Chapter 8

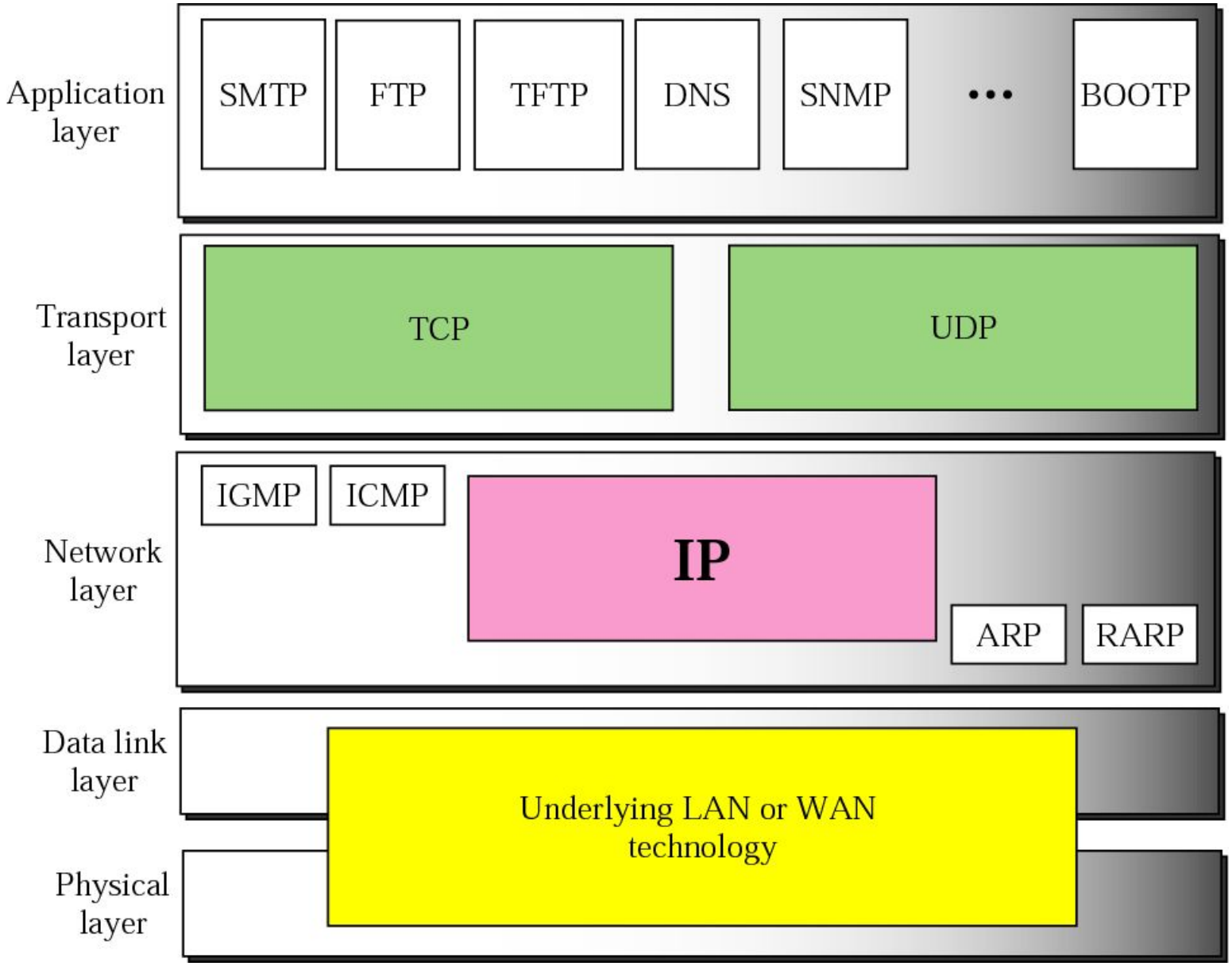
Internet Protocol (IP)

CONTENTS

- **DATAGRAM**
- **FRAGMENTATION**
- **OPTIONS**
- **CHECKSUM**

Figure 8-1

Position of IP in TCP/IP protocol suite



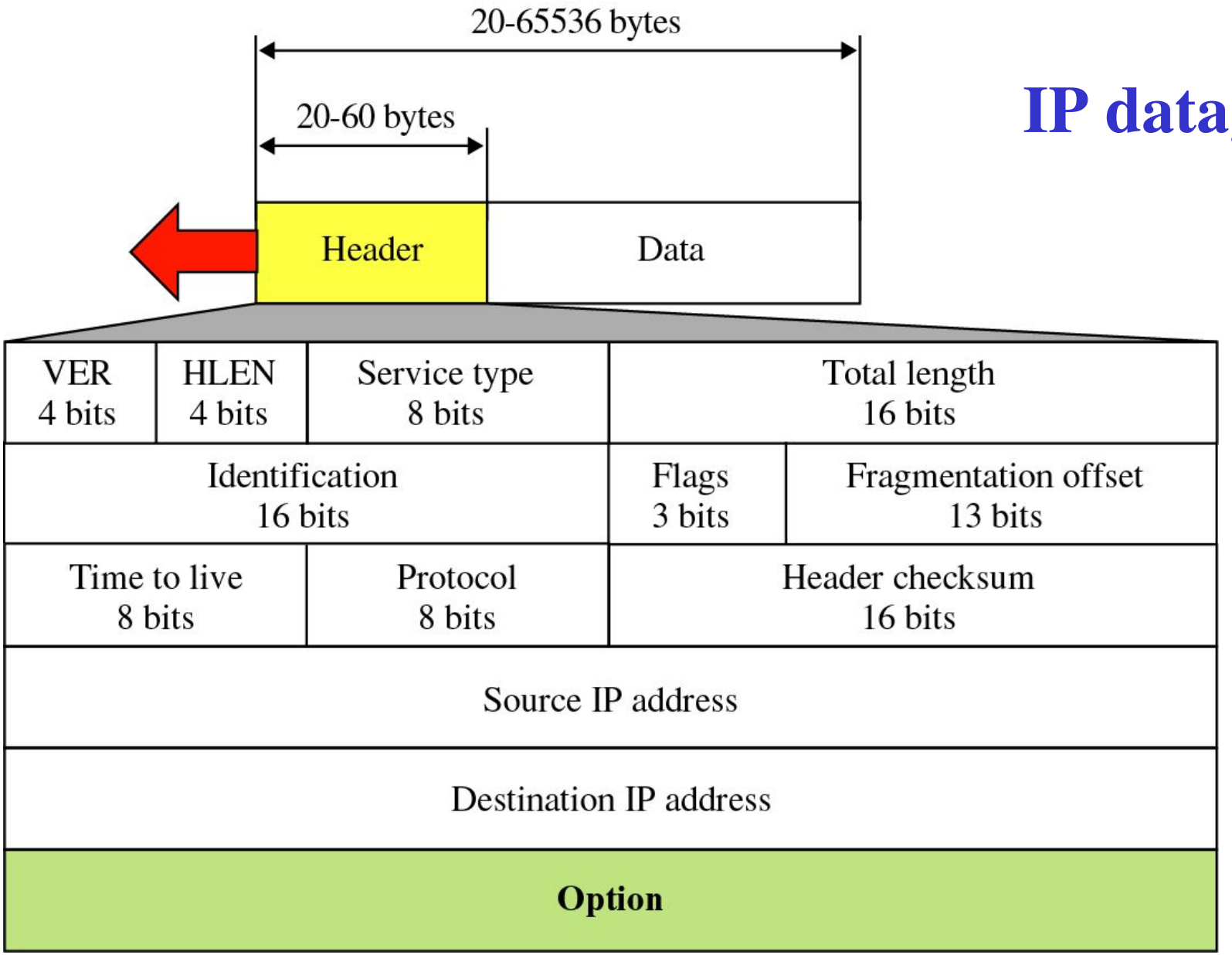
8.1

DATAGRAM

Packets in the network layer are called datagrams

Figure 8-2

IP datagram



- **Version (VER). This 4-bit field defines the version of the IP protocol.** Currently

the version is 4. However, version 6 (or IPv6) may totally replace version 4 in the future. This field tells the IP software running in the processing machine that the datagram has the format of version 4. All fields must be interpreted as specified in the fourth version of the protocol. If the machine is using some other version of IP, the datagram is discarded rather than interpreted incorrectly.

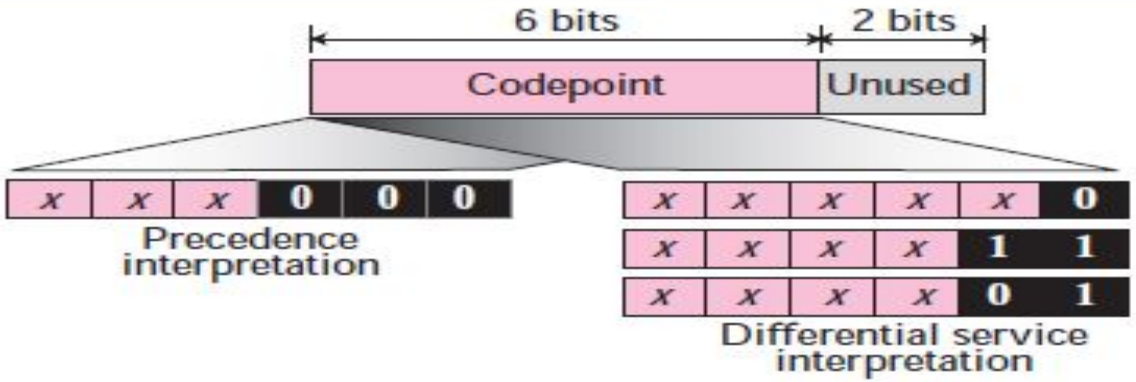
- **Header length (HLEN). This 4-bit field defines the total length of the datagram** header in 4-byte words. This field is needed because the length of the header is variable (between 20 and 60 bytes). When there are no options, the **header length** is 20 bytes, and the value of this field is 5 (5 × 4= 20). When the option field is at its maximum size, the value of this field is 15 (15 × 4= 60).

- **Service type.** In the original design of IP header, this field was referred to as **type of service (TOS)**, which defined how the datagram should be handled. Part of the field was used to define the precedence of the datagram; the rest defined the type of service (low delay, high throughput, and so on). IETF has changed the interpretation of this 8-bit field. This field now defines a set of **differentiated services.** The new interpretation is shown in Figure 7.3.

  In this interpretation, the first 6 bits make up the **codepoint** subfield and the last 2 bits are not used. The codepoint subfield can be used in two different ways.

  a. When the 3 right-most bits are 0s, the 3 left-most bits are interpreted the same as the precedence bits in the service type interpretation. In other words, it is compatible with the old interpretation. The precedence defines the eight-level

Figure 7.3 *Service type*



priority of the datagram (0 to 7) in issues such as congestion. If a router is congested and needs to discard some datagrams, those datagrams with lowest precedence are discarded first. Some datagrams in the Internet are more important than the others. For example, a datagram used for network management is much more urgent and important than a datagram containing optional information for a group.

b. When the 3 right-most bits are not all 0s, It defines services based on the priority assignment by the Internet or local authorities according to Table 7.1. The first category contains 24 service types; the second and the third each contain 16. The first category is assigned by the Internet authorities (IETF). The second category can be used by local authorities (organizations). The third category is temporary and can be used for experimental purposes. Note that these assignments have not yet been finalized.

Table 7.1 *Values for codepoints*

Category	*Codepoint*	*Assigning Authority*
1	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental

- **Total length.** This is a 16-bit field that defines the total length (header plus data) of the IP datagram in bytes. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by four.

$$\text{Length of data} = \text{total length} - \text{header length}$$

Since the field length is 16 bits, the total length of the IP datagram is limited to 65,535 ($2^{16} - 1$) bytes, of which 20 to 60 bytes are the header and the rest is data from the upper layer.

The *total length* field defines the total length of the datagram including the header.

- **Identification.** This field is used in fragmentation (discussed in the next section).
- **Flags.** This field is used in fragmentation (discussed in the next section).
- **Fragmentation offset.** This field is used in fragmentation (discussed in the next section).

- **Time to live.** A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero. However, for this scheme, all the machines must have synchronized clocks and must know how long it takes for a datagram to go from one machine to another. Today, this field is mostly used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routes between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

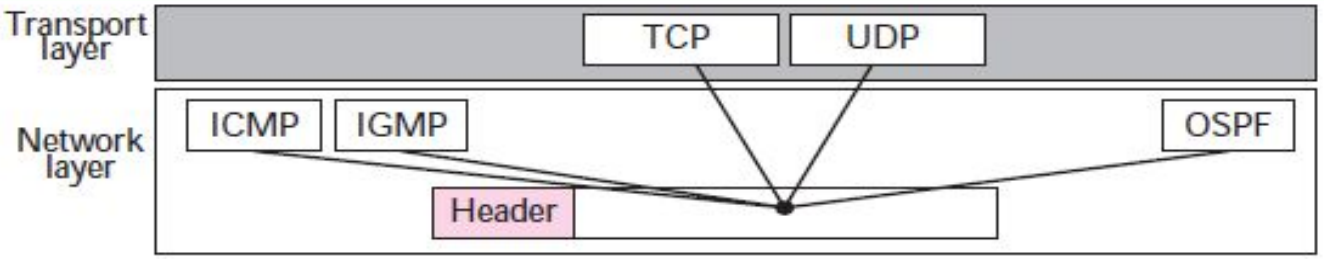
  This field is needed because routing tables in the Internet can become corrupted. A datagram may travel between two or more routers for a long time without ever getting delivered to the destination host. This field limits the lifetime of a datagram.

  Another use of this field is to intentionally limit the journey of the packet. For example, if the source wants to confine the packet to the local network, it can store

  1 in this field. When the packet arrives at the first router, this value is decremented to 0, and the datagram is discarded.

- **Protocol.** This 8-bit field defines the higher-level protocol that uses the services of the IP layer. An IP datagram can encapsulate data from several higher level protocols such as TCP, UDP, ICMP, and IGMP. This field specifies the final destination protocol to which the IP datagram should be delivered. In other words, since the IP protocol multiplexes and demultiplexes data from different higher-level protocols, the value of this field helps in the demultiplexing process when the datagram arrives at its final destination (see Figure 7.5).

Figure 7.5 *Multiplexing*



Some of the value of this field for different higher-level protocols is shown in Table 7.2.

Table 7.2 *Protocols*

Value	*Protocol*	*Value*	*Protocol*
1	ICMP	17	UDP
2	IGMP	89	OSPF
6	TCP		

- **Checksum.** The checksum concept and its calculation are discussed later in this chapter.
- **Source address.** This 32-bit field defines the IP address of the source. This field must remain unchanged during the time the IP datagram travels from the source host to the destination host.
- **Destination address.** This 32-bit field defines the IP address of the destination. This field must remain unchanged during the time the IP datagram travels from the source host to the destination host.

Example 7.1

An IP packet has arrived with the first 8 bits as shown:

```
01000010
```

The receiver discards the packet. Why?

Example 7.1

An IP packet has arrived with the first 8 bits as shown:

```
01000010
```

The receiver discards the packet. Why?

Solution

There is an error in this packet. The 4 left-most bits (0100) show the version, which is correct. The next 4 bits (0010) show the wrong header length ($2 \times 4 = 8$). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example 7.2

In an IP packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

Solution

The HLEN value is 8, which means the total number of bytes in the header is 8×4 or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

Note

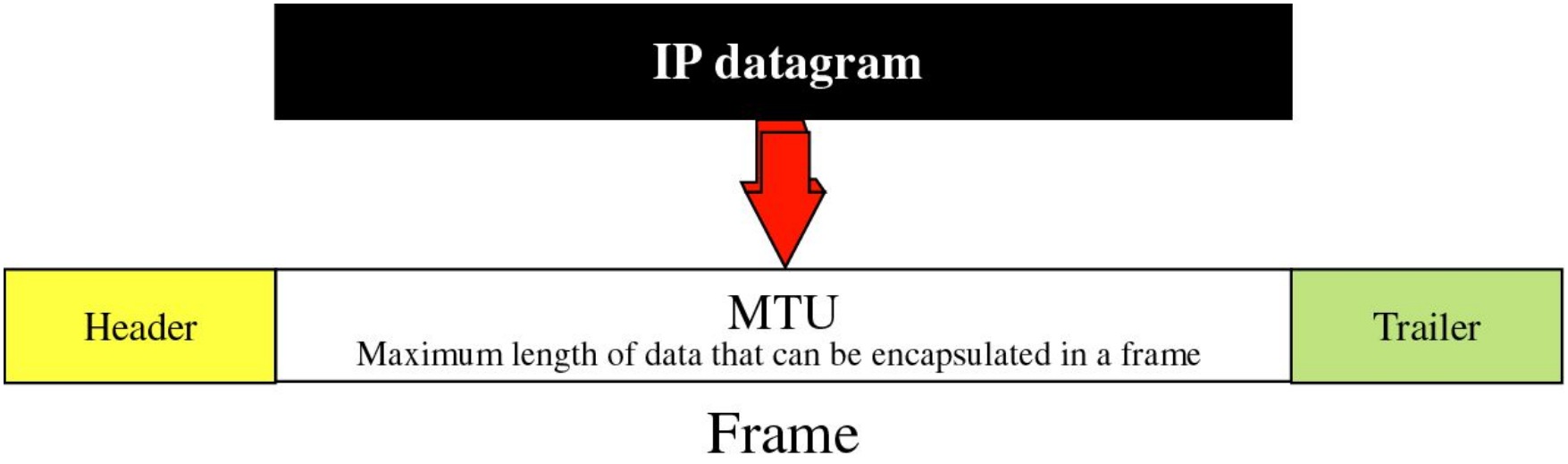
The total length field defines the total length of the datagram including the header.

8.2

FRAGMENTATION

Figure 8-6

MTU



A maximum transmission unit (MTU) is the largest size packet or frame, specified in octets (eight-bit bytes), that can be sent in a packet- or frame-based network such as the Internet.

Fragmentation (cont)

- A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU.
- In other words, a datagram can be fragmented several times before it reaches the final destination.

Fragmentation (cont)

- The reassembly of the datagram, however, is done only by the destination host because each fragment becomes an independent datagram.
- Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all of the fragments belonging to the same datagram should finally arrive at the destination host.
- So it is logical to do the reassembly at the final destination.

Fragmentation(cont)

- When a datagram is fragmented, required parts of the header must be copied by all fragments.
- **Only data in a datagram is fragmented.**

Fields Related to Fragmentation

- The fields that are related to fragmentation and reassembly of an IP datagram are the identification, flags, and fragmentation offset fields.

❑ **Identification.**

- This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IP address must uniquely define a datagram as it leaves the source host.

- all fragments have the same identification number, which is also the same as the original datagram.
- The identification number helps the destination in reassembling the datagram.
- It knows that all fragments having the same identification value should be assembled into one datagram.

- **Flags.**
  - This is a three-bit field. The first bit is reserved (not used). The second bit is called the *do not fragment bit*. *If its value is 1, the machine must not fragment the* datagram.
  - If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host .
  - If its value is 0, the datagram can be fragmented if necessary.
  - The third bit is called the *more fragment bit*. *If its value is 1, it means the datagram is not the* last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment

Figure 8-7

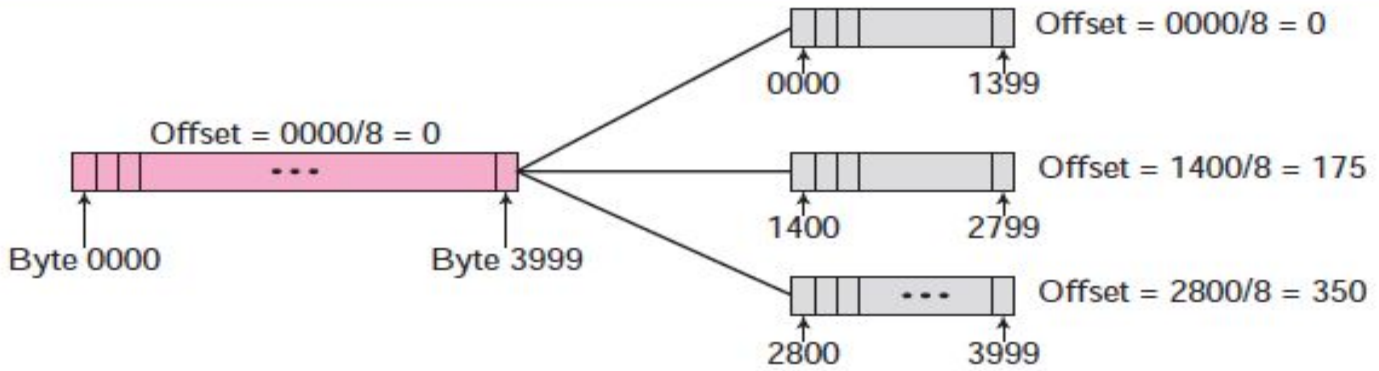
Flag field

D: Do not fragment

M: More fragments

- **Fragmentation offset.** This 13-bit field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes. Figure 7.8 shows a datagram with a data size of 4000 bytes fragmented into three fragments. The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 = 0. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 = 350.

Figure 7.8 *Fragmentation example*



Remember that the value of the offset is measured in units of 8 bytes.

Example 7.5

A packet has arrived with an *M* bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the *M* bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

Example 7.6

A packet has arrived with an *M* bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the *M* bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See also the next example.

Example 7.7

A packet has arrived with an *M* bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?

Solution

Because the *M* bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 7.8

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

8.3

OPTIONS

7.4 OPTIONS

The header of the IP datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options, which can be a maximum of 40 bytes.

Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software. This means that all implementations must be able to handle options if they are present in the header.

Figure 7.10 *Option format*

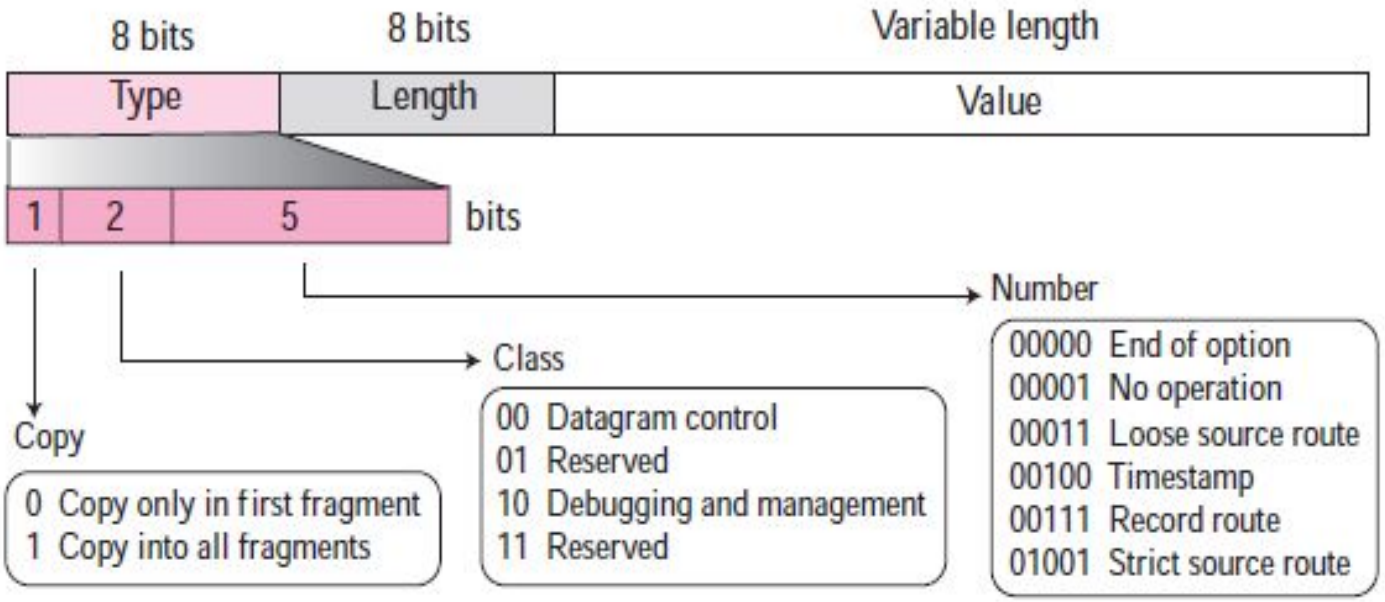
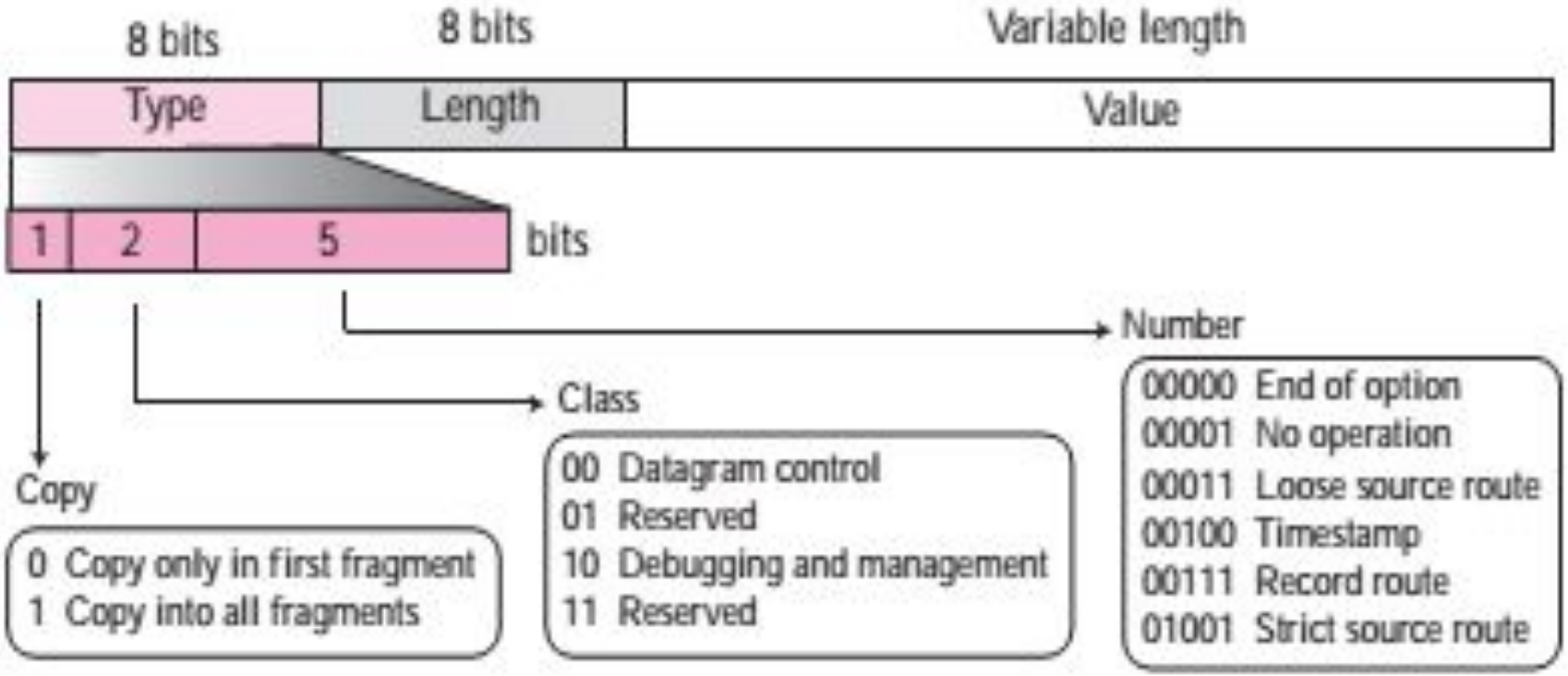


Figure 8-10

Option format



- Copy controls the presence of the option in fragmentation.
- Class defines the general purpose of the option.
- Number defines the type of option

Type

The **type field** is 8 bits long and contains three subfields: copy, class, and number.

- **Copy.** This 1-bit subfield controls the presence of the option in fragmentation. When its value is 0, it means that the option must be copied only to the first fragment. If its value is 1, it means the option must be copied to all fragments.
- **Class.** This 2-bit subfield defines the general purpose of the option. When its value is 00, it means that the option is used for datagram control. When its value is 10, it means that the option is used for debugging and management. The other two possible values (01 and 11) have not yet been defined.
- **Number.** This 5-bit subfield defines the type of option. Although 5 bits can define up to 32 different types, currently only 6 types are in use. These will be discussed in a later section.

Length

The **length field** defines the total length of the option including the type field and the length field itself. This field is not present in all of the option types.

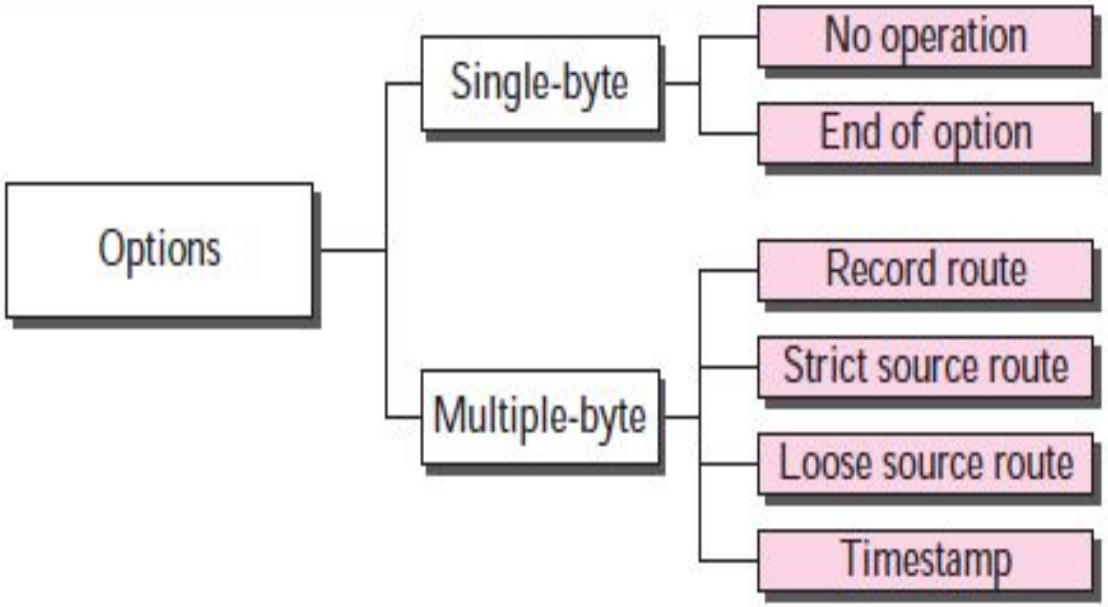
Value

The **value field** contains the data that specific options require. Like the length field, this field is also not present in all option types.

Option Types

As mentioned previously, only six options are currently being used. Two of these are 1-byte options, and they do not require the length or the data fields. Four of them are multiple-byte options; they require the length and the data fields (see Figure 7.11).

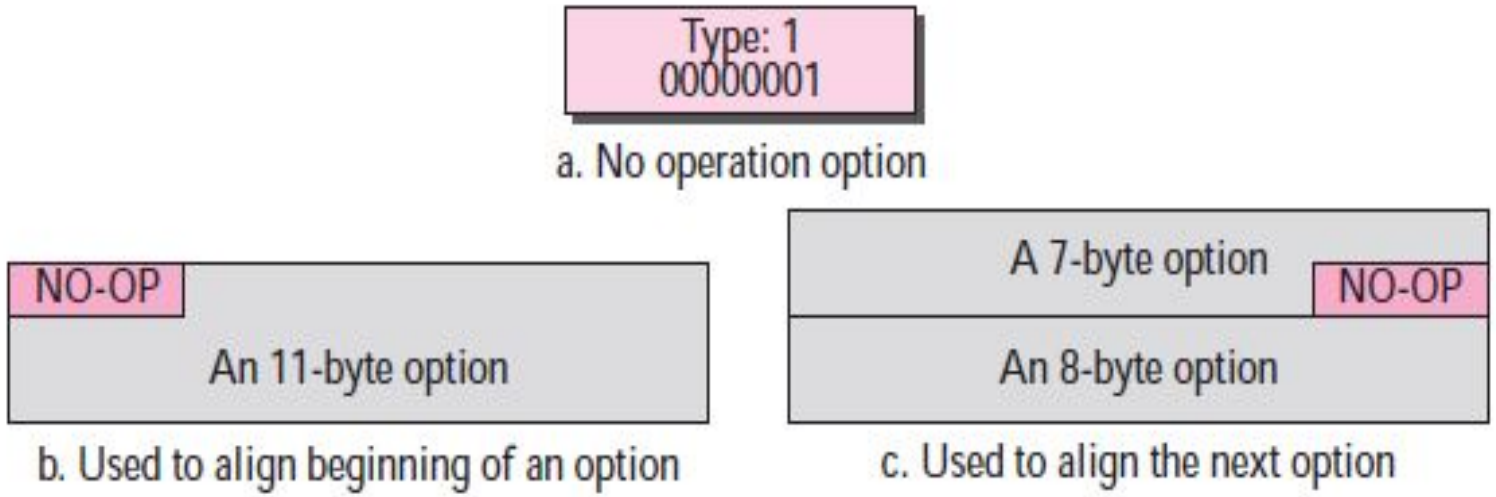
Figure 7.11 *Categories of options*



No-Operation Option

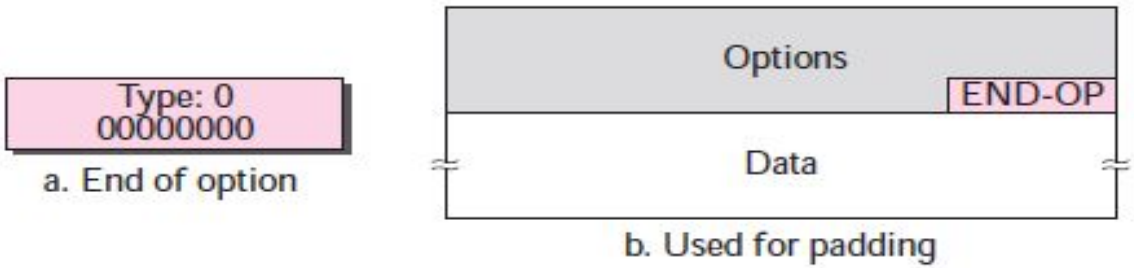
A **no-operation option** is a 1-byte option used as a filler between options. For example, it can be used to align the next option on a 16-bit or 32-bit boundary (see Figure 7.12).

Figure 7.12 *No operation option*



- ***End-of-Option Option***
- **An end-of-option is also a 1-byte option used for padding at the end of the option field**. It, however, can only be used as the last option. Only one end-of-option option can be used. After this option, the receiver looks for the payload .

Figure 7.13 *End-of-option option*



Record-Route Option

A **record-route option** is used to record the Internet routers that handle the datagram.

Figure 7.14 *Record-route option*

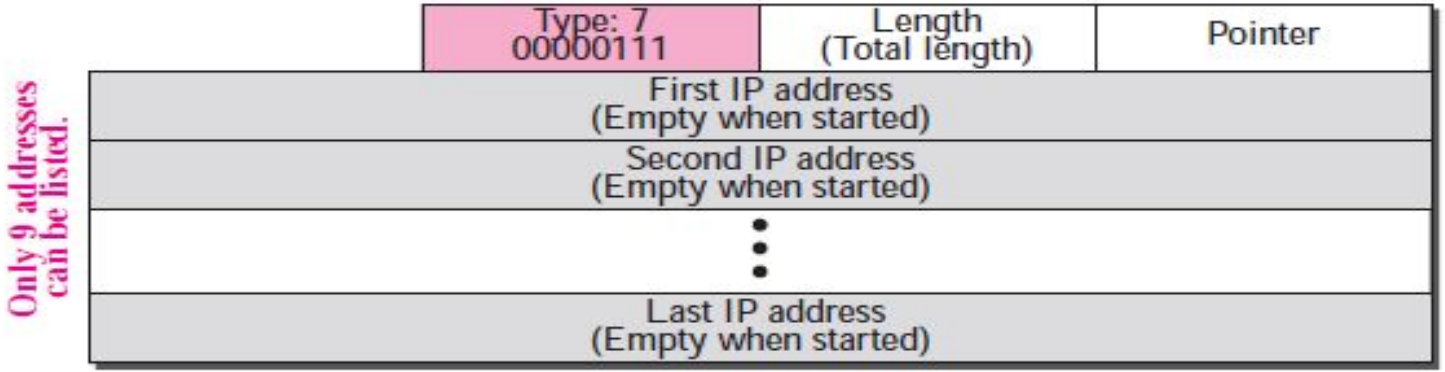


Figure 8-16

Strict source route option

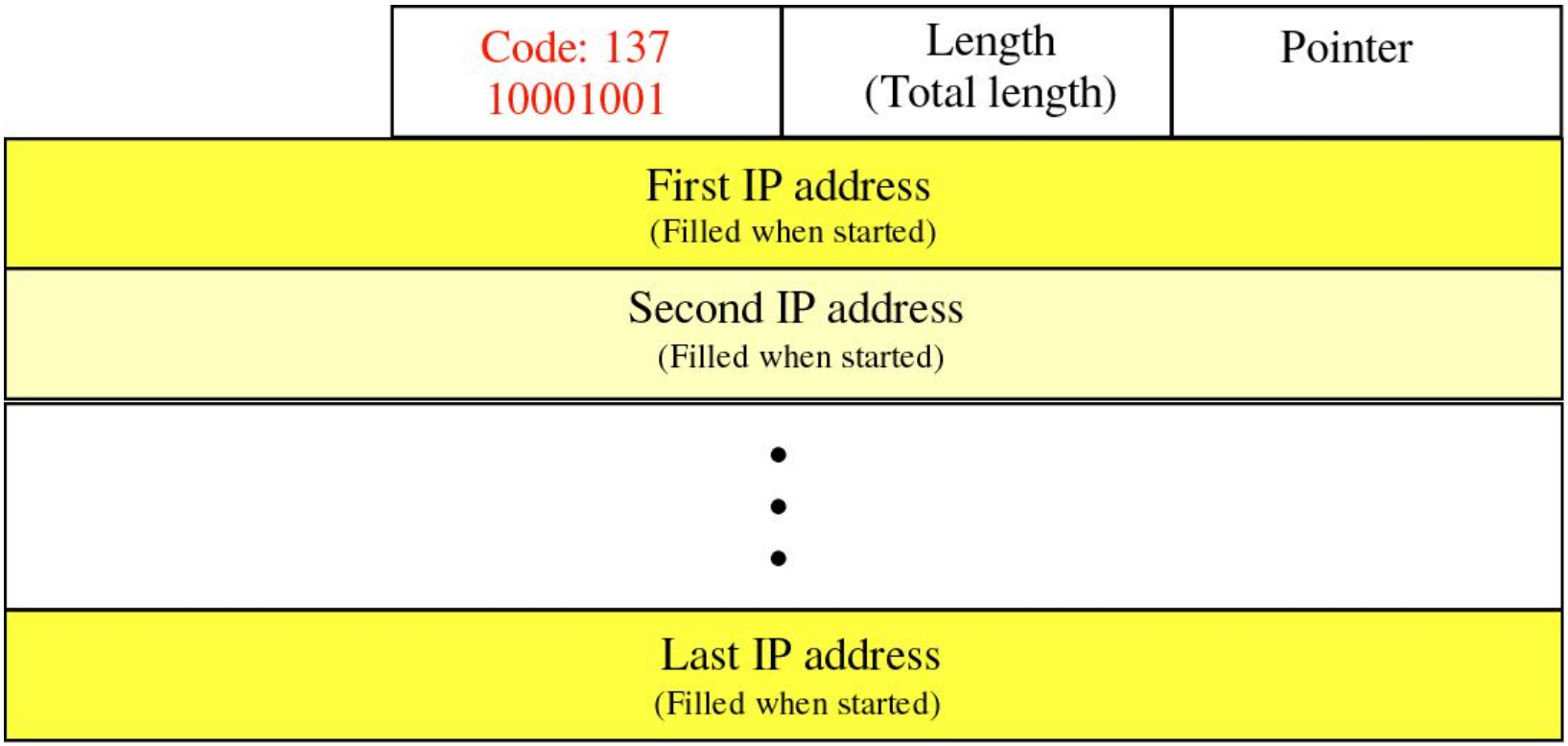


Figure 8-18

Loose source route option

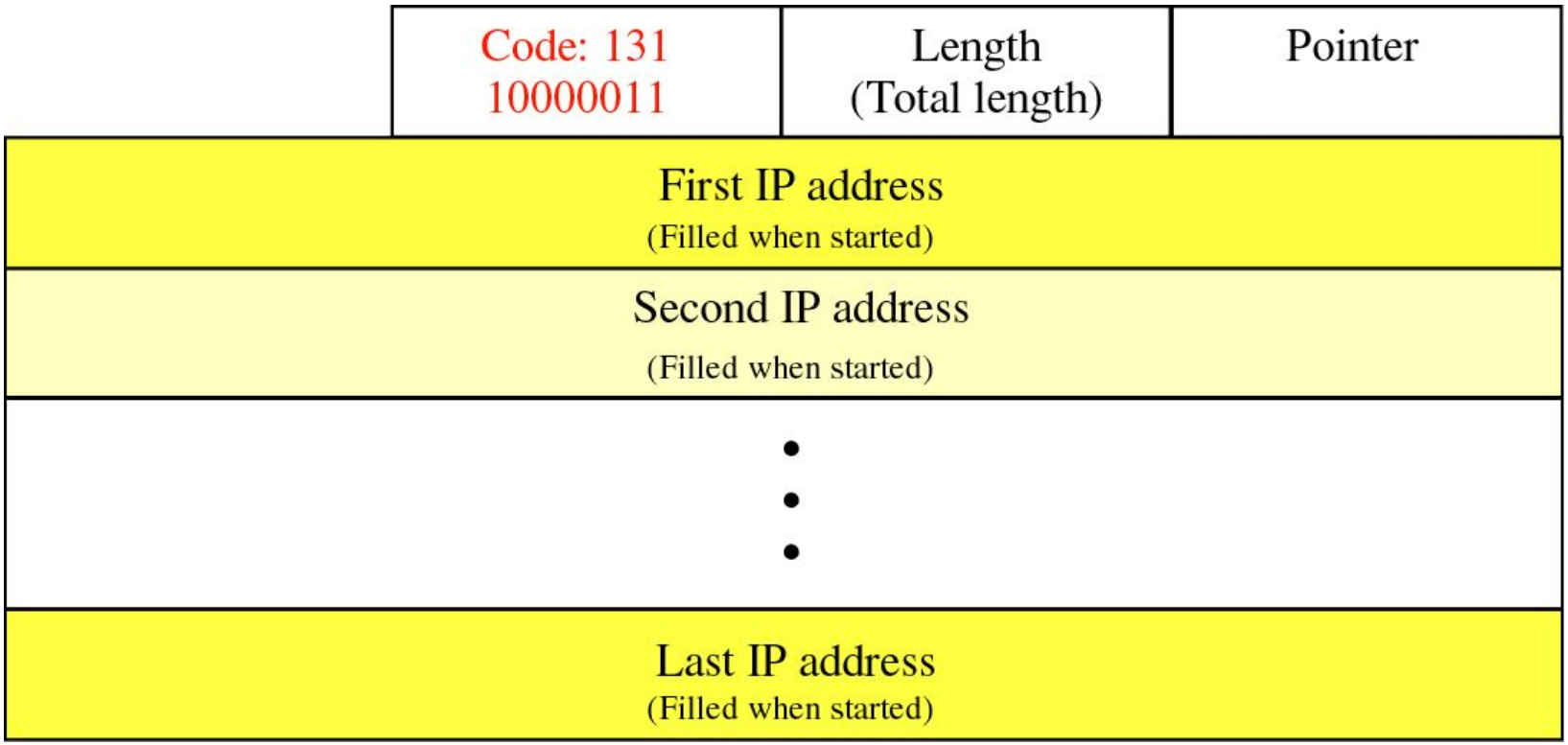
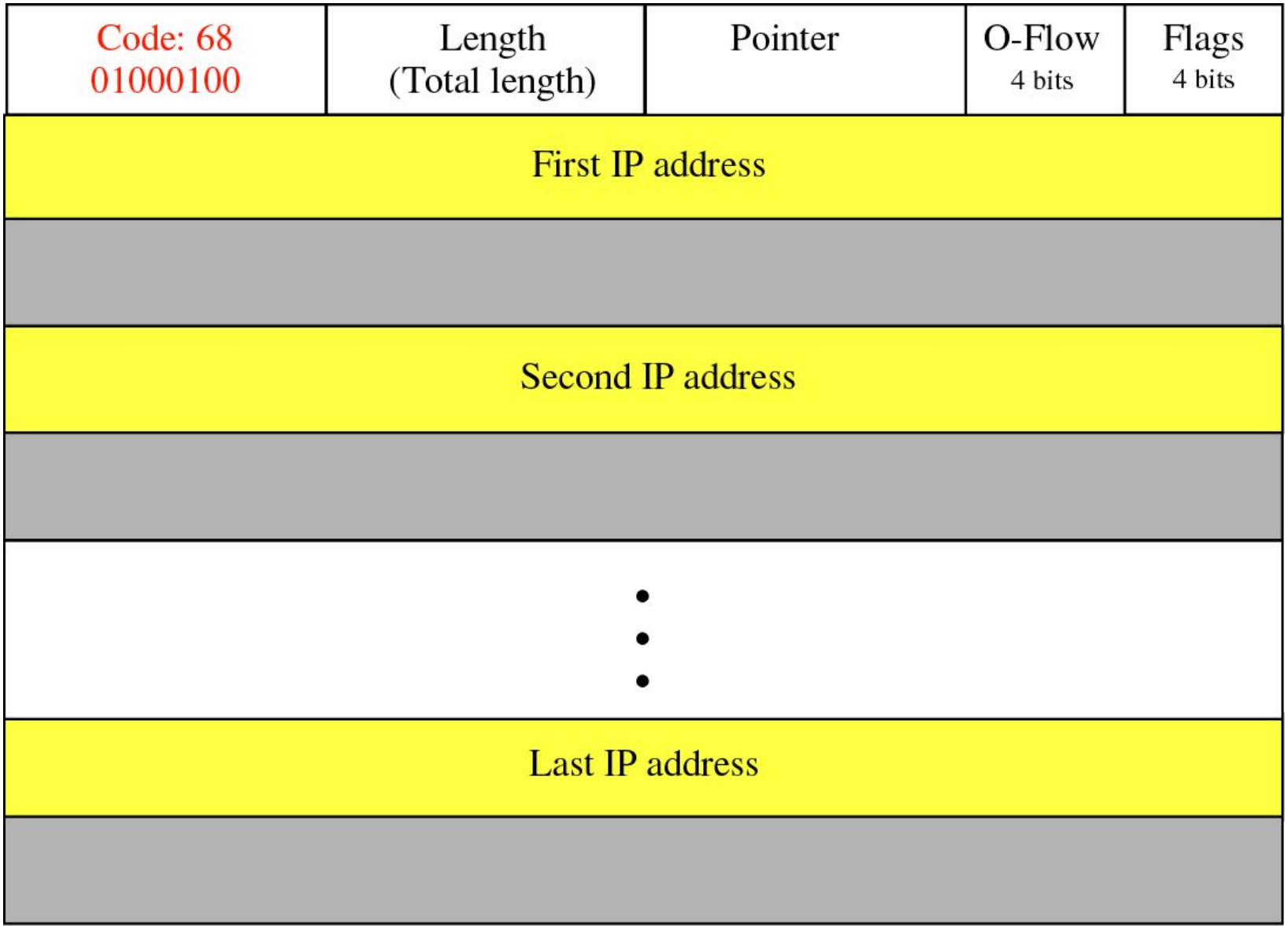


Figure 8-19

Timestamp option



Security issues

Security Issues

There are three security issues that are particularly applicable to the IP protocol: packet sniffing, packet modification, and IP spoofing.

Packet Sniffing

An intruder may intercept an IP packet and make a copy of it. Packet sniffing is a passive attack, in which the attacker does not change the contents of the packet. This type of attack is very difficult to detect because the sender and the receiver may never know that the packet has been copied. Although packet sniffing cannot be stopped, *encryption* of the packet can make the attacker effort useless. The attacker may still sniff the packet, but it cannot find its contents.

Packet Modification

The second type of attack is to modify the packet. The attacker intercepts the packet, changes its contents, and sends the new packet to the receiver. The receiver believes that the packet is coming from the original sender. This type of attack can be detected using a *data integrity* mechanism. The receiver before opening and using the contents of the message can use this mechanism to make sure that the packet has not been changed during the transmission.

- ***IP Spoofing***
- An attacker can masquerade as somebody else and create an IP packet that carries the source address of another computer. An attacker can send an IP packet to a bank pretending that it is coming from one of the customers. This type of attack can be prevented using an *origin authentication mechanism.*

IPSec

The IP packets today can be protected from the previously mentioned attacks using a protocol called IPSec (IP Security). This protocol, which is used in conjunction with the IP protocol, creates a connection-oriented service between two entities in which they can exchange IP packets without worrying about the three attacks discussed before. We will discuss IPSec in detail in Chapter 30, it is enough to mention that IPSec provides the following four services:

Defining Algorithms and Keys

The two entities that want to create a secure channel between themselves can agree on some available algorithms and keys to be used for security purposes.

Packets Encryption

The packets exchanged between two parties can be encrypted for privacy using one of the encryption algorithms and a shared key agreed upon in the first step. This makes the packet sniffing attack useless.

Data Integrity

Data integrity guarantees that the packet is not modified during the transmission. If the received packet does not pass the data integrity test, it is discarded. This prevents the second attack, packet modification, described above.

Origin Authentication

IPsec can authenticate the origin of the packet to be sure that the packet is not created by an imposter. This can prevent IP spoofing attack as described above.