of common diagnostic utilities (e.g., ping, traceroute) do not work properly.

When discussing ICMP, we shall use the term ICMP to refer to ICMP in general, and the terms ICMPv4 and ICMPv6 to refer specifically to the versions of ICMP used with IPv4 and IPv6, respectively. As we shall see, ICMPv6 plays a far more important role in the operation of IPv6 than ICMPv4 does for IPv4.

In IPv6, ICMPv6 is used for several purposes beyond simple error reporting and signaling. It is used for Neighbor Discovery (ND), which plays the same role as ARP does for IPv4. It also includes the Router Discovery function used for configuring hosts and multicast address management. Finally, it is also used to help manage handoffs in Mobile IPv6.

2.5.2. Encapsulation in IPv4 and IPv6

ICMP messages are encapsulated for transmission within IP datagrams, as shown in Fig. 2.22. In IPv4, a Protocol field value of 1 indicates that the datagram caries ICMPv4. In IPv6, the ICMPv6 message may begin after zero or more extension headers. The last extension header before the ICMPv6 header includes a Next Header field with value 58. ICMP messages may be fragmented like other IP datagrams, although this is not common.

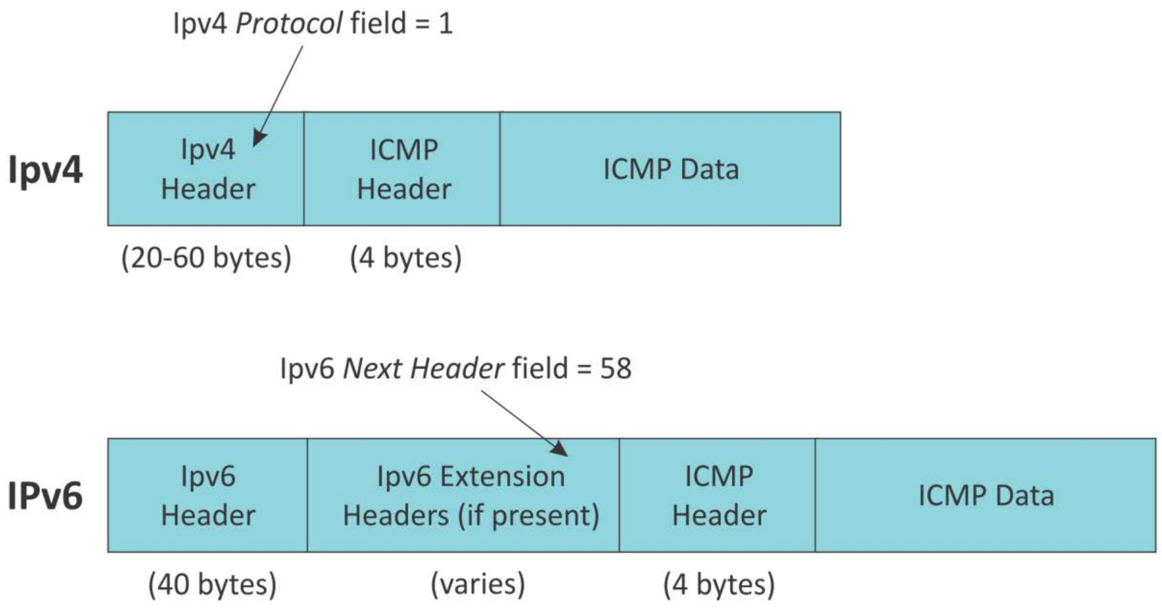


Fig. 2.22. Encapsulation of ICMP messages in IPv4 and IPv6

Fig. 2.23. shows the format of both ICMPv4 and ICMPv6 messages. The first 4 bytes have the same format for all messages, but the remainder differ from one message to the next.

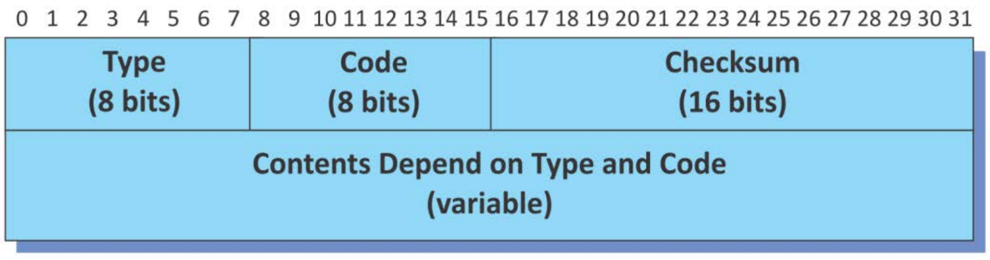


Fig. 2.23. ICMP message format

2.5.3 ICMPv4 Messages

For ICMPv4, the informational messages include Echo Request and Echo Reply (types 8 and 0, respectively), and Router Advertisement and Router Solicitation (types 9 and 10, respectively, together called Router Discovery). The most common error message types are Destination Unreachable (type 3), Redirect (type 5), Time Exceeded (type 11), and Parameter Problem (type 12).

The official list of message types is maintained by the Internet Assigned Numbers Authority (IANA). Many of these message types were defined by the original ICMPv4 specification [RFC0792] in 1981, prior to any significant experience using them. Additional experience and the development of other protocols (e.g., DHCP) have resulted in many of the messages defined then to cease being used. When IPv6 (and ICMPv6) was designed, this fact was understood, so a somewhat more rational arrangement of types and codes has been defined for ICMPv6.

2.5.4 ICMPv6 Messages

ICMPv6 is responsible not only for error and informational messages but also for a great deal of IPv6 router and host configuration.

In ICMPv6, as in ICMPv4, messages are grouped into the informational and error classes. In ICMPv6, however, all the error messages have a 0 in the high-order bit of the Type field. Thus, ICMPv6 types 0 through 127 are all errors, and types 128 through 255 are all informational. Many of the informational messages are request/reply pairs.

In comparing the common ICMPv4 messages with the ICMPv6 standard messages, we conclude that some of the effort in designing ICMPv6 was to eliminate the unused messages from the original specification while retaining the useful ones. Following this approach, ICMPv6 also makes use of the Code field, primarily to refine the meanings of certain error messages. Some standard ICMPv6 message types are: Destination Unreachable, Time Exceeded, and Parameter Problem for which more than the code value 0 has been defined.

In addition to the Type and Code fields that define basic functions in ICMPv6, a large number of standard options are also supported, some of which are required. This distinguishes ICMPv6 from ICMPv4 (ICMPv4 does not have options). Currently, standard ICMPv6 options are defined for use only with the ICMPv6 ND messages (types 135 and 136) using the Option Format field.

2.5.5 Processing of ICMP Messages

In ICMP, the processing of incoming messages varies from system to system. Generally speaking, the incoming informational requests are handled automatically by the operating system, and the error messages are delivered to user processes or to a transport protocol such as TCP. The processes may choose to act on them or ignore them. Exceptions to this general rule include the Redirect message and the Destination Unreachable—Fragmentation Required messages. The former results in an automatic update to the host’s routing table, whereas the latter is used in the path MTU discovery (PMTUD) mechanism, which is generally implemented by the transport-layer protocols such as TCP. In ICMPv6 the handling of messages has been tightened somewhat. The following rules are applied when processing incoming ICMPv6 messages:

1. Unknown ICMPv6 error messages must be passed to the upper-layer process that produced the datagram causing the error (if possible).
2. Unknown ICMPv6 informational messages are dropped.
3. ICMPv6 error messages include as much of the original (“offending”) IPv6 datagram that caused the error as will fit without making the error message datagram exceed the minimum IPv6 MTU (1280 bytes).
4. When processing ICMPv6 error messages, the upper-layer protocol type is extracted from the original or “offending” packet (contained in the body of the ICMPv6 error message) and used to select the appropriate upper-layer process. If this is not possible, the error message is silently dropped after any IPv6-layer processing.
5. There are special rules for handling errors.
6. An IPv6 node must limit the rate of ICMPv6 error messages it sends. There are a variety of ways of implementing the rate-limiting function, including the token bucket approach.

2.5.6 ICMP based tools

2.5.6.1 The “traceroute” Tool

The traceroute tool is used to determine the routers used along a path from a sender to a destination. We shall discuss the operation of the IPv4 version. The approach involves sending datagrams first with an IPv4 TTL field set to 1 and allowing the expiring datagrams to induce routers along the path to send ICMPv4 Time Exceeded (code 0) messages. Each round, the sending TTL value is increased by 1, causing the routers that are one hop farther to expire the datagrams and generate ICMP messages. These messages are sent from the router’s primary IPv4 address “facing” the sender.

2.5.6.2 The “ping” Tool

One of the most commonly used ICMP message pairs is Echo Request and Echo Response (or Reply). In ICMPv4 these are types 8 and 0, respectively, and in ICMPv6 they are types 128 and 129, respectively. ICMP Echo Request messages may be of nearly arbitrary size (limited by the ultimate size of the encapsulating IP datagram). With ICMP Echo Reply messages, the ICMP implementation is required to return any data received back to the sender, even if multiple IP fragments are involved.

As with other ICMP query/informational messages, the server must echo the Identifier and Sequence Number fields back in the reply.

These messages are sent by the ping program, which is commonly used to quickly determine if a computer is reachable on the Internet. At one time, if you could “ping” a host, you could almost certainly reach it by other means (remote login, other services, etc.). With firewalls in common use, however, this is now far from certain.

Note
The name ping is taken from the sonar operation to locate objects. The ping program was written by Mike Muuss in December 1983.

Implementations of ping set the Identifier field in the ICMP message to some number that the sending host can use to demultiplex returned responses. In UNIX-based systems, for example, the process ID of the sending process is typically placed in the Identifier field. This allows the ping application to identify the returned responses if there are multiple instances of ping running at the same time on the same host, because the ICMP protocol does not have the benefit of transport-layer port numbers. This field is often known as the Query Identifier field when referring to firewall behavior.

When a new instance of the ping program is run, the Sequence Number field starts with the value 0 and is increased by 1 every time a new Echo Request message is sent. ping prints the sequence number of each returned packet, allowing the user to see if packets are missing, reordered, or duplicated. Recall that IP (and consequently ICMP) is a best-effort datagram delivery service, so any of these three conditions can occur. ICMP does, however, include a data checksum not provided by IP.

Here is an example where an ICMPv4 Echo Request was sent to www.google.ro

```
C:\Users\Dan>ping www.google.ro -t

Pinging www.google.ro [173.194.39.184] with 32 bytes of data:
Reply from 173.194.39.184: bytes=32 time=15ms TTL=58
Reply from 173.194.39.184: bytes=32 time=16ms TTL=58
Reply from 173.194.39.184: bytes=32 time=15ms TTL=58
Reply from 173.194.39.184: bytes=32 time=18ms TTL=58
Reply from 173.194.39.184: bytes=32 time=15ms TTL=58
Reply from 173.194.39.184: bytes=32 time=16ms TTL=58
Reply from 173.194.39.184: bytes=32 time=14ms TTL=58
Reply from 173.194.39.184: bytes=32 time=42ms TTL=58
Reply from 173.194.39.184: bytes=32 time=14ms TTL=58

Ping statistics for 173.194.39.184:
    Packets: Sent = 9, Received = 9, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 14ms, Maximum = 42ms, Average = 18ms
Control-C
^C
C:\Users\Dan>
```