Module 2: Network Layer (Syllabus)

Internet Protocol: Connectionless Datagram Delivery (IPv4, IPv6) – Introduction, Connectionless Delivery System Characteristics, Purpose And Importance Of The Internet Protocol, The IP Datagram, Datagram Type Of Service And Differentiated Services, Datagram Encapsulation, Datagram Size, Network MTU and Fragmentation, Datagram Reassembly, Header Fields Used For Datagram Reassembly, Time To Live (IPv4) And Hop Limit (IPv6), Optional IP Items, Options Processing During Fragmentation.

Internet Protocol: Error And Control Messages (ICMP) – Introduction, The Internet Control Message Protocol, Error Reporting Vs. Error Correction, ICMP Message Delivery, 5 Conceptual Layering, ICMP Message Format

Virtual Network

A user thinks of an internet as a single virtual network that interconnects all hosts, and through which communication is possible; its underlying architecture is both hidden and irrelevant.

In a sense, an internet is an abstraction of physical networks because, at the lowest level, it provides the same functionality: accepting packets and delivering them. Higher levels of internet software add most of the rich functionality users perceive.

Internet Architecture And Philosophy

Conceptually, a TCPIIP internet provides three sets of services as shown in Figure 7.1; their arrangement in the figure suggests dependencies among them. At the lowest level, a connectionless delivery service provides a foundation on which everything rests. At the next level, a reliable transport service provides a higher level platform on which applications depend. We will soon explore each of these services, understand what they provide, and see the protocols associated with them.

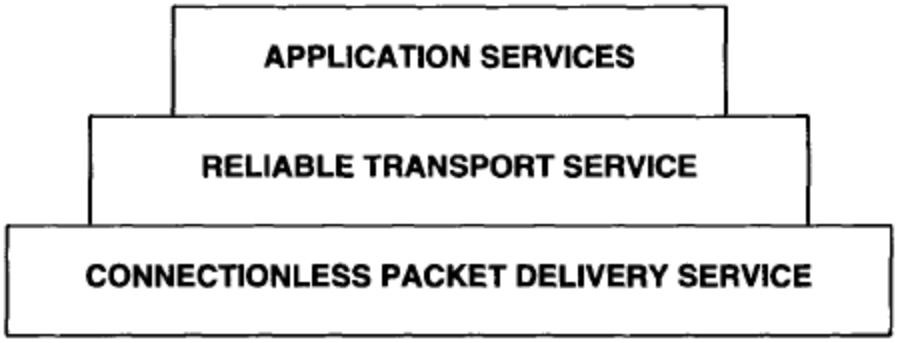

Figure 7.1 The three conceptual layers of internet services.

Connectionless Delivery System

The most fundamental internet service consists of a packet delivery system. Technically, the service is defined as an unreliable, best-effort, connectionless packet delivery system, analogous to the service provided by network hardware that operates on a best-effort delivery paradigm. The service is called unreliable because delivery is not guaranteed. The packet may be lost, duplicated, delayed, or delivered out of order, but the service will not detect such conditions, nor will it inform the sender or receiver. The service is called connectionless because each packet is treated independently from all others. A sequence of packets sent from one computer to another may travel over different paths, or some may be lost while others are delivered. Finally, the service is said to use best-effort delivery because the internet software makes an earnest attempt to deliver packets. That is, the internet does not discard packets capriciously; unreliability arises only when resources are exhausted or underlying networks fail

Purpose Of The lnternet Protocol

The protocol that defines the unreliable, connectionless delivery mechanism is called the Internet Protocol and is usually referred to by its initials, IP. IP provides three important definitions. First, the IP protocol defines the basic unit of data transfer used throughout a TCP/IP internet. Thus, it specifies the exact format of all data as it passes across the internet. Second, IP software performs the routing function, choosing a path over which data will be sent. Third, in addition to the precise, formal specification of data formats and routing, IP includes a set of rules that embody the idea of unreliable packet delivery. The rules characterize how hosts and routers

should process packets, how and when error messages should be generated, and the conditions under which packets can be discarded. IP is such a fundamental part of the design that a TCP/IP internet is sometimes called an IP-based technology

The lnternet Datagram

The internet calls its basic transfer unit an Internet datagram, sometimes referred to as an IP datagram or merely a datagram. Like a typical physical network frame, a datagram is divided into header and data areas. Also like a frame, the datagram header contains the source and destination addresses and a type field that identifies the contents of the datagram. The difference, of course, is that the datagram header contains IP addresses whereas the frame header contains physical addresses. Figure 7.2 shows the general form of a datagram

Figure 7.2 General form of an IP datagram, the TCP/IP analogy to a network frame. IP specifies the header format including the source and destination IP addresses. IP does not specify the format of the data area; it can be used to transport arbitrary data.

Datagram Format

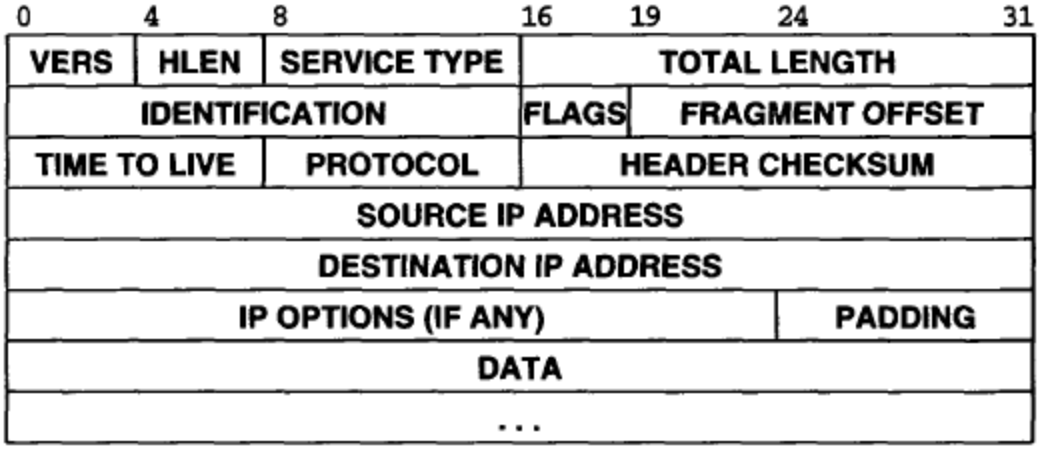

Figure 7.3 Format of an Internet datagram, the basic unit of transfer in a TCP/IP internet.

Because datagram processing occurs in software, the contents and format are not constrained by any hardware. For example, the first 4bit field in a datagram (VERS) contains the version of the IP protocol that was used to create the datagram. It is used to verify that the sender, receiver, and any routers in between them agree on the format of the datagram. All IP software is required to check the version field before processing a datagram to ensure it matches the fomlat the software expects. If standards change, machines will reject datagrams with protocol versions that differ from theirs, preventing them from misinterpreting datagram contents according to an outdated format. The current IP protocol version is 4. Consequently, the term IPv4 is often used to denote the current protocol.

The header length field (HLEN), also 4 bits, gives the datagram header length measured in 32-bit words. As we will see, all fields in the header have fixed length except for the IP OPTIONS and corresponding PADDING fields. The most common header, which contains no options and no padding, measures 20 octets and has a header length field equal to 5.

The TOTAL LENGTH field gives the length of the IP datagram measured in octets, including octets in the header and data. The size of the data area can be computed by subtracting the length of the header (HLEN) from the TOTAL LENGTH. Because the TOTAL LENGTH field is 16 bits long, the maximum possible size of an IP datagram is 216 or 65,535 octets. In most applications this is not a severe limitation. It may become more important in the future if higher speed networks can carry data packets larger than 65,535 octets.

Datagram Type Of Service And Differentiated Services

Informally called Type Of Service (TOS), the 8-bit SERVICE TYPE field specifies how the datagram should be handled. The field was originally divided into five subfields as shown in Figure 7.4:

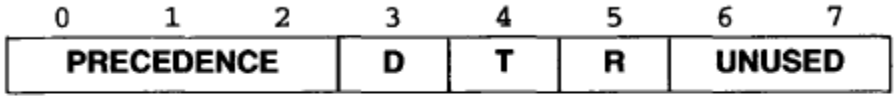

Figure 7.4 The original five subfields that comprise the 8-bit *SERVICE TYPE* field.

Three PRECEDENCE bits specify datagram precedence, with values ranging from 0 (normal precedence) through 7 (network control), allowing senders to indicate the importance of each datagram. Although some routers ignore type of service, it is an important concept because it provides a mechanism that can allow control information to have precedence over data. For example, many routers use a precedence value of 6 or 7 for routing traffic to make it possible for the routers to exchange routing information even when networks are congested.

Bits D, T, and R specify the type of transport desired for the datagram. When set, the D bit requests low delay, the T bit requests high throughput, and the R bit requests high reliability. Of course, it may not be possible for an internet to guarantee the type of transport requested (i.e., it could be that no path to the destination has the requested property). Thus, we think of the transport request as a hint to the routing algorithms, not as a demand. If a router does know more than one possible route to a given destination, it can use the type of transport field to select one with characteristics closest to those desired. For example, suppose a router can select between a

low capacity leased line or a high bandwidth (but high delay) satellite connection. Datagrams carrying keystrokes from a user to a remote computer could have the D bit set requesting that they be delivered as quickly as possible, while datagrams carrying a bulk file transfer could have the T bit set requesting that they travel across the high capacity satellite path.

In the late 1990s, the IETF redefined the meaning of the 8-bit SERVICE TYPE field to accommodate a set of diferentiated services (DS). Figure 7.5 illustrates the resulting definition.

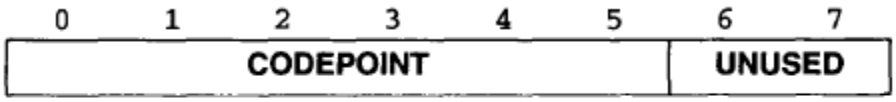

Figure 7.5 The differentiated services (DS) interpretation of the *SERVICE TYPE* field in an IP datagram.

Under the differentiated services interpretation, the first six bits comprise a codepoint, which is sometimes abbreviated DSCP (Differenciated Services Code Point) and the last two bits are left unused. A codepoint value maps to an underlying service definition, typically through an array of pointers. Although it is possible to define 64 separate services, the designers suggest that a given router will only have a few services, and multiple codepoints will map to each service. Moreover, to maintain backward compatibility with the original definition, the standard distinguishes between the first three bits of the codepoint (the bits that were formerly used for precedence) and the last three bits. When the last three bits contain zero, the precedence bits define eight broad classes of service that adhere to the same guidelines as the original definition: datagrams with a higher number in their precedence field are given preferential treatment over datagrams with a lower number. That is, the eight ordered classes are defined by codepoint values of the form:

xxx 0 0 0

where x denotes either a zero or a one.

The differentiated services design also accommodates another existing practice the widespread use of precedence 6 or 7 for routing traffic. The standard includes a special case to handle these precedence values. A router is required to implement at least two priority schemes: one for

normal traffic and one for high-priority traffic. When the last three bits of the CODEPOINT field are zero, the router must map a codepoint with precedence 6 or 7 into the higher priority class and other codepoint values into the lower priority class. Thus, if a datagram arrives that was sent using the original TOS scheme, a router using the differentiated services scheme will honor precedence 6 and 7 as the datagram sender expects

The 64 codepoint values are divided into three administrative groups as Figure 7.6 illustrates.

Pool	Codepoint	Assigned By
1	xxxxx0	Standards organization
2	xxxx11	Local or experimental
3	xxxx01	Local or experimental for now

Figure 7.6 The three administrative pools of codepoint values.

As the figure indicates, half of the values (i.e., the 32 values in pool I) must be assigned interpretations by the ETF. Currently, all values in pools 2 and 3 are available for experimental or local use. However, if the standards bodies exhaust all values in pool I, they may also choose to assign values in pool 3.

The division into pools may seem unusual because it relies on the low-order bits of the value to distinguish pools. Thus, rather than a contiguous set of values, pool I contains every other codepoint value (i.e., the even numbers between 2 and 64). The division was chosen to keep the eight codepoints corresponding to values xxx 0 0 0 in the same pool.

Whether the original TOS interpretation or the revised differentiated services interpretation is used, it is important to realize that routing software must choose from among the underlying physical network technologies at hand and must adhere to local policies. Thus, specifying a level of service in a datagram does not guarantee that routers along the path will agree to honor the request. To summarize:

We regard the service type specification as a hint to the routing algorithm that helps it choose among various paths to a destination based on local policies and its knowledge of the hardware technologies available on those paths. An internet does not guarantee to provide any particular type of service.

Datagram Encapsulation

The idea of carrying one datagram in one network frame is called encapsulation. To the underlying network, a datagram is like any other message sent from one machine to another. The hardware does not recognize the datagram format, nor does it understand the IP destination address. Thus, as Figure 7.7 shows, when one machine sends an IP datagram to another, the entire datagram travels in the data portion of the network frame

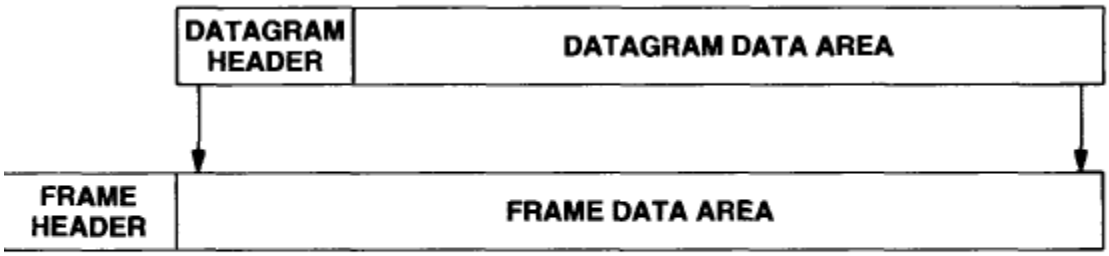

Figure 7.7 The encapsulation of an IP datagram in a frame. The physical network treats the entire datagram, including the header, as data.

Datagram Size, Network MTU, and Fragmentation

In the ideal case, the entire IP datagram fits into one physical frame, making transmission across the physical net efficient. To achieve such efficiency, the designers of IP might have selected a maximum datagram size such that a datagram would always fit into one frame. But which frame size should be chosen? After all, a datagram may travel across many types of physical networks as it moves across an internet to its final destination.

To understand the problem, we need a fact about network hardware: each packetswitching technology places a fixed upper bound on the amount of data that can be transferred in one

physical frame. For example, Ethernet limits transfers to 1500 octets of data, while FDDI permits approximately 4470 octets of data per frame. We refer to these limits as the network's maximum transfer unit or MTU. MTU sizes can be quite small: some hardware technologies limit transfers to 128 octets or less. Limiting datagram to fit the smallest possible MTU in the internet makes transfers inefficient when datagrams pass across a network that can carry larger size frames. However, allowing datagrams to be larger than the minimum network MTU in an internet means that a datagram may not always fit into a single network frame.

As Figure 7.8 illustrates, fragmentation usually occurs at a router somewhere along the path between the datagram source and its ultimate destination. The router receives a datagram from a network with a large MTU and must send it over a network for which the MTU is smaller than the datagram size.

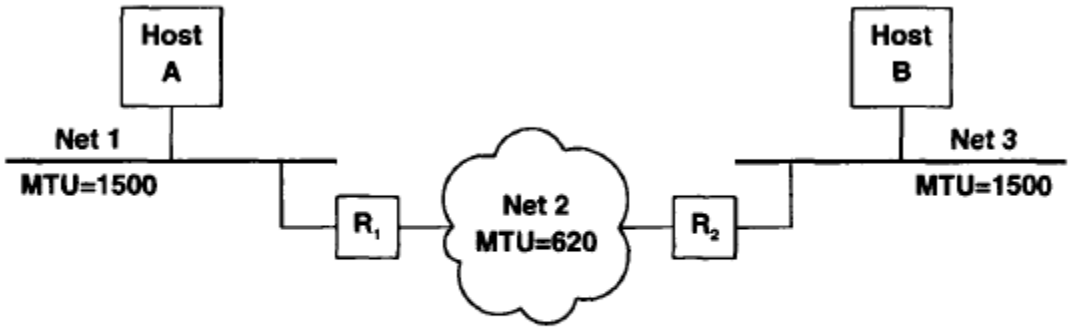

Figure 7.8 An illustration of where fragmentation occurs. Router R_1 fragments large datagrams sent from A to B; R_2 fragments large datagrams sent from B to A.

In the figure, both hosts attach directly to Ethernets which have an MTU of 1500 octets. Thus, both hosts can generate and send datagrams up to 1500 octets long. The path between them, however, includes a network with an MTU of 620. If host A sends host B a datagram larger than 620 octets, router R, will fragment the datagram. Similarly, if B sends a large datagram to A, router R, will fragment the datagram.

Fragment size is chosen so each fragment can be shipped across the underlying network in a single frame. In addition, because **IP** represents the offset of the data in multiples of eight octets,

the fragment size must be chosen to be a multiple of eight. Of course, choosing the multiple of eight octets nearest to the network MTU does not usually divide the datagram into equal size pieces; the last piece is often shorter than the others. Fragments must be reassembled to produce a complete copy of the original datagram before it can be processed at the destination.

The **IP** protocol does not limit datagrams to a small size, nor does it guarantee that large datagrams will be delivered without fragmentation. The source can choose any datagram size it thinks appropriate; fragmentation and reassembly occur automatically, without the source taking special action. The **IP** specification states that routers must accept datagrarns up to the maximum of the MTUs of networks to which they attach. In addition, a router must always handle datagrams of up to 576 octets. (Hosts are also required to accept, and reassemble if necessary, datagrams of at least 576 octets.)

Fragmenting a datagram means dividing it into several pieces. It may surprise you to learn that each piece has the same format as the original datagram. Figure 7.9 illustrates the result of fragmentation.

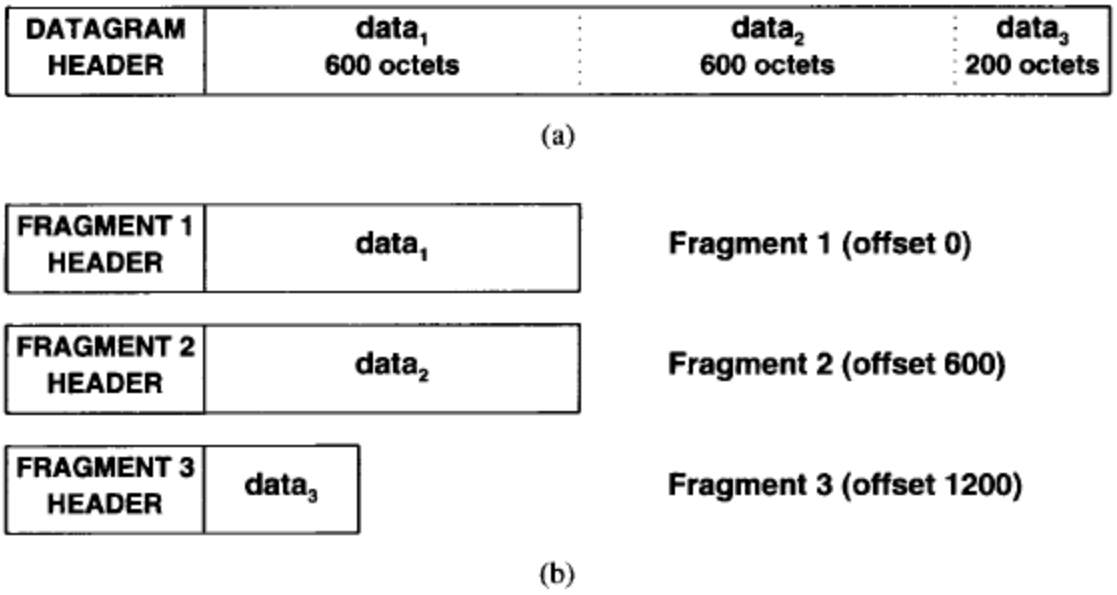

Figure 7.9 (a) An original datagram carrying 1400 octets of data and (b) the three fragments for network MTU of 620. Headers 1 and 2 have the *more fragments* bit set. Offsets shown are decimal octets; they must be divided by 8 to get the value stored in the fragment headers.

Each fragment contains a datagram header that duplicates most of the original datagram header (except for a bit in the FLAGS field that shows it is a fragment), followed by as much data as can be carried in the fragment while keeping the total length smaller than the MTU of the network over which it must travel.

Reassembly Of Fragments

In a TCP/IP internet, once a datagram has been fragmented, the fragments travel as separate datagrams all the way to the ultimate destination where they must be reassembled. Preserving fragments all the way to the ultimate destination has two disadvantages. First, because datagrams are not reassembled immediately after passing across a network with small MTU, the small fragments must be carried from the point of fragmentation to the ultimate destination. Reassembling datagrams at the ultimate destination can lead to inefficiency: even if some of the physical networks encountered after the point of fragmentation have large MTU capability, only small fragments traverse them. Second, if any fragments are lost, the datagram cannot be eassembled. The receiving machine starts a reassembly timer when it receives an initial

fragment. If the timer expires before all fragments arrive, the receiving machine dlscards the surviving pieces without processing the datagram. Thus, the probability of datagram loss increases when fragmentation occurs because the loss of a single fragment results in loss of the entire datagram.

Despite the minor disadvantages, performing reassembly at the ultimate destination works well. It allows each fragment to be routed independently, and does not require intermediate routers to store or reassemble fragments.

Fragmentation Control

Three fields in the datagram header, IDENTIFICATION, FLAGS, and FRAGMENT OFFSET, control fragmentation and reassembly of datagrams. Field IDENTIFICATION contains a unique integer that identifies the datagram. Recall that when a router fragments a datagram, it copies most of the fields in the datagram header into each fragment. Thus, the IDENTIFICATION field must be copied. Its primary purpose is to allow the destination to know which arriving fragments belong to which datagrams. As a fragment arrives, the destination uses the IDENTIFICATION field along with the datagram source address to identify the datagram. Computers sending IP datagrams must generate a unique value for the IDENTIFICATION field for each datagram?. One technique used by IP software keeps a global counter in memory, increments it each time a new datagram is created, and assigns the result as the datagram's IDENTIFICATION field.

Recall that each fragment has exactly the same format as a complete datagram. For a fragment, field FRAGMENT OFFSET specifies the offset in the original datagram of the data being carried in the fragment, measured in units of 8 octets, starting at offset zero. To reassemble the datagram, the destination must obtain all fragments starting with the fragment that has offset 0 through the fragment with highest offset. Fragments do not necessarily arrive in order, and there is no communication between the router that fragmented the datagram and the destination trying to reassemble it.

The low-order two bits of the 3-bit FLAGS field control fragmentation. Usually, application software using TCPIIP does not care about fragmentation because both fragmentation and reassembly are automatic procedures that occur at a low level in the operating system, invisible to end users. However, to test internet software or debug operational problems, it may be important to test sizes of datagrams for which fragmentation occurs. The first control bit aids in such testing by specifying whether the datagram may be fragmented. It is called the do notfragment bit because setting it to 1 specifies that the datagram should not be fragmented. An application may choose to disallow fragmentation when only the entire datagram is useful. For example, consider a bootstrap sequence in which a small embedded system executes a program in ROM that sends a request over the internet to which another machine responds by sending back a memory image. If the embedded system has been designed so it needs the entire image or none of it, the datagram should have the do not fragment bit set. Whenever a router needs to fragment a datagram that has the do not fragment bit set, the router discards the datagram and sends an error message back to the source.

The low order bit in the FLAGS field specifies whether the fragment contains data from the middle of the original datagram or from the end. It is called the more fragments bit. To see why such a bit is needed, consider the IP software at the ultimate destination attempting to reassemble a datagram. It will receive fragments (possibly out of order) and needs to know when it has received all fragments for a datagram. When a fragment arrives, the TOTAL LENGTH field in the header refers to the size of the fragment and not to the size of the original datagram, so the destination cannot use the TOTAL LENGTH field to tell whether it has collected all fragments. The more fragments bit solves the problem easily: once the destination receives a fragment with the more fragments bit turned off, it knows this fragment carries data from the tail of the original datagram. From the FRAGMENT OFFSET and TOTAL LENGTH fields, it can compute the length of the original datagram. By examining the FRAGMENT OFFSET and TOTAL LENGTH of all fragments that have arrived, a receiver can tell whether the fragments on hand contain all pieces needed to reassemble the original datagram

Time to Live (TTL)

In principle, field TIME TO LNE specifies how long, in seconds, the datagram is allowed to remain in the internet system. The idea is both simple and important: whenever a computer

injects a datagram into the internet, it sets a maximum time that the datagram should survive. Routers and hosts that process datagrams must decrement the TIME TO LNE field as time passes and remove the datagram from the internet when its time expires.

Estimating exact times is difficult because routers do not usually know the transit time for physical networks. A few rules simplify processing and make it easy to handle datagrams without synchronized clocks. First, each router along the path from source to destination is required to decrement the TIME TO LNE field by I when it processes the datagram header. Furthermore, to handle cases of overloaded routers that introduce long delays, each router records the local time when the datagram arrives, and decrements the TIME TO WEby the number of seconds the datagram remained inside the router waiting for service

Whenever a TIME TO WEfi eld reaches zero, the router discards the datagram and sends an error message back to the source. The idea of keeping a timer for datagrams is interesting because it guarantees that datagram cannot travel around an internet forever, even if routing tables become corrupt and routers route datagrams in a circle.

Although once important, the notion of a router delaying a datagram for many seconds is now outdated - current routers and networks are designed to forward each datagram within a reasonable time. If the delay becomes excessive, the router simply discards the datagram. Thus, in practice, the TIME TO WEac ts as a "hop limit" rather than an estimate of delay. Each router only decrements the value by 1.

Other Datagram Header Fields

Field PROTOCOL is analogous to the type field in a network frame; the value specifies which high-level protocol was used to create the message carried in the DATA area of the datagram. In essence, the value of PROTOCOL specifies the fomat f the DATA area. The mapping between a

high level protocol and the integer value used in the PROTOCOL field must be administered by a central authority to guarantee agreement across the entire Internet.

Field HEADER CHECKSUM ensures integrity of header values. The IP checksum is formed by treating the header as a sequence of 16-bit integers (in network byte order), adding them together using one's complement arithmetic, and then taking the one's complement of the result. For purposes of computing the checksum, field HEADER CHECKSUM is assumed to contain zero.

It is important to note that the checksum only applies to values in the IP header and not to the data. Separating the checksum for headers and data has advantages and disadvantages. Because the header usually occupies fewer octets than the data, having a separate checksum reduces processing time at routers which only need to compute header checksums. The separation also allows higher level protocols to choose their own checksum scheme for the data. The chief disadvantage is that higher level protocols are forced to add their own checksum or risk having corrupted data go undetected.

Fields SOURCE IP ADDRESS and DESTINATION IP ADDRESS contain the 32-bit IP addresses of the datagram's sender and intended recipient. Although the datagram may be routed through many intermediate routers, the source and destination fields never change; they speclfy the IP addresses of the original source and ultimate destination

The field labeled DATA in Figure 7.3 shows the beginning of the data area of the datagram. Its length depends, of course, on what is being sent in the datagram. The IP OPTIONS field, discussed below, is variable length. The field labeled PADDING, depends on the options selected. It represents bits containing zero that may be needed to ensure the datagram header extends to an exact multiple of 32 bits (recall that the header length field is specified in units of 32-bit words).

Internet Datagram Options

The IP OPTIONS field following the destination address is not required in every datagram; options are included primarily for network testing or debugging. Options processing is an integral part of the IP protocol, however, so all standard implementations must include it.

The length of the IP OPTIONS field varies depending on which options are selected. Some options are one octet long; they consist of a single octet option code. Other options are variable length. When options are present in a datagram, they appear contiguously, with no special separators between them. Each option consists of a single octet option code, which may be followed by a single octet length and a set of data octets for that option. The option code octet is divided into three fields as Figure 7.10 shows.

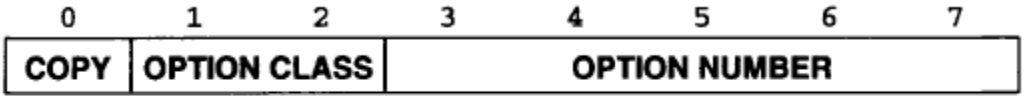

Figure 7.10 The division of the option code octet into three fields of length 1, 2, and 5 bits.

The fields of the OPTION CODE consist of a 1-bit COPY flag, a 2-bit OPTION CLASS, and the 5-bit OPTION NUMBER. The COPY flag controls how routers treat options during fragmentation. When the COPY bit is set to I, it specifies that the option should be copied into all fragments. When set to 0, the COPY bit means that the option should only be copied into the first fragment and not into all fragments.

(i)Record Route Option

The record route option allows the source to create an empty list of IP addresses and arrange for each router that handles the datagram to add its IP address to the list. Figure 7.13 shows the format of the record route option.

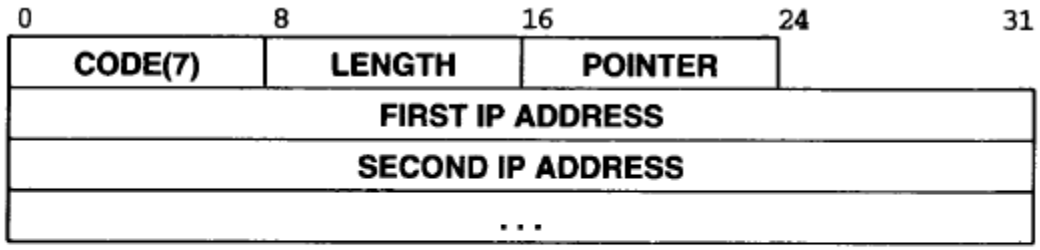

Figure 7.13 The format of the record route option in an IP datagram. The option begins with three octets immediately followed by a list of addresses. Although the diagram shows addresses in 32 bit units, they are not aligned on any octet boundary in a datagram.

(ii)Source Route Options

The idea behind source routing is that it provides a way for the sender to dictate a path through the internet. For example, to test the throughput over a particular physical network, N, system administrators can use source routing to force IP datagrams to traverse network N even if routers would normally choose a path that did not include it.

(iii) Timestamp Option

The timestamp option works like the record route option in that the timestamp option contains an initially empty list, and each router along the path from source to destination fills in one item in the list. Each entry in the list contains two 32-bit items: the IP address of the router that supplied the entry and a 32-bit integer timestamp

ICMP (The lnternet Control Message Protocol)

To allow routers in an internet to report errors or provide information about unexpected circumstances, the designers added a special-purpose message mechanism to the TCP/IP protocols. The mechanism, known as the Internet Control Message Protocol (ICMP), is considered a required part of IP and must be included in every IP implementation.

The Internet Control Message Protocol allows routers to send error or control messages to other routers or hosts; ICMP provides communication between the Internet Protocol sofrware on one machine and the Internet Protocol sofrware on another.

Error Reporting vs. Error Correction

Technically, ICMP is an error reporting mechanism. It provides a way for routers that encounter an error to report the error to the original source. Although the protocol specification outlines intended uses of ICMP and suggests possible actions to take in response to error reports, ICMP does not fully specify the action to be taken for each possible error. In short,

When a datagram causes an error, ICMP can only report the error condition back to the original source of the datagram; the source must relate the error to an individual application program or take other action to correct the problem

ICMP Message Delivery

ICMP messages require two levels of encapsulation as Figure 9.1 shows. Each ICMP message travels across the internet in the data portion of an IP datagram, which itself travels across each physical network in the data portion of a frame. Datagrams carryin ICMP messages are routed exactly like datagrams carrying information for users; i ere is no additional reliability or priority. Thus, error messages themselves may be lost dr discarded. Furthermore, in an already congested network, the error message may cause additional congestion. An exception is made to the error handling procedures if an IP datagram carrying an ICMP message causes an error. The exception, established to avoid the problem of having error messages about error messages, specifies that ICMP messages are not generated for errors that result from datagrams carrying ICMP error messages.

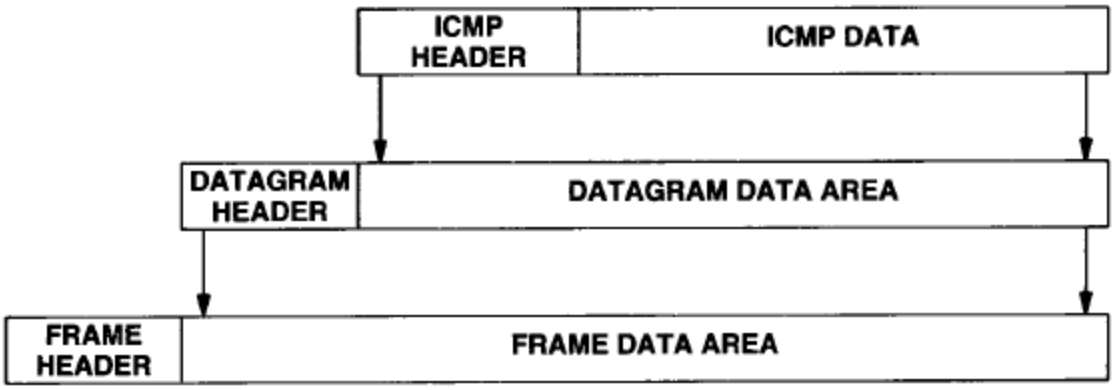

Figure 9.1 Two levels of ICMP encapsulation. The ICMP message is encapsulated in an IP datagram, which is further encapsulated in a frame for transmission. To identify ICMP, the datagram protocol field contains the value *1*.

It is important to keep in mind that even though ICMP messages are encapsulated and sent using IP, ICMP is not considered a higher level protocol - it is a required part of IP. The reason for using IP to deliver ICMP messages is that they may need to travel across several physical networks to reach their final destination. Thus, they cannot be delivered by the physical transport alone.

ICMP Message Format

Although each ICMP message has its own format, they all begin with the same three fields: an 8-bit integer message TYPE field that identifies the message, an 8-bit CODE field that provides further information about the message type, and a 16-bit CHECKSUM field (ICMP uses the same additive checksum algorithm as IP, but the ICMP checksum only covers the ICMP message). In addition, ICMP messages that report errors always include the header and first 64 data bits of the datagram causing the problem.

The reason for returning more than the datagram header alone is to allow the receiver to determine more precisely which protocol(s) and which application program were responsible for the datagram. As we will see later, higher-level protocols in the TCPIIP suite are designed so that crucial information is encoded in the fist 64 bits.

The ICMP TYPE field defines the meaning of the message as well as its format. The types include:

Type Field	ICMP Message Type
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request (obsolete)
16	Information Reply (obsolete)
17	Address Mask Request
18	Address Mask Reply

Testing Destination Reachability And Status (Ping)

TCP/IP protocols provide facilities to help network managers or users identlfy network problems. One of the most frequently used debugging tools invokes the ICMP echo request and echo reply messages. A host or router sends an ICMP echo request message to a specified destination. Any machine that receives an echo request formulates an echo reply and returns it to the original sender. The request contains an optional data area; the reply contains a copy of the data sent in the request. The echo request and associated reply can be used to test whether a destination is reachable and responding. Because both the request and reply travel in IP datagrams, successful receipt of a reply verifies that major pieces of the transport system work. First, IP software on the source computer must route the datagram. Second, intermediate routers between the source and destination must be operating and must route the datagram correctly. Third, the destination machine must be running (at least it must respond to interrupts), and both ICMP and IP software must be working. Finally, all routers along the return path must have correct routes.

Echo Request And Reply Message Format

Figure 9.2 shows the format of echo request and reply messages

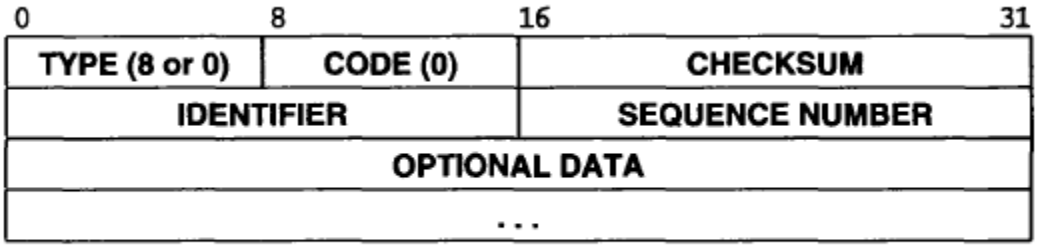

Figure 9.2 ICMP echo request or reply message format.

The field listed as OPTIONAL DATA is a variable length field that contains data to be returned to the sender. An echo reply always returns exactly the same data as was received in the request. Fields IDENTIFIER and SEQUENCE NUMBER are used by the sender to match replies to requests. The value of the TYPE field specifies whether the message is a request (8) or a reply (0).