connection of many networks.

5.6.1 The IP Protocol

An appropriate place to start our study of the network layer in the Internet is the format of the IP datagrams themselves. An IP datagram consists of a header part and a text part. The header has a 20-byte fixed part and a variable length optional part. The header format is shown in Fig. 5-53. It is transmitted in big-endian order: from left to right, with the high-order bit of the *Version* field going first. (The SPARC is big endian; the Pentium is little-endian.) On little endian machines, software conversion is required on both transmission and reception.

The *Version* field keeps track of which version of the protocol the datagram belongs to. By including the version in each datagram, it becomes possible to have the transition between versions take years, with some machines running the old version and others running the new one. Currently a transition between IPv4 and IPv6 is going on, has already taken years, and is by no means close to being finished (Durand, 2001; Wiljakka, 2002; and Waddington and Chang, 2002). Some people even think it will never happen (Weiser, 2001). As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15.

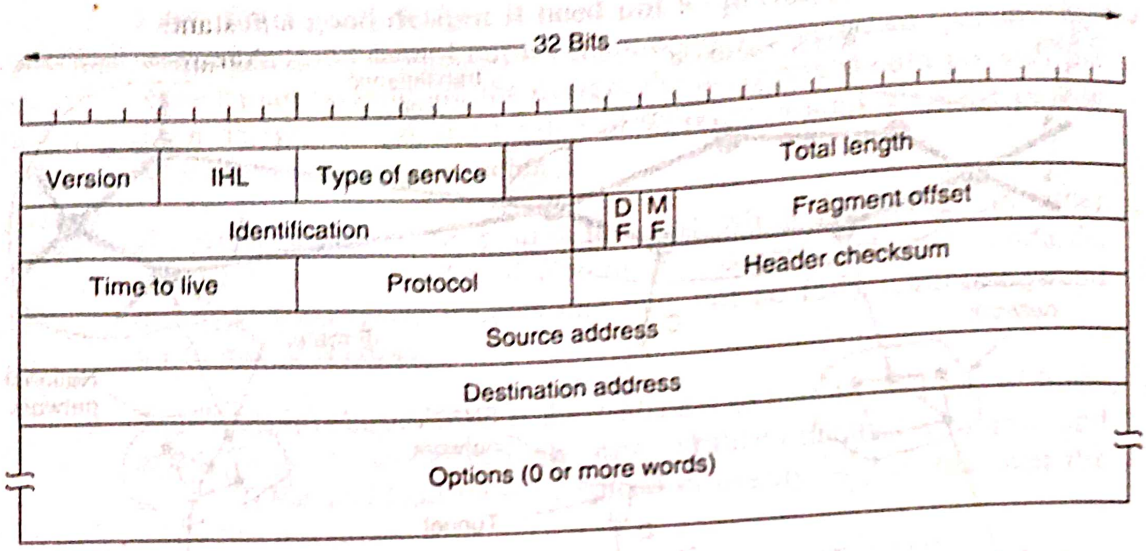

Figure 5-53. The IPv4 (Internet Protocol) header.

which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making that option useless.

The *Type of service* field is one of the few fields that has changed its meaning (slightly) over the years. It was and is still intended to distinguish between different classes of service. Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission.

Originally, the 6-bit field contained (from left to right), a three-bit *Precedence* field and three flags, *D*, *T*, and *R*. The *Precedence* field was a priority, from 0 (normal) to 7 (network control packet). The three flag bits allowed the host to specify what it cared most about from the set {Delay, Throughput, Reliability}. In theory, these fields allow routers to make choices between, for example, a satellite link with high throughput and high delay or a leased line with low throughput and low delay. In practice, current routers often ignore the *Type of service* field altogether.

Eventually, IETF threw in the towel and changed the field slightly to accommodate differentiated services. Six of the bits are used to indicate which of the service classes discussed earlier each packet belongs to. These classes include the four queueing priorities, three discard probabilities, and the historical classes.

The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future gigabit networks, larger datagrams may be needed.

The *Identification* field is needed to allow the destination host to determine which datagram a newly arrived fragment belongs to. All the fragments of a datagram contain the same *Identification* value.

sender knowing its number.

Subnets

As we have seen, all the hosts in a network must have the same network number. This property of IP addressing can cause problems as networks grow. For example, consider a university that started out with one class B network used by the Computer Science Dept. for the computers on its Ethernet. A year later, the Electrical Engineering Dept. wanted to get on the Internet, so they bought a repeater to extend the CS Ethernet to their building. As time went on, many other departments acquired computers and the limit of four repeaters per Ethernet was quickly reached. A different organization was required.

Getting a second network address would be hard to do since network addresses are scarce and the university already had enough addresses for over 60,000 hosts. The problem is the rule that a single class A, B, or C address refers to one network, not to a collection of LANs. As more and more organizations ran into this situation, a small change was made to the addressing system to deal with it.

The solution is to allow a network to be split into several parts for internal use but still act like a single network to the outside world. A typical campus network

nowadays might look like that of Fig. 5-57, with a main router connected to an ISP or regional network and numerous Ethernets spread around campus in different departments. Each of the Ethernets has its own router connected to the main router (possibly via a backbone LAN, but the nature of the interrouter connection is not relevant here).

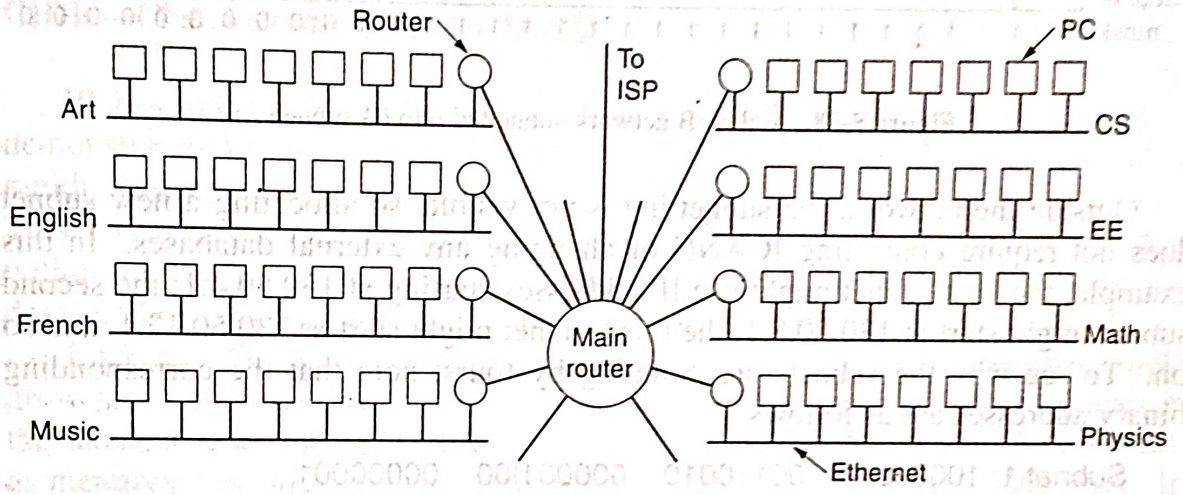

Figure 5-57. A campus network consisting of LANs for various departments.

In the Internet literature, the parts of the network (in this case, Ethernets) are called **subnets**. As we mentioned in Chap. 1, this usage conflicts with "subnet" to mean the set of all routers and communication lines in a network. Hopefully, it will be clear from the context which meaning is intended. In this section and the next one, the new definition will be the one used exclusively.

When a packet comes into the main router, how does it know which subnet (Ethernet) to give it to? One way would be to have a table with 65,536 entries in the main router telling which router to use for each host on campus. This idea would work, but it would require a very large table in the main router and a lot of manual maintenance as hosts were added, moved, or taken out of service.

Instead, a different scheme was invented. Basically, instead of having a single class B address with 14 bits for the network number and 16 bits for the host number, some bits are taken away from the host number to create a subnet number. For example, if the university has 35 departments, it could use a 6-bit subnet number and a 10-bit host number, allowing for up to 64 Ethernets, each with a maximum of 1022 hosts (0 and −1 are not available, as mentioned earlier). This split could be changed later if it turns out to be the wrong one.

To implement subnetting, the main router needs a **subnet mask** that indicates the split between network + subnet number and host, as shown in Fig. 5-58. Subnet masks are also written in dotted decimal notation, with the addition of a slash followed by the number of bits in the network + subnet part. For the example of Fig. 5-58, the subnet mask can be written as 255.255.252.0. An alternative notation is /22 to indicate that the subnet mask is 22 bits long.

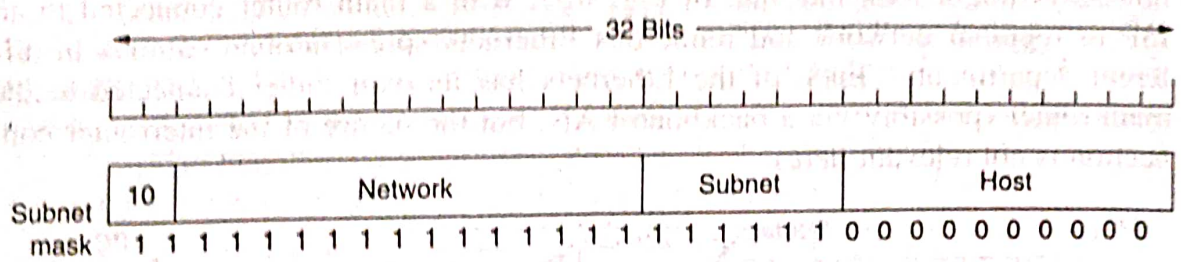

Figure 5-58. A class B network subnetted into 64 subnets.

Outside the network, the subnetting is not visible, so allocating a new subnet does not require contacting ICANN or changing any external databases. In this example, the first subnet might use IP addresses starting at 130.50.4.1; the second subnet might start at 130.50.8.1; the third subnet might start at 130.50.12.1; and so on. To see why the subnets are counting by fours, note that the corresponding binary addresses are as follows:

Subnet 1: 10000010 00110010 000001|00 00000001
Subnet 2: 10000010 00110010 000010|00 00000001
Subnet 3: 10000010 00110010 000011|00 00000001

Here the vertical bar (|) shows the boundary between the subnet number and the host number. To its left is the 6-bit subnet number; to its right is the 10-bit host number.

To see how subnets work, it is necessary to explain how IP packets are processed at a router. Each router has a table listing some number of (network, 0) IP addresses and some number of (this-network, host) IP addresses. The first kind tells how to get to distant networks. The second kind tells how to get to local hosts. Associated with each table is the network interface to use to reach the destination, and certain other information.

When an IP packet arrives, its destination address is looked up in the routing table. If the packet is for a distant network, it is forwarded to the next router on the interface given in the table. If it is a local host (e.g., on the router's LAN), it is sent directly to the destination. If the network is not present, the packet is forwarded to a default router with more extensive tables. This algorithm means that each router only has to keep track of other networks and local hosts, not (network, host) pairs, greatly reducing the size of the routing table.

When subnetting is introduced, the routing tables are changed, adding entries of the form (this-network, subnet, 0) and (this-network, this-subnet, host). Thus, a router on subnet *k* knows how to get to all the other subnets and also how to get to all the hosts on subnet *k*. It does not have to know the details about hosts on other subnets. In fact, all that needs to be changed is to have each router do a Boolean AND with the network's subnet mask to get rid of the host number and look up the resulting address in its tables (after determining which network class it is).

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-68. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which will probably take a decade, routers will be able to examine this field to tell what kind of packet they have. As an aside, making this test wastes a few instructions in the critical path, so many implementations are likely to try to avoid it by using some field in the data link header to distinguish IPv4 packets from IPv6 packets. In this way, packets can be passed to the correct network layer handler directly. However, having the data link layer be aware of network packet types completely violates the design principle that each layer should not be aware of the meaning of the bits given to it from the layer above. The discussions between the "Do it right" and "Make it fast" camps will no doubt be lengthy and vigorous.

The *Traffic class* field is used to distinguish between packets with different real-time delivery requirements. A field designed for this purpose has been in IP

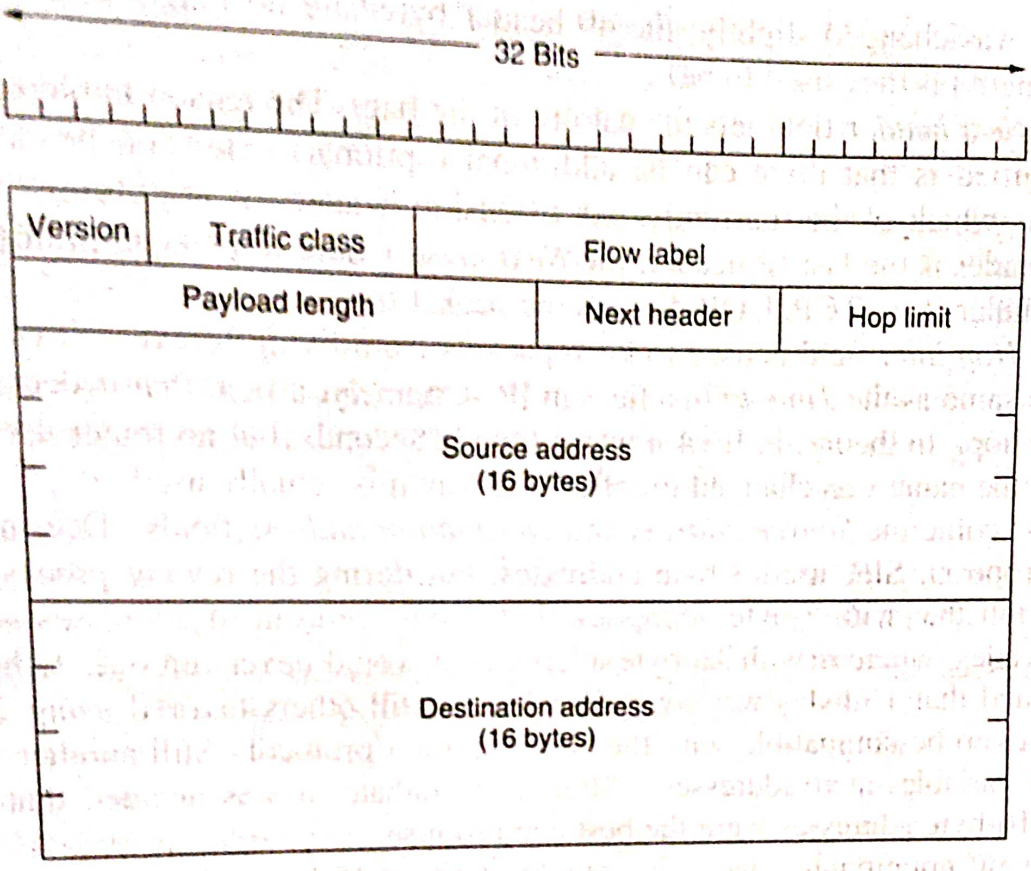

Figure 5-68. The IPv6 fixed header (required).

since the beginning, but it has been only sporadically implemented by routers. Experiments are now underway to determine how best it can be used for multimedia delivery.

The *Flow label* field is also still experimental but will be used to allow a source and destination to set up a pseudoconnection with particular properties and requirements. For example, a stream of packets from one process on a certain source host to a certain process on a certain destination host might have stringent delay requirements and thus need reserved bandwidth. The flow can be set up in advance and given an identifier. When a packet with a nonzero *Flow label* shows up, all the routers can look it up in internal tables to see what kind of special treatment it requires. In effect, flows are an attempt to have it both ways: the flexibility of a datagram subnet and the guarantees of a virtual-circuit subnet.

Each flow is designated by the source address, destination address, and flow number, so many flows may be active at the same time between a given pair of IP addresses. Also, in this way, even if two flows coming from different hosts but with the same flow label pass through the same router, the router will be able to tell them apart using the source and destination addresses. It is expected that flow labels will be chosen randomly, rather than assigned sequentially starting at 1, so routers as expected to hash them.

The *Payload length* field tells how many bytes follow the 40-byte header of Fig. 5-68. The name was changed from the IPv4 *Total length* field because the

meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be).

The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one. If this header is the last IP header, the *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to.

The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In theory, in IPv4 it was a time in seconds, but no router used it that way, so the name was changed to reflect the way it is actually used.

Next come the *Source address* and *Destination address* fields. Deering's original proposal, SIP, used 8-byte addresses, but during the review process many people felt that with 8-byte addresses IPv6 would run out of addresses within a few decades, whereas with 16-byte addresses it would never run out. Other people argued that 16 bytes was overkill, whereas still others favored using 20-byte addresses to be compatible with the OSI datagram protocol. Still another faction wanted variable-sized addresses. After much debate, it was decided that fixed-length 16-byte addresses were the best compromise.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this:

8000:0000:0000:0000:0123:4567:89AB:CDEF

Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes

8000::123:4567:89AB:CDEF

Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number, for example

::192.31.20.46

Perhaps it is unnecessary to be so explicit about it, but there are a lot of 16-byte addresses. Specifically, there are 2^{128} of them, which is approximately 3×10^{38}. If the entire earth, land and water, were covered with computers, IPv6 would allow 7×10^{23} IP addresses per square meter. Students of chemistry will notice that this number is larger than Avogadro's number. While it was not the intention to give every molecule on the surface of the earth its own IP address, we are not that far off.

In practice, the address space will not be used efficiently, just as the telephone number address space is not (the area code for Manhattan, 212, is nearly full, but

that for Wyoming, 307, is nearly empty). In RFC 3194, Durand and Huitema calculated that, using the allocation of telephone numbers as a guide, even in the most pessimistic scenario there will still be well over 1000 IP addresses per square meter of the entire earth's surface (land and water). In any likely scenario, there will be trillions of them per square meter. In short, it seems unlikely that we will run out in the foreseeable future.

It is instructive to compare the IPv4 header (Fig. 5-53) with the IPv6 header (Fig. 5-68) to see what has been left out in IPv6. The *IHL* field is gone because the IPv6 header has a fixed length. The *Protocol* field was taken out because the *Next header* field tells what follows the last IP header (e.g., a UDP or TCP segment).

All the fields relating to fragmentation were removed because IPv6 takes a different approach to fragmentation. To start with, all IPv6-conformant hosts are expected to dynamically determine the datagram size to use. This rule makes fragmentation less likely to occur in the first place. Also, the minimum has been raised from 576 to 1280 to allow 1024 bytes of data and many headers. In addition, when a host sends an IPv6 packet that is too large, instead of fragmenting it, the router that is unable to forward it sends back an error message. This message tells the host to break up all future packets to that destination. Having the host send packets that are the right size in the first place is ultimately much more efficient than having the routers fragment them on the fly.

Finally, the *Checksum* field is gone because calculating it greatly reduces performance. With the reliable networks now used, combined with the fact that the data link layer and transport layers normally have their own checksums, the value of yet another checksum was not worth the performance price it extracted. Removing all these features has resulted in a lean and mean network layer protocol. Thus, the goal of IPv6—a fast, yet flexible, protocol with plenty of address space—has been met by this design.

Extension Headers

Some of the missing IPv4 fields are occasionally still needed, so IPv6 has introduced the concept of an (optional) **extension header**. These headers can be supplied to provide extra information, but encoded in an efficient way. Six kinds of extension headers are defined at present, as listed in Fig. 5-69. Each one is optional, but if more than one is present, they must appear directly after the fixed header, and preferably in the order listed.

Some of the headers have a fixed format; others contain a variable number of variable-length fields. For these, each item is encoded as a (Type, Length, Value) tuple. The *Type* is a 1-byte field telling which option this is. The *Type* values have been chosen so that the first 2 bits tell routers that do not know how to process the option what to do. The choices are: skip the option; discard the packet; discard the packet and send back an ICMP packet; and the same as the previous

Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

Figure 5-69. IPv6 extension headers.

one, except do not send ICMP packets for multicast addresses (to prevent one bad multicast packet from generating millions of ICMP reports).

The *Length* is also a 1-byte field. It tells how long the value is (0 to 255 bytes). The *Value* is any information required, up to 255 bytes.

The hop-by-hop header is used for information that all routers along the path must examine. So far, one option has been defined: support of datagrams exceeding 64K. The format of this header is shown in Fig. 5-70. When it is used, the *Payload length* field in the fixed header is set to zero.

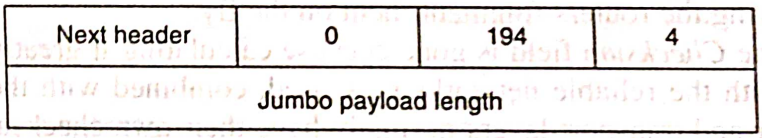

Figure 5-70. The hop-by-hop extension header for large datagrams (jumbograms).

As with all extension headers, this one starts out with a byte telling what kind of header comes next. This byte is followed by one telling how long the hop-by-hop header is in bytes, excluding the first 8 bytes, which are mandatory. All extensions begin this way.

The next 2 bytes indicate that this option defines the datagram size (code 194) and that the size is a 4-byte number. The last 4 bytes give the size of the datagram. Sizes less than 65,536 bytes are not permitted and will result in the first router discarding the packet and sending back an ICMP error message. Datagrams using this header extension are called **jumbograms**. The use of jumbograms is important for supercomputer applications that must transfer gigabytes of data efficiently across the Internet.

The destination options header is intended for fields that need only be interpreted at the destination host. In the initial version of IPv6, the only options defined are null options for padding this header out to a multiple of 8 bytes, so initially it will not be used. It was included to make sure that new routing and host software can handle it, in case someone thinks of a destination option some day.

The routing header lists one or more routers that must be visited on the way to the destination. It is very similar to the IPv4 loose source routing in that all addresses listed must be visited in order, but other routers not listed may be visited in between. The format of the routing header is shown in Fig. 5-71.

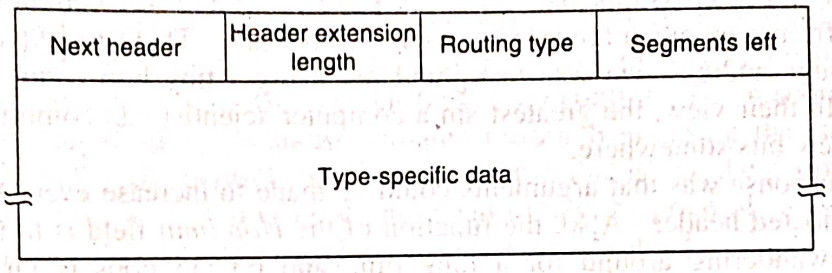

Figure 5-71. The extension header for routing.

The first 4 bytes of the routing extension header contain four 1-byte integers. The *Next header* and *Header entension length* fields were described above. The *Routing type* field gives the format of the rest of the header. Type 0 says that a reserved 32-bit word follows the first word, followed by some number of IPv6 addresses. Other types may be invented in the future as needed. Finally, the *Segments left* field keeps track of how many of the addresses in the list have not yet been visited. It is decremented every time one is visited. When it hits 0, the packet is on its own with no more guidance about what route to follow. Usually at this point it is so close to the destination that the best route is obvious.

The fragment header deals with fragmentation similarly to the way IPv4 does. The header holds the datagram identifier, fragment number, and a bit telling whether more fragments will follow. In IPv6, unlike in IPv4, only the source host can fragment a packet. Routers along the way may not do this. Although this change is a major philosophical break with the past, it simplifies the routers' work and makes routing go faster. As mentioned above, if a router is confronted with a packet that is too big, it discards the packet and sends an ICMP packet back to the source. This information allows the source host to fragment the packet into smaller pieces using this header and try again.

The authentication header provides a mechanism by which the receiver of a packet can be sure of who sent it. The encrypted security payload makes it possible to encrypt the contents of a packet so that only the intended recipient can read it. These headers use cryptographic techniques to accomplish their missions.