Chapter 7

Internet Protocol Version4 (IPv4)

TCP/IP
Protocol Suite

Fourth Edition

Behrouz A. Forouzan
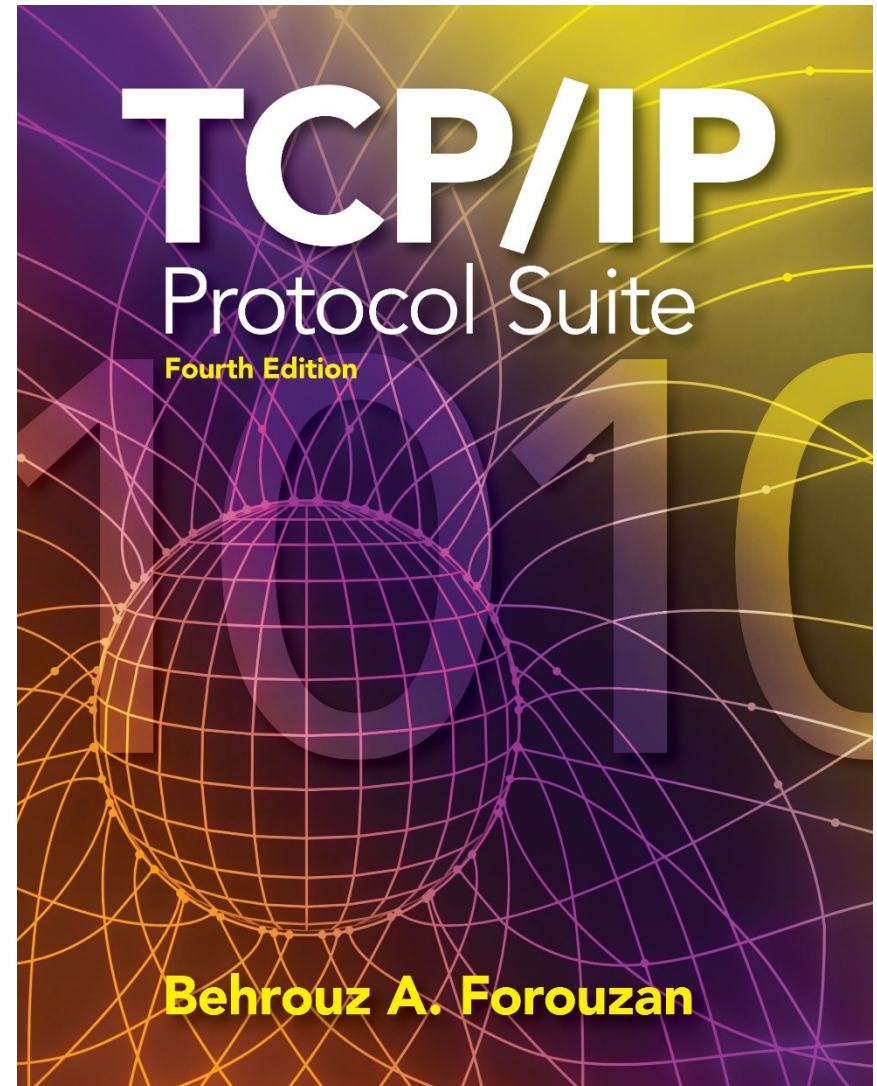
OBJECTIVES:

❑ **To explain the general idea behind the IP protocol and the position of IP in TCP/IP protocol suite.**

❑ **To show the general format of an IPv4 datagram.**

❑ **To discuss fragmentation and reassembly of datagrams.**

❑ **To discuss several options that can be in an IPv4 datagram and their applications.**

❑ **To show how a checksum is calculated for the header of an IPv4 datagram at the sending site and how the checksum is checked at the receiver site.**

❑ **To discuss IP over ATM and compare it with IP over LANs and/or point-to-point WANs.**

❑ **To show a simplified version of the IP package and give the pseudocode for some modules.**
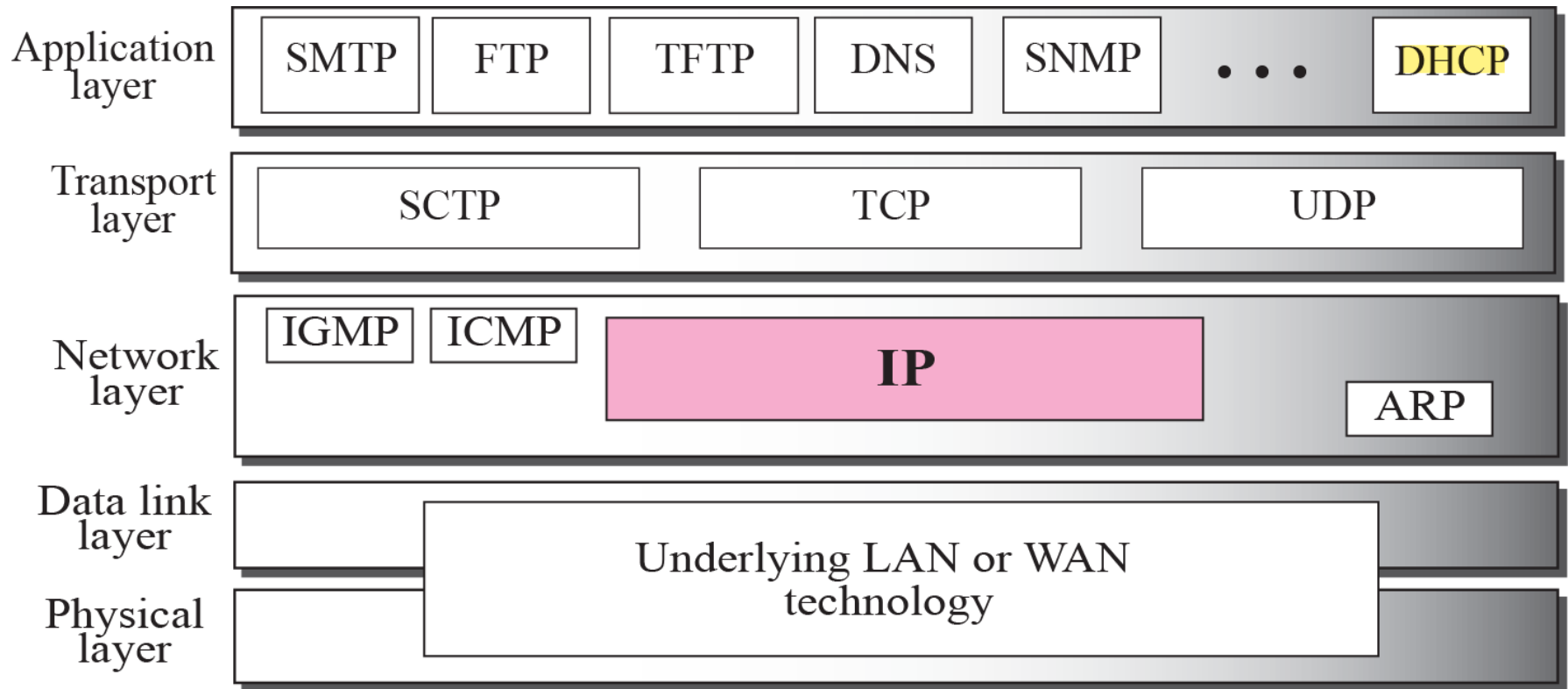
Chapter Outline

7-1 INTRODUCTION

The Internet Protocol (IP) is the transmission mechanism used by the TCP/IP protocols at the network layer.

Topics Discussed in the Section

✓ **Relationship of IP to the rest of the TCP/IP Suite**
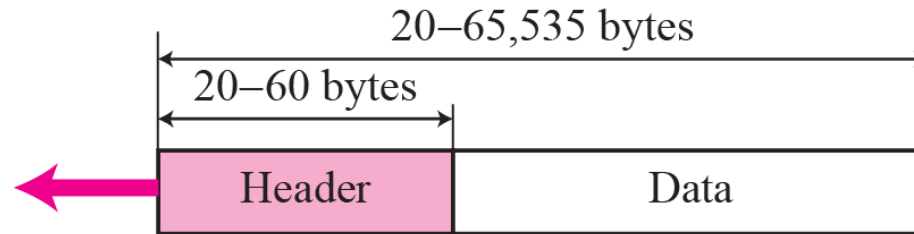
Figure 7.1   Position of IP in TCP/IP protocol suite

7-2   DATAGRAMS

Packets in the network (internet) layer are called *datagrams*. A datagram is a variable-length packet consisting of two parts: header and data. The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections. A brief description of each field is in order.
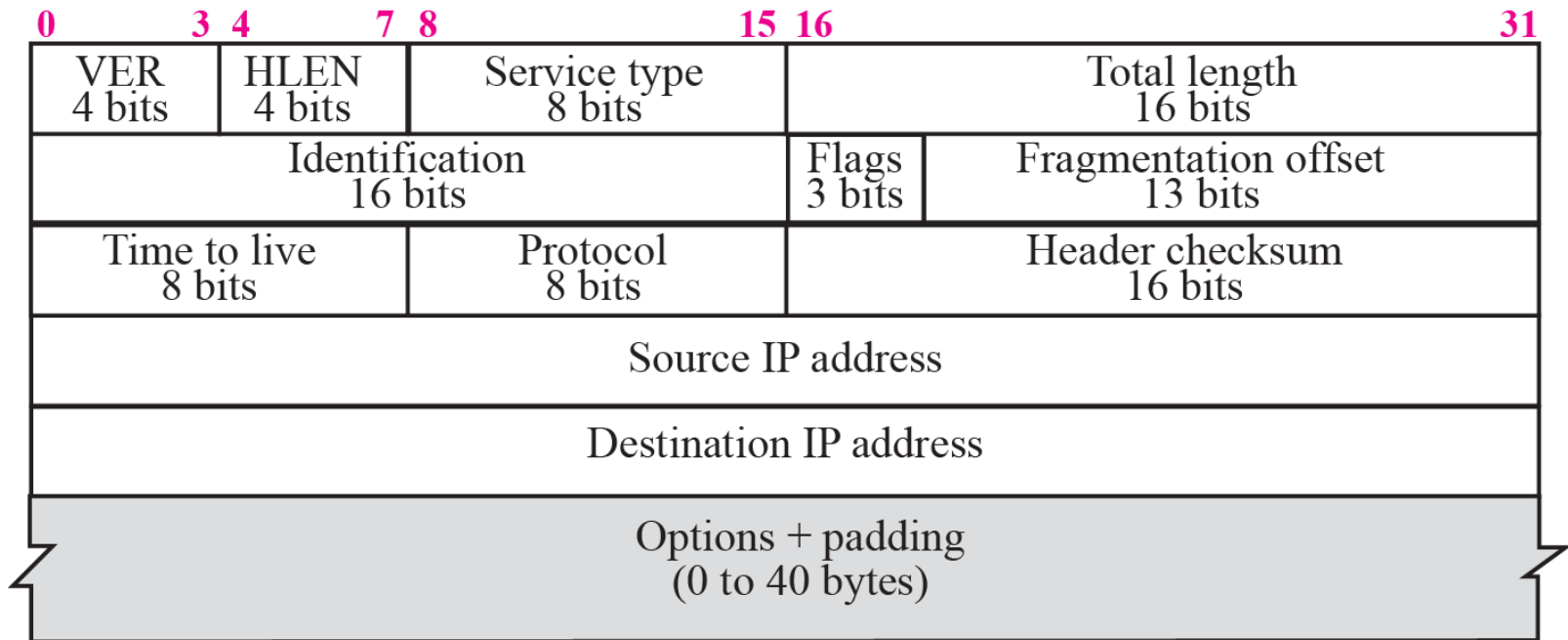
Topics Discussed in the Section

✓ **Format of the datagram packet**

✓ **Some examples**

Figure 7.2   IP datagram

a. IP datagram

VER 4 bits	HLEN 4 bits	Service type 8 bits		Total length 16 bits	
Identification 16 bits			Flags 3 bits	Fragmentation offset 13 bits	
Time to live 8 bits		Protocol 8 bits		Header checksum 16 bits	
Source IP address					
Destination IP address					
Options + padding (0 to 40 bytes)					

b. Header format

Figure 7.3 *Service type*

6 bits | 2 bits

Codepoint | Unused

| X | X | X | **0** | **0** | **0** |

Precedence
interpretation

X	X	X	X	X	**0**
X	X	X	X	**1**	**1**
X	X	X	X	**0**	1

Differential service
interpretation
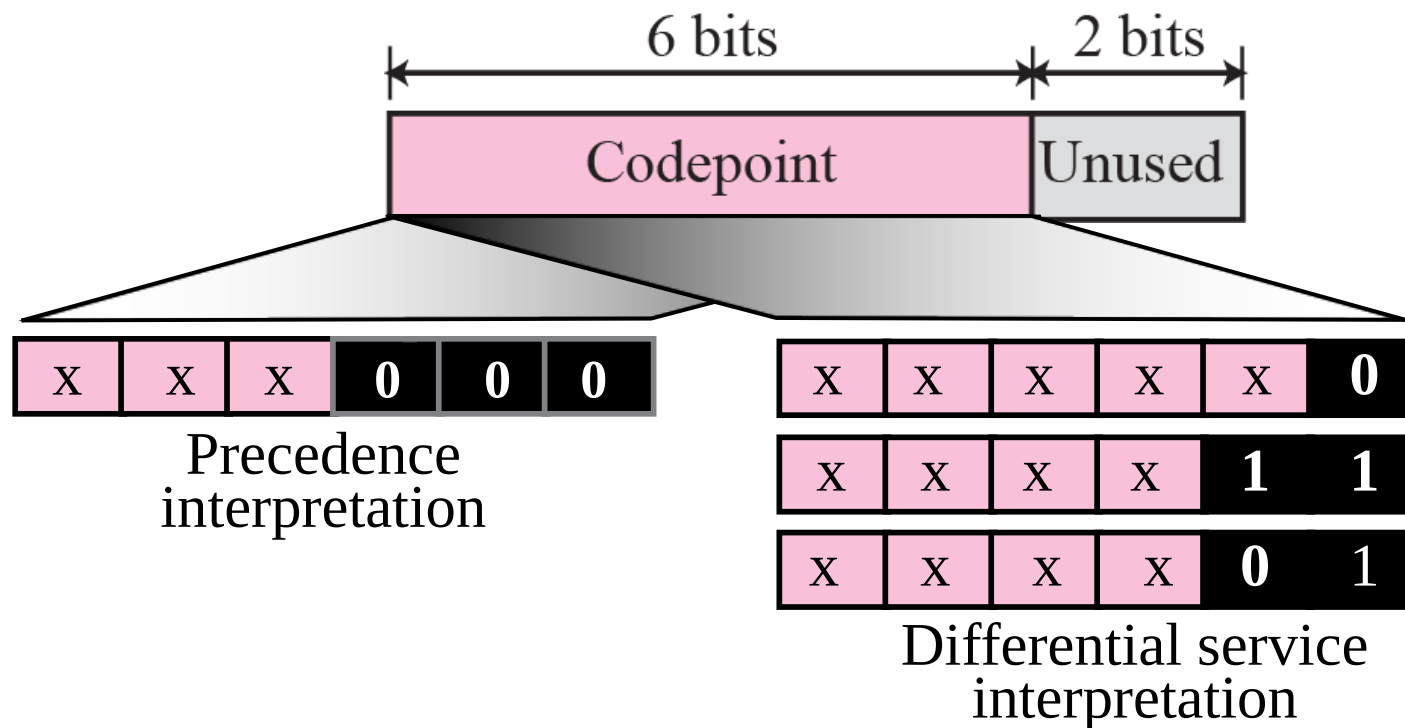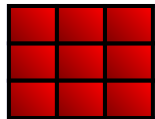
Table 7.1 *Values for codepoints*

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

Note

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

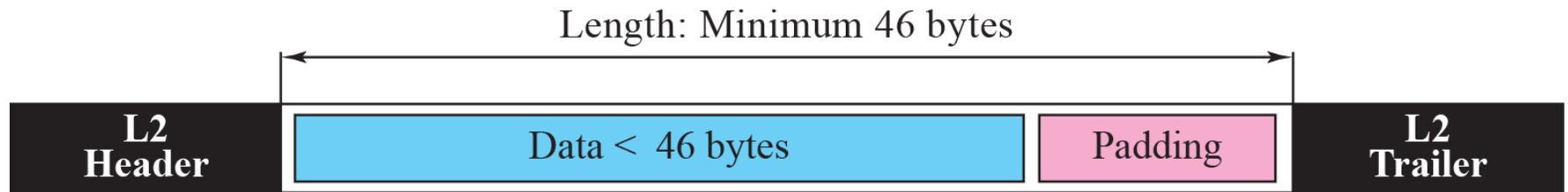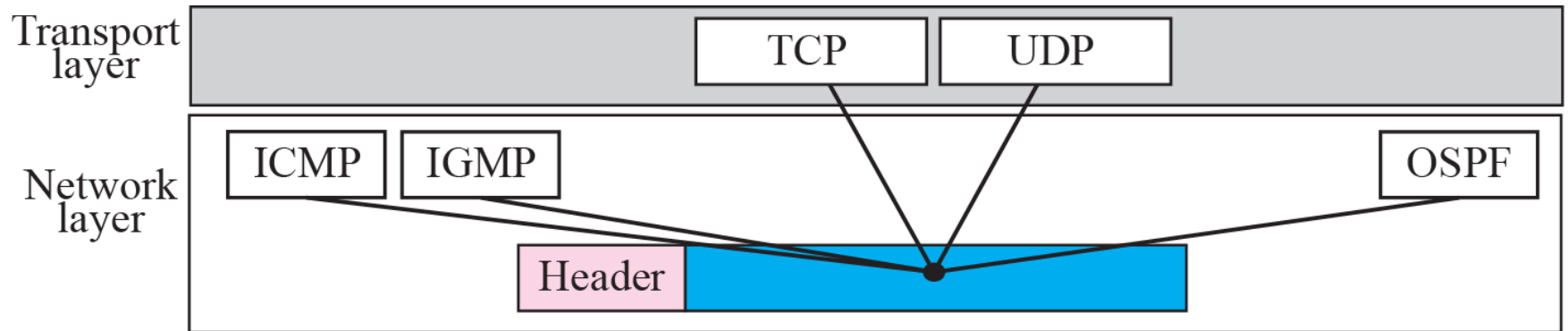# Figure 7.4   *Encapsulation of a small datagram in an Ethernet frame*
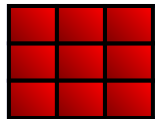
Figure 7.5   *Multiplexing*

Table 7.2 *Protocols*

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

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.

Example 7.3

In an IP packet, the value of HLEN is 5_{16} and the value of the total length field is 0028_{16}. How many bytes of data are being carried by this packet?

Solution

The HLEN value is 5, which means the total number of bytes in the header is 5 × 4 or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40 − 20).

Example 7.4

An IP packet has arrived with the first few hexadecimal digits as shown below:

45000028000100000102 . . .

How many hops can this packet travel before being dropped? The data belong to what upper layer protocol?

Solution

To find the time-to-live field, we skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper layer protocol is IGMP (see Table 7.2)
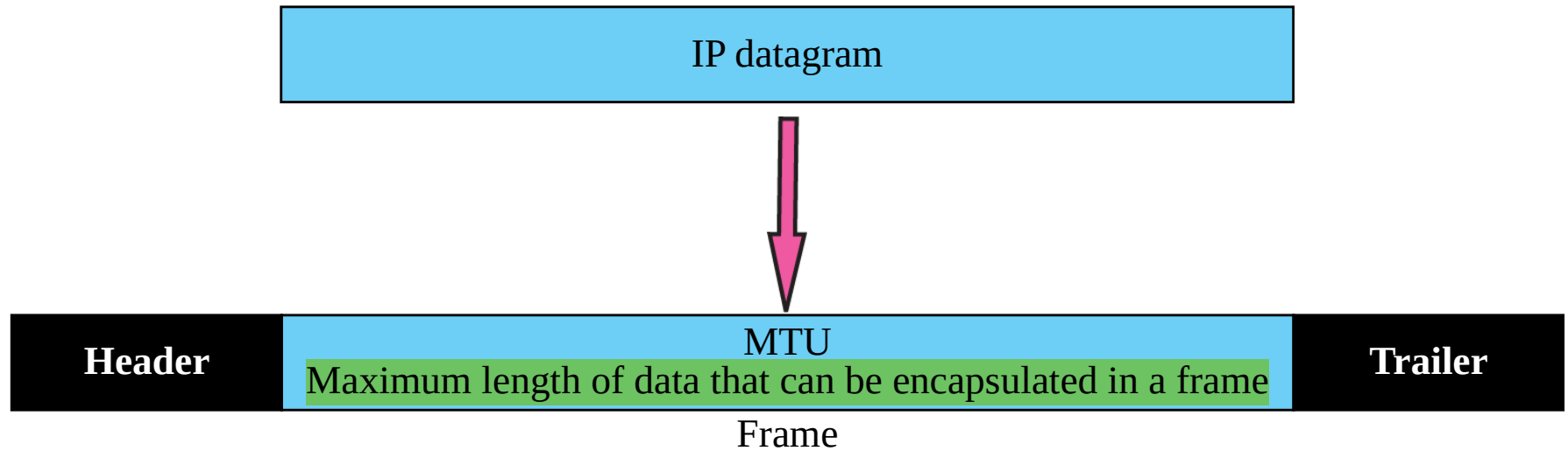
7-3 FRAGMENTATION

A datagram can travel through different networks. Each router decapsulates the IP datagram from the frame it receives, processes it, and then encapsulates it in another frame. The **format and size** of the received frame depend on the **protocol used by the physical network** through which the frame has just traveled. The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel.

Topics Discussed in the Section

✓ **Maximum Transfer Unit (MTU)**

✓ **Fields Related to Fragmentation**

Figure 7.6 *MTU*

IP datagram

| Header | MTU
Maximum length of data that can be encapsulated in a frame | Trailer |

Frame

Only data in a datagram is fragmented.

Figure 7.7   *Flags field*

D: Do not fragment
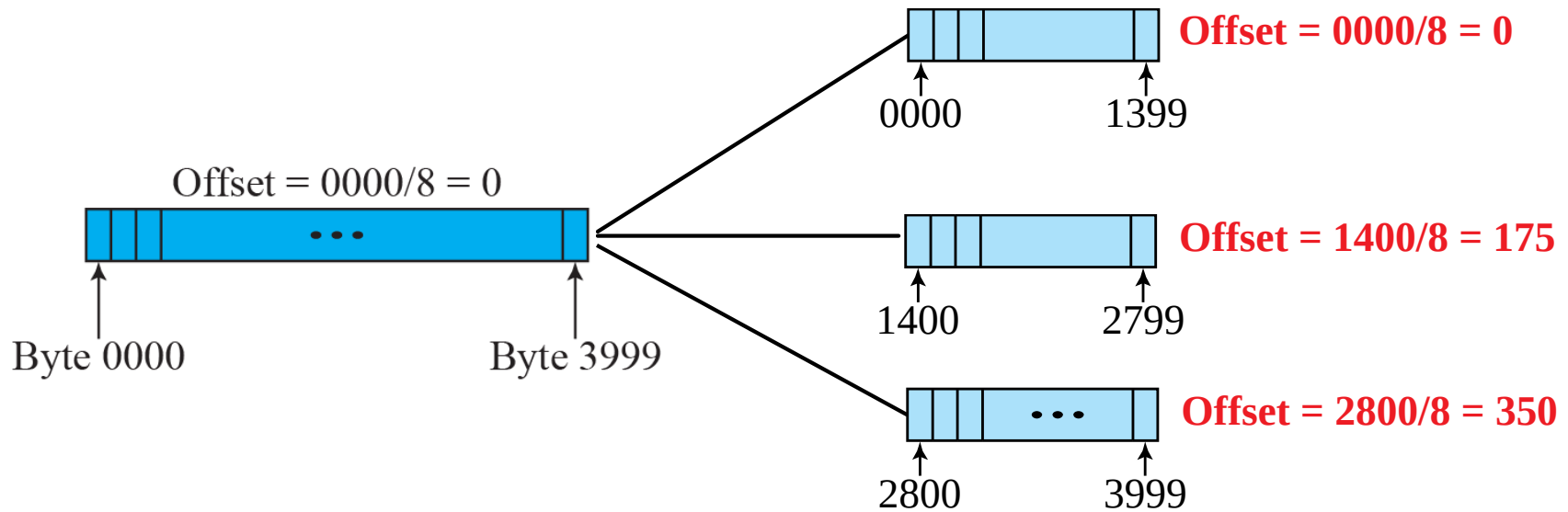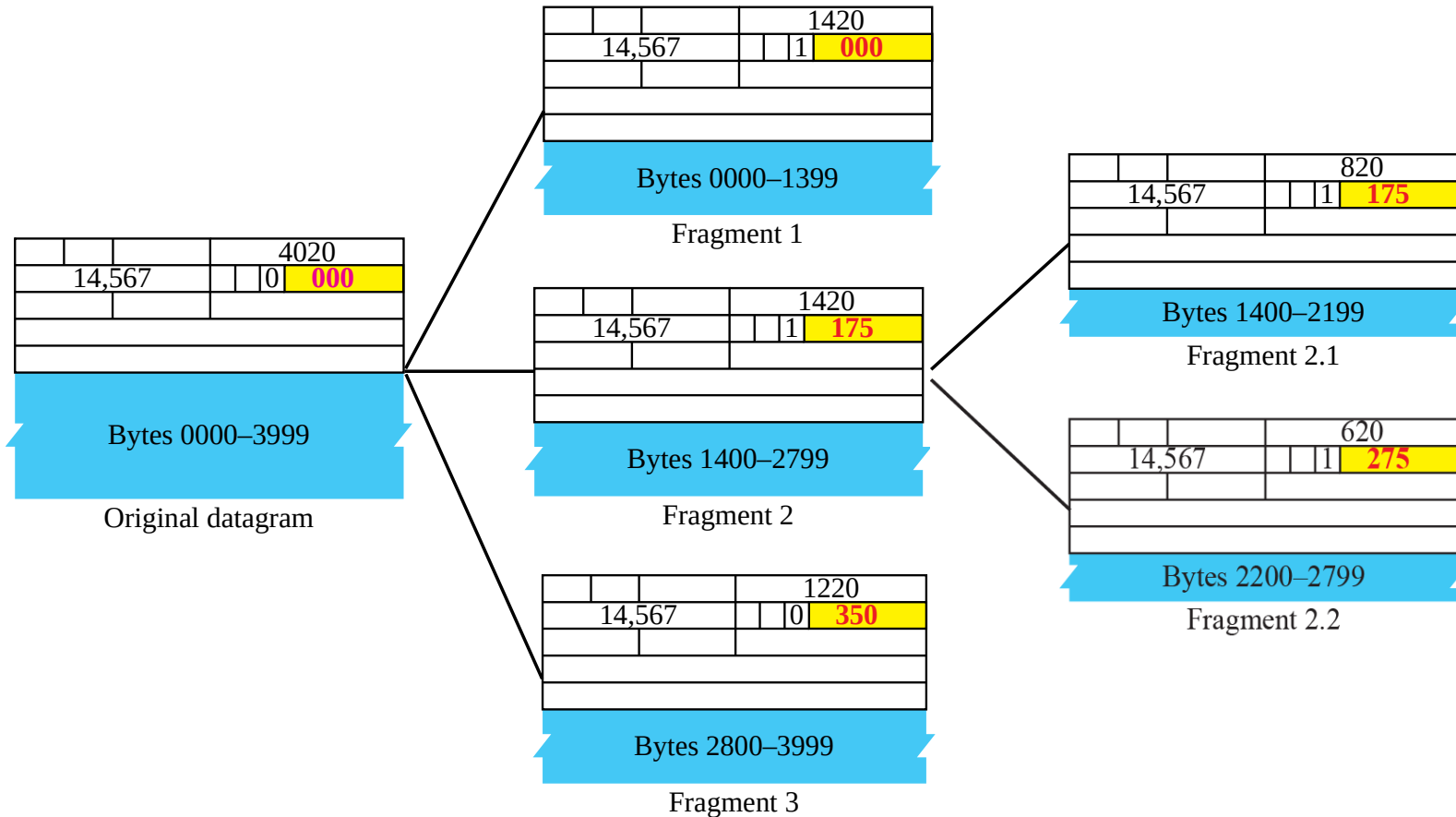M: More fragments

Figure 7.8   *Fragmentation example*

Offset = 0000/8 = 0

0000   1399

Offset = 1400/8 = 175

1400   2799

Offset = 2800/8 = 350

2800   3999

Offset = 0000/8 = 0

Byte 0000   Byte 3999

Figure 7.9   *Detailed fragmentation example*

Fragment 1 — 14,567, 1, 000, 1420, Bytes 0000–1399

Original datagram — 14,567, 0, 000, 4020, Bytes 0000–3999

Fragment 2 — 14,567, 1, 175, 1420, Bytes 1400–2799

Fragment 2.1 — 14,567, 1, 175, 820, Bytes 1400–2199

Fragment 2.2 — 14,567, 1, 275, 620, Bytes 2200–2799

Fragment 3 — 14,567, 0, 350, 1220, Bytes 2800–3999

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.

Example 7.9

A packet has arrived in which the offset value is 100, the value of HLEN is 5 and the value of the total length field is 100. What is the number of the first byte and the last byte?

Solution

The first byte number is $100 \times 8 = 800$. The total length is 100 bytes and the header length is 20 bytes (5×4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

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.**

Topics Discussed in the Section

- ✓ **Format**
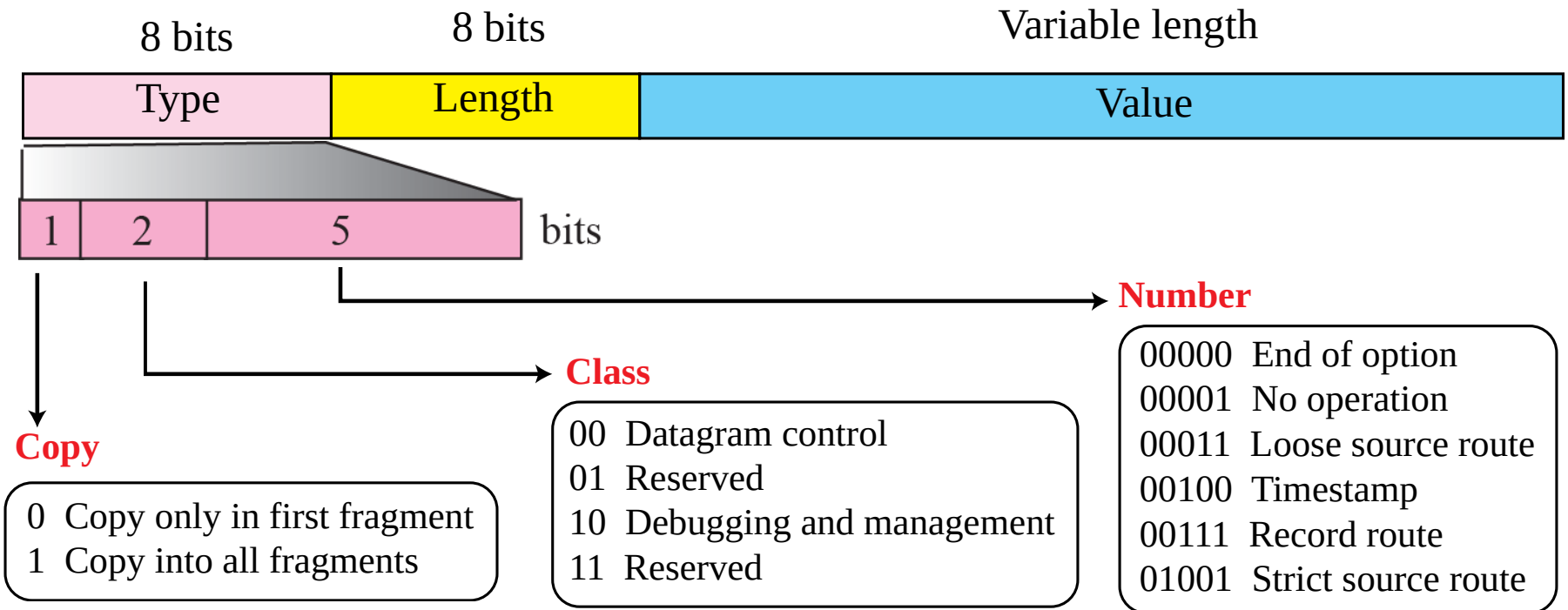- ✓ **Option Types**

Figure 7.10   *Option format*

8 bits   8 bits   Variable length

Type	Length	Value

1	2	5	bits

Number

00000 End of option
00001 No operation
00011 Loose source route
00100 Timestamp
00111 Record route
01001 Strict source route

Class

00 Datagram control
01 Reserved
10 Debugging and management
11 Reserved

Copy

0 Copy only in first fragment
1 Copy into all fragments

Figure 7.11 *Categories of options*
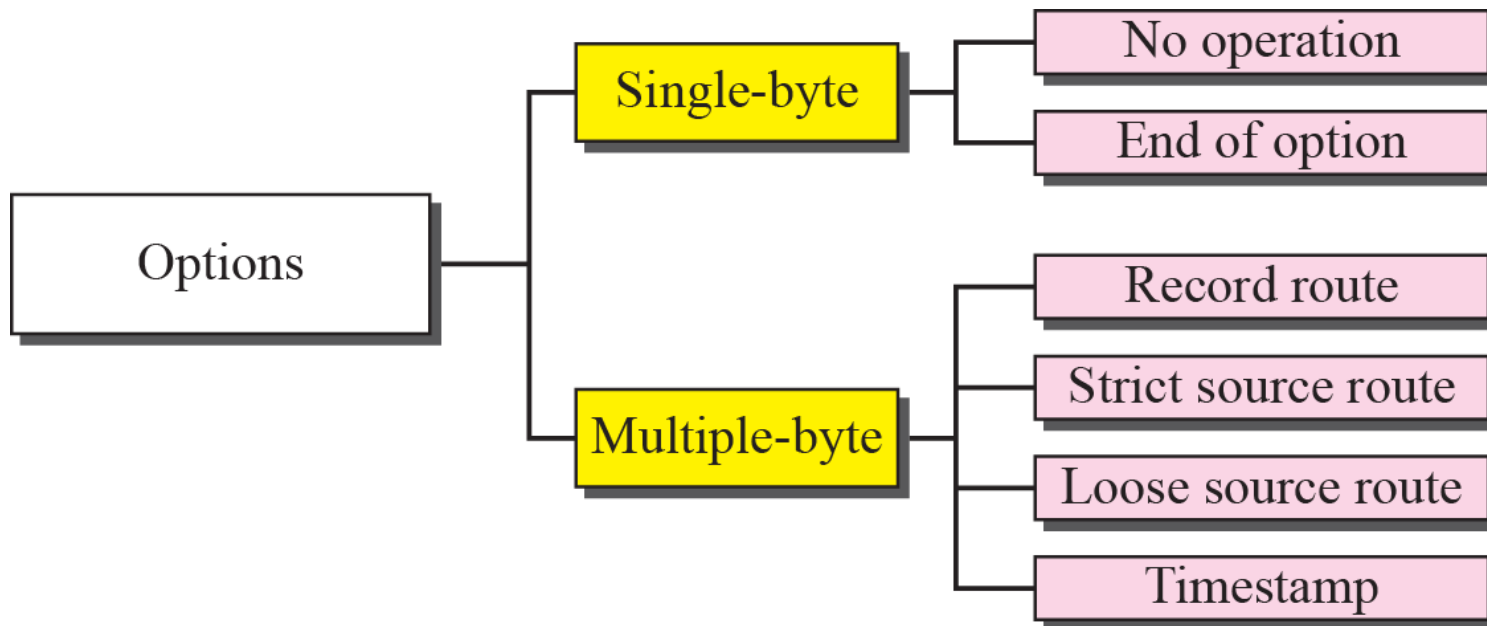
Figure 7.12 *No operation option*

Type: 1
00000001
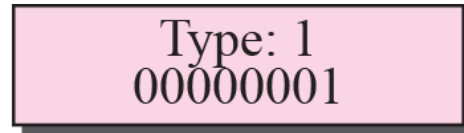
a. No operation option

NO-OP
An 11-byte option
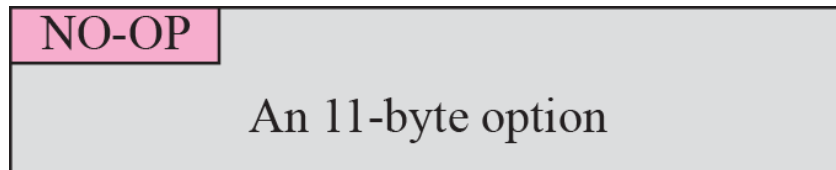
b. Used to align beginning of an option

A 7-byte option
NO-OP
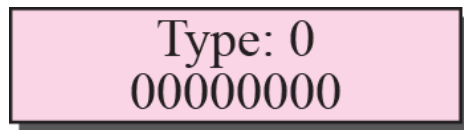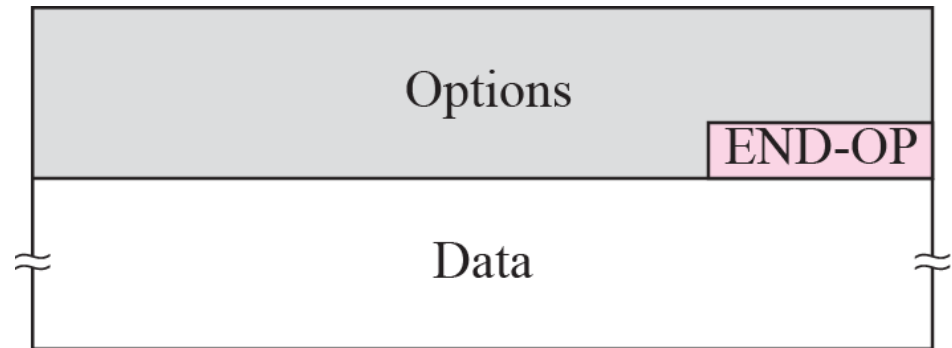An 8-byte option
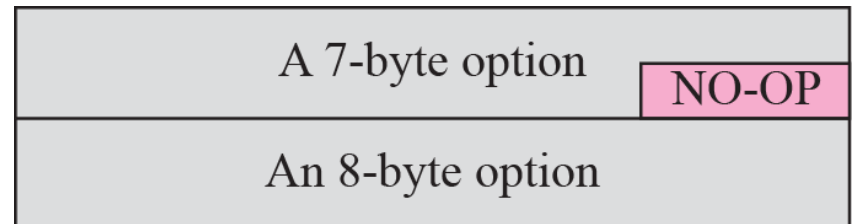
c. Used to align the next option

Figure 7.13 *Endo-of-option option*

a. End of option

b. Used for padding

Figure 7.14 *Record-route option*

Type: 7 00000111	Length (Total length)	Pointer
First IP address (Empty when started)		
Second IP address (Empty when started)		
• • •		
Last IP address (Empty when started)		

Only 9 addresses can be listed.

Figure 7.15   Record-route concept
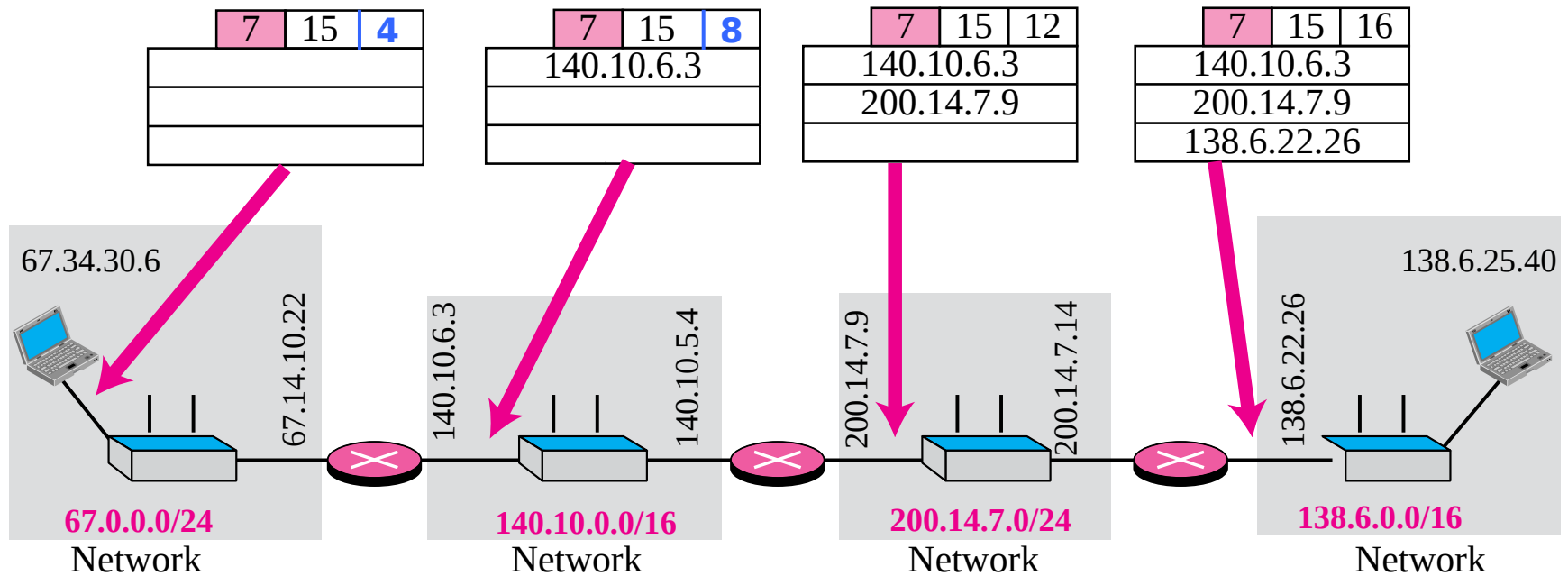
Figure 7.16 *Strict-source-route option*

Type: 137 10001001	Length (Total length)	Pointer

Only 9 addresses can be listed.

First IP address (Filled when started)
Second IP address (Filled when started)
• • •
Last IP address (Filled when started)

Figure 7.17 *Strict-source-route option*
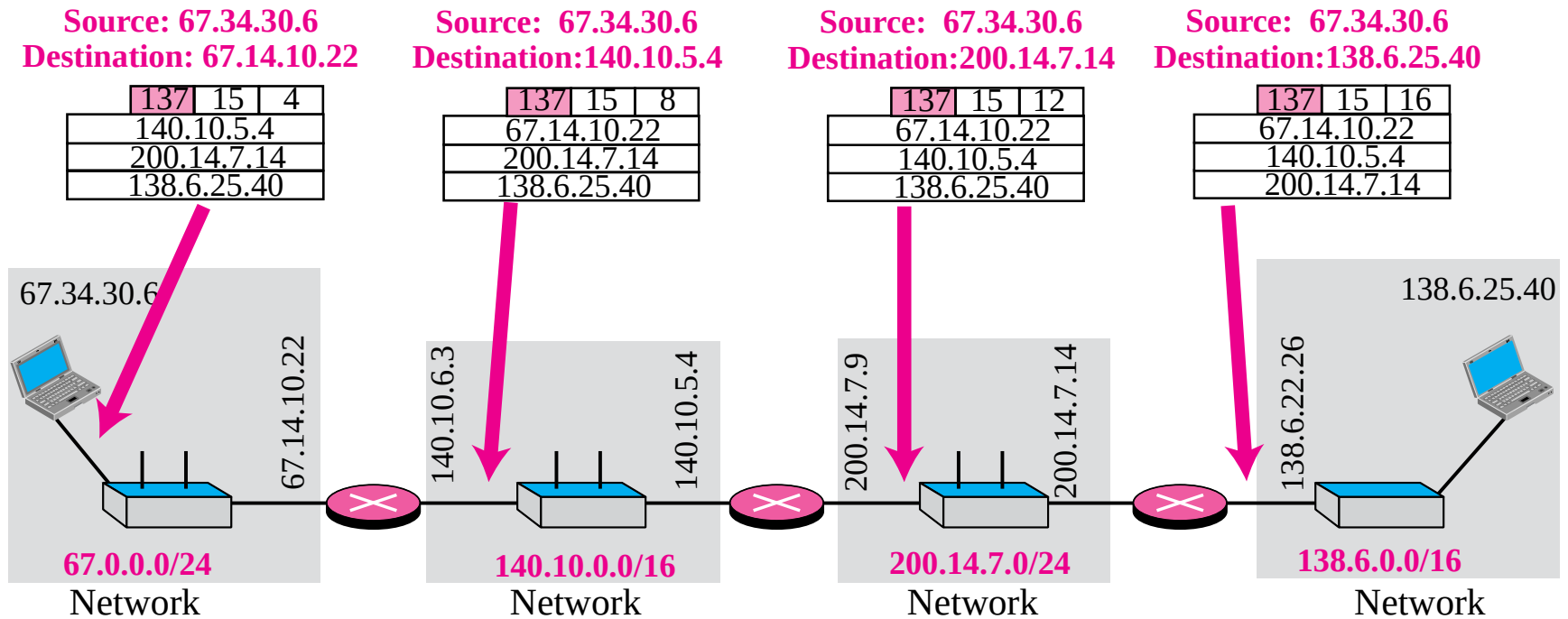
Figure 7.18 *Loose-source-route option*

Type: 131 10000011	Length (Total length)	Pointer
First IP address (Filled when started)		
Second IP address (Filled when started)		
• • •		
Last IP address (Filled when started)		

Only 9 addresses can be listed.

Figure 7.19 *Time-stamp option*

Code: 68 01000100	Length (Total length)	Pointer	O-Flow 4 bits	Flags 4 bits
First IP address				
Second IP address				
• • •				
Last IP address				

Figure 7.20 *Use of flags in timestamp*

Enter timestamps
only

Flag: 0

Enter IP addresses
and timestamps

Flag: 1

IP addresses given,
enter timestamps

3

140.10.6.3

200.14.7.9

138.6.22.26

Flag: 3

Figure 7.21 *Timestamp concept*
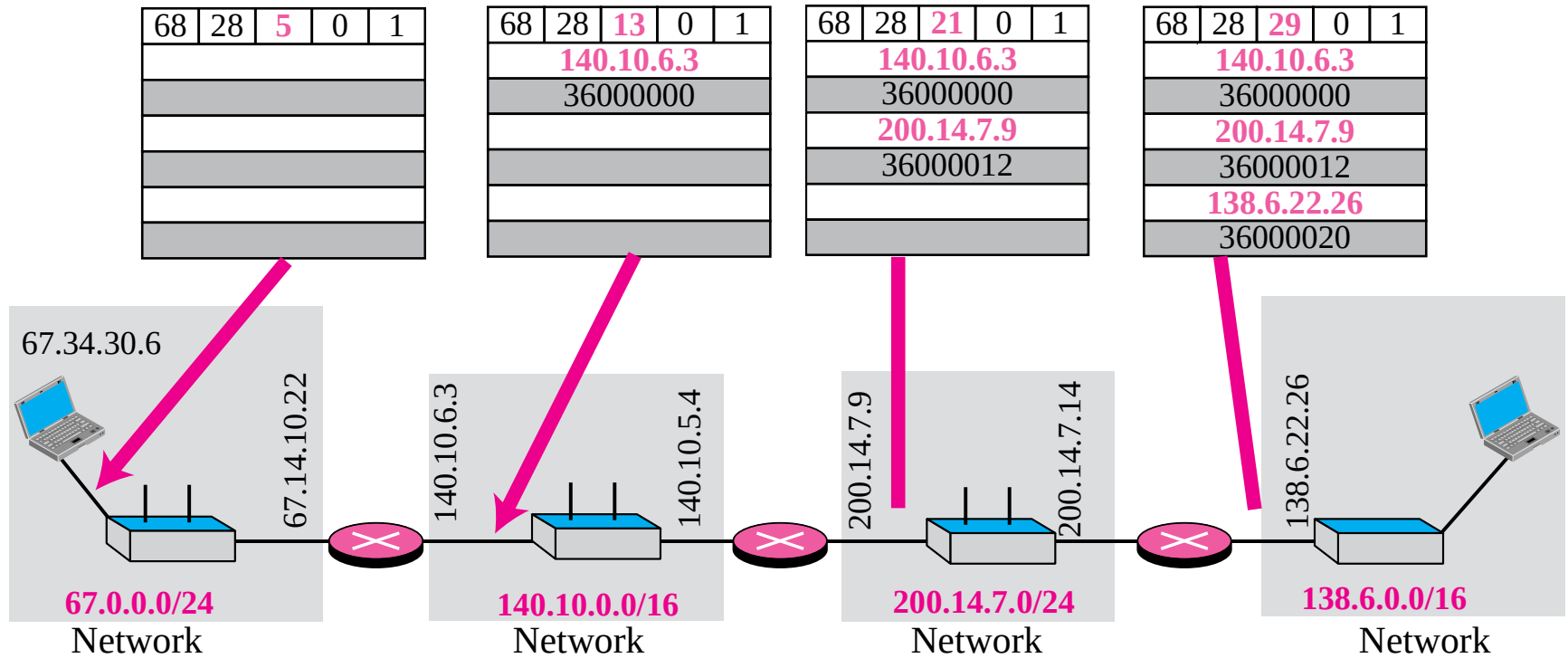
Example 7.10

Which of the six options must be copied to each fragment?

Solution

We look at the first (left-most) bit of the type for each option.

a. No operation: type is 00000001; not copied.
b. End of option: type is 00000000; not copied.
c. Record route: type is 00000111; not copied.
d. Strict source route: type is 10001001; copied.
e. Loose source route: type is 10000011; copied.
f. Timestamp: type is 01000100; not copied.

Example 7.11

Which of the six options are used for datagram control and which for debugging and managements?

Solution

We look at the second and third (left-most) bits of the type.

a. No operation: type is 00000001; datagram control.

b. End of option: type is 00000000; datagram control.

c. Record route: type is 00000111; datagram control.

d. Strict source route: type is 10001001; datagram control.

e. Loose source route: type is 10000011; datagram control.

f. Timestamp: type is 01000100; debugging and management control.

Example 7.12

One of the utilities available in UNIX to check the traveling of the IP packets is **ping**. In the next chapter, we talk about the ping program in more detail. In this example, we want to show how to use the program to see if a host is available. We ping a server at De Anza College named fhda.edu. The result shows that the IP address of the host is 153.18.8.1. The result also shows the number of bytes used.

```
$  ping fhda.edu
PING fhda.edu (153.18.8.1) 56(84) bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq =
0 ttl=62 time=1.87 ms
...
```

Example 7.13

We can also use the ping utility with the **-R option to implement the record route option**. The result shows the interfaces and IP addresses.

```
$ ping -R fhda.edu
PING fhda.edu (153.18.8.1) 56(124) bytes of data.
64 bytes from tiptoe.fhda.edu
(153.18.8.1): icmp_seq=0 ttl=62 time=2.70 ms
RR:   voyager.deanza.fhda.edu (153.18.17.11)
      Dcore_G0_3-69.fhda.edu (153.18.251.3)
      Dbackup_V13.fhda.edu (153.18.191.249)
      tiptoe.fhda.edu (153.18.8.1)
      Dbackup_V62.fhda.edu (153.18.251.34)
      Dcore_G0_1-6.fhda.edu (153.18.31.254)
      voyager.deanza.fhda.edu (153.18.17.11)
```

Example 7.14

The **traceroute** utility can also be used to **keep track of the route of a packet.** The result shows the three routers visited.

```
$ traceroute fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 38 byte packets
 1  Dcore_G0_1-6.fhda.edu (153.18.31.254)  0.972 ms  0.902 ms
    0.881 ms
 2  Dbackup_V69.fhda.edu (153.18.251.4)  2.113 ms  1.996 ms
    2.059 ms
 3  tiptoe.fhda.edu (153.18.8.1)  1.791 ms  1.741 ms  1.751 ms
```

Example 7.15

The traceroute program can be used to implement loose source routing. The ==-g option allows us to define the routers to be visited==, from the source to destination. The following shows how we can send a packet to the fhda.edu server with the requirement that the packet visit the router 153.18.251.4.

```
$ traceroute -g  153.18.251.4 fhda.edu.
traceroute to fhda.edu (153.18.8.1), 30 hops max, 46 byte packets
 1   Dcore_G0_1-6.fhda.edu (153.18.31.254)  0.976 ms  0.906 ms
     0.889 ms
 2   Dbackup_V69.fhda.edu (153.18.251.4)  2.168 ms  2.148 ms
     2.037 ms
```

Example 7.16

The traceroute program can also be used to implement strict source routing. The **-G option forces the packet to visit the routers** defined in the command line. The following shows how we can send a packet to the fhda.edu server and force the packet to visit only the router 153.18.251.4.

```
$ traceroute -G  153.18.251.4 fhda.edu.
traceroute to fhda.edu (153.18.8.1), 30 hops max, 46 byte packets
  1  Dbackup_V69.fhda.edu (153.18.251.4)  2.168 ms  2.148 ms
     2.037 ms
```

7-5 CHECKSUM

The error detection method used by most TCP/IP protocols is called the checksum. The checksum protects against the corruption that may occur during the transmission of a packet. It is redundant information added to the packet. The checksum is calculated at the sender and the value obtained is sent with the packet. The receiver repeats the same calculation on the whole packet including the checksum. If the result is satisfactory (see below), the packet is accepted; otherwise, it is rejected.

Topics Discussed in the Section

- ✓ **Checksum Calculation at the Sender**
- ✓ **Checksum Calculation at the Receiver**
- ✓ **Checksum in the Packet**
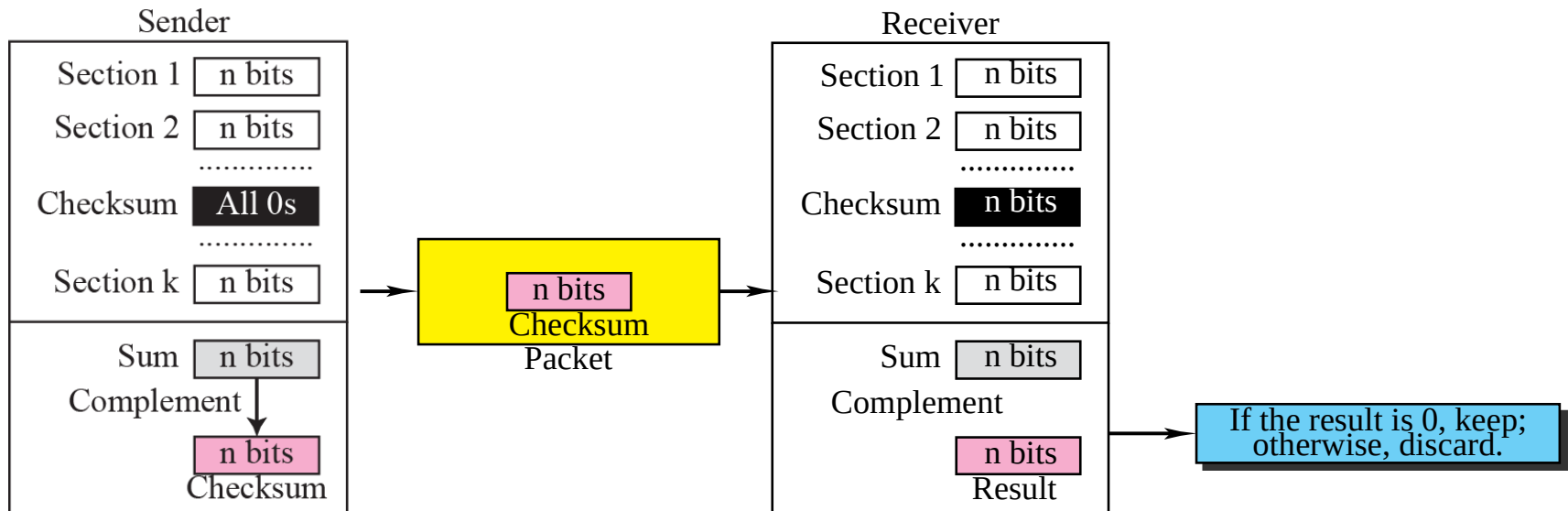
Figure 7.22   *Checksum concept*
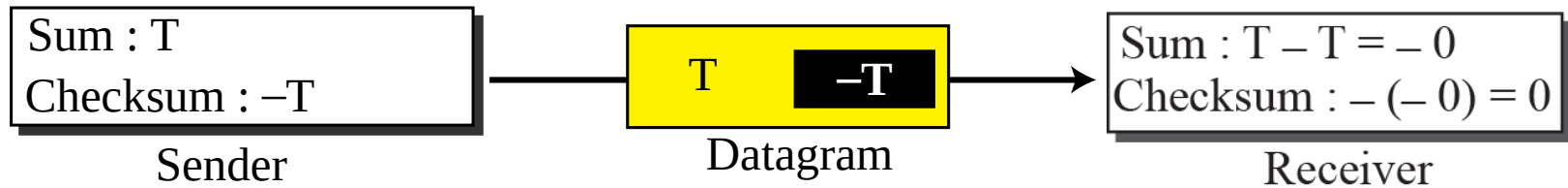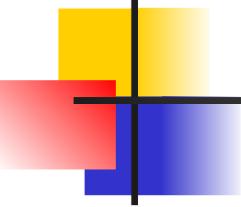
Figure 7.23   *Checksum in one's complement arithmetic*

Sum : T
Checksum : $-$T

Sender

T $-$T

Datagram

Sum : T $-$ T $= -$ 0
Checksum : $-$ ($-$ 0) $=$ 0

Receiver

Note

Checksum in IP covers only the header, not the data.

Example 7.17

Figure 7.24 shows an example of a checksum calculation at the sender site for an IP header without options. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. The result is inserted in the checksum field.

Figure 7.24 *Example of checksum calculation at the sender*

4, 5, and 0	→	01000101	00000000
28	→	00000000	00011100
1	→	00000000	00000001
0 and 0	→	00000000	00000000
4 and 17	→	00000100	00010001
0	→	00000000	00000000
10.12	→	00001010	00001100
14.5	→	00001110	00000101
12.6	→	00001100	00000110
7.9	→	00000111	00001001
Sum	→	**01110100**	**01001110**
Checksum	→	**10001011**	**10110001**

| 5	0		
1		0	
	17		
10.12.14.5			
12.6.7.9			

Example 7.18

Figure 7.25 shows the checking of checksum calculation at the receiver site (or intermediate router) assuming that no errors occurred in the header. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. Since the result is 16 0s, the packet is accepted.

Figure 7.25 *Example of checksum calculation at the receiver*

4	5	0	28
1		0	0
4	17	35761	
10.12.14.5			
12.6.7.9			

```
4, 5,  and 0  ———➤   01000101   00000000
          28  ———➤   00000000   00011100
           1  ———➤   00000000   00000001
     0 and 0  ———➤   00000000   00000000
    4 and 17  ———➤   00000100   00010001
    Checksum  ———➤   10001011   10110001
       10.12  ———➤   00001010   00001100
        14.5  ———➤   00001110   00000101
        12.6  ———➤   00001100   00000110
         7.9  ———➤   00000111   00001001
                    ───────────────────
         Sum  ———➤   1111 1111  1111 1111
    Checksum  ———➤   0000 0000  0000 0000
```
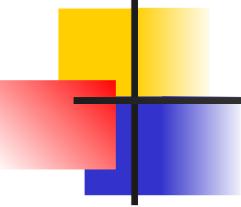
Note

Appendix D gives an algorithm for checksum calculation.