

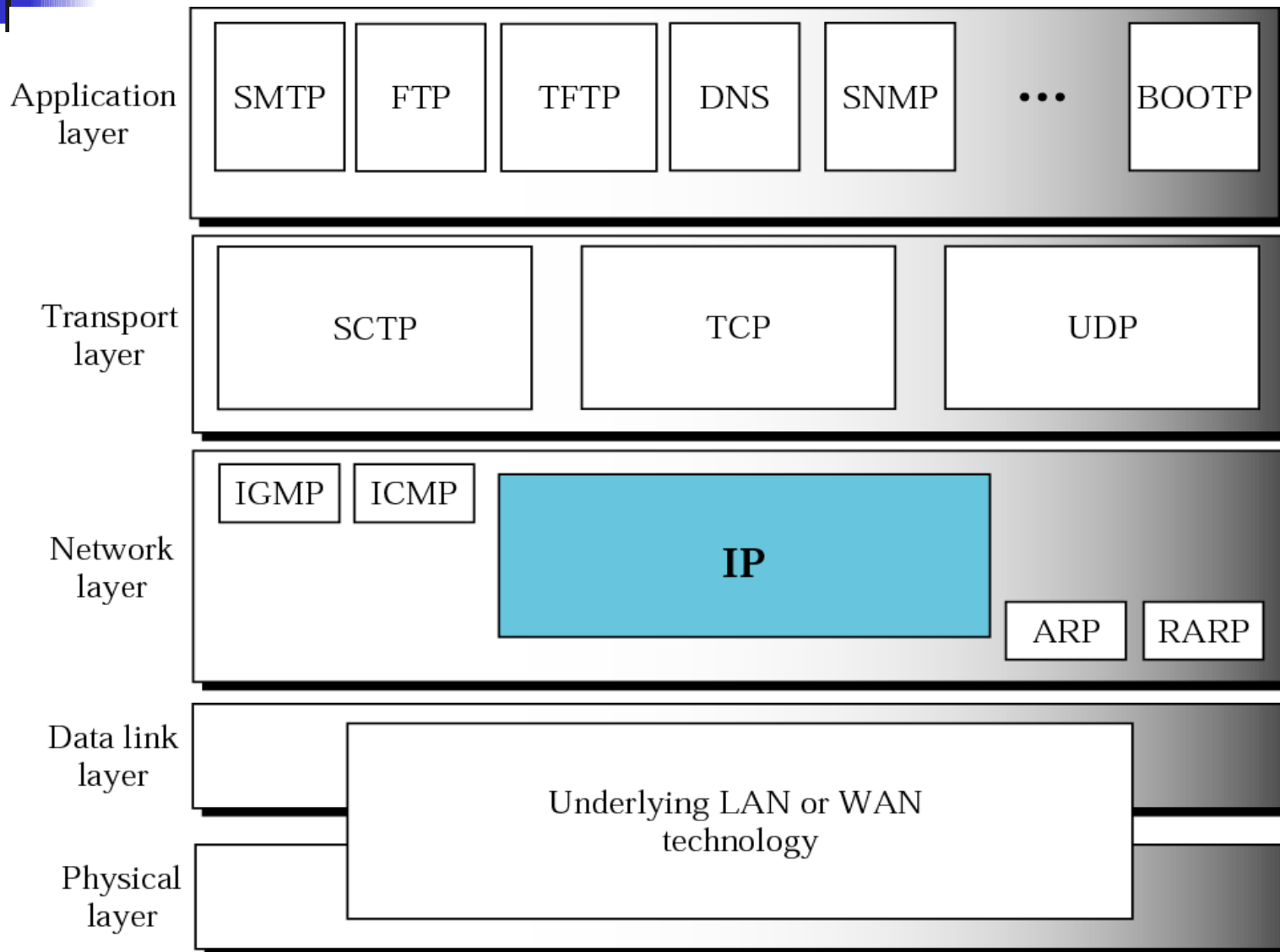
Internet Protocol

Objectives

Upon completion you will be able to:

- *Understand the format and fields of a datagram*
- *Understand the need for fragmentation and the fields involved*
- *Understand the options available in an IP datagram*
- *Be able to perform a checksum calculation*
- *Understand the components and interactions of an IP package*

Figure 8.1 *Position of IP in TCP/IP protocol suite*



Protocol Name: Protocol for packet network connection

Proposed By: Vint Cerf and Bob Kahn

At: IEEE

When: May 1974

RFC: 791

Internet Protocol

Unreliable

Connectionless Datagram Protocol

Best-effort-delivery

No error checking or tracking

Ex. Post Office

IP Paired with TCP for guaranteed delivery

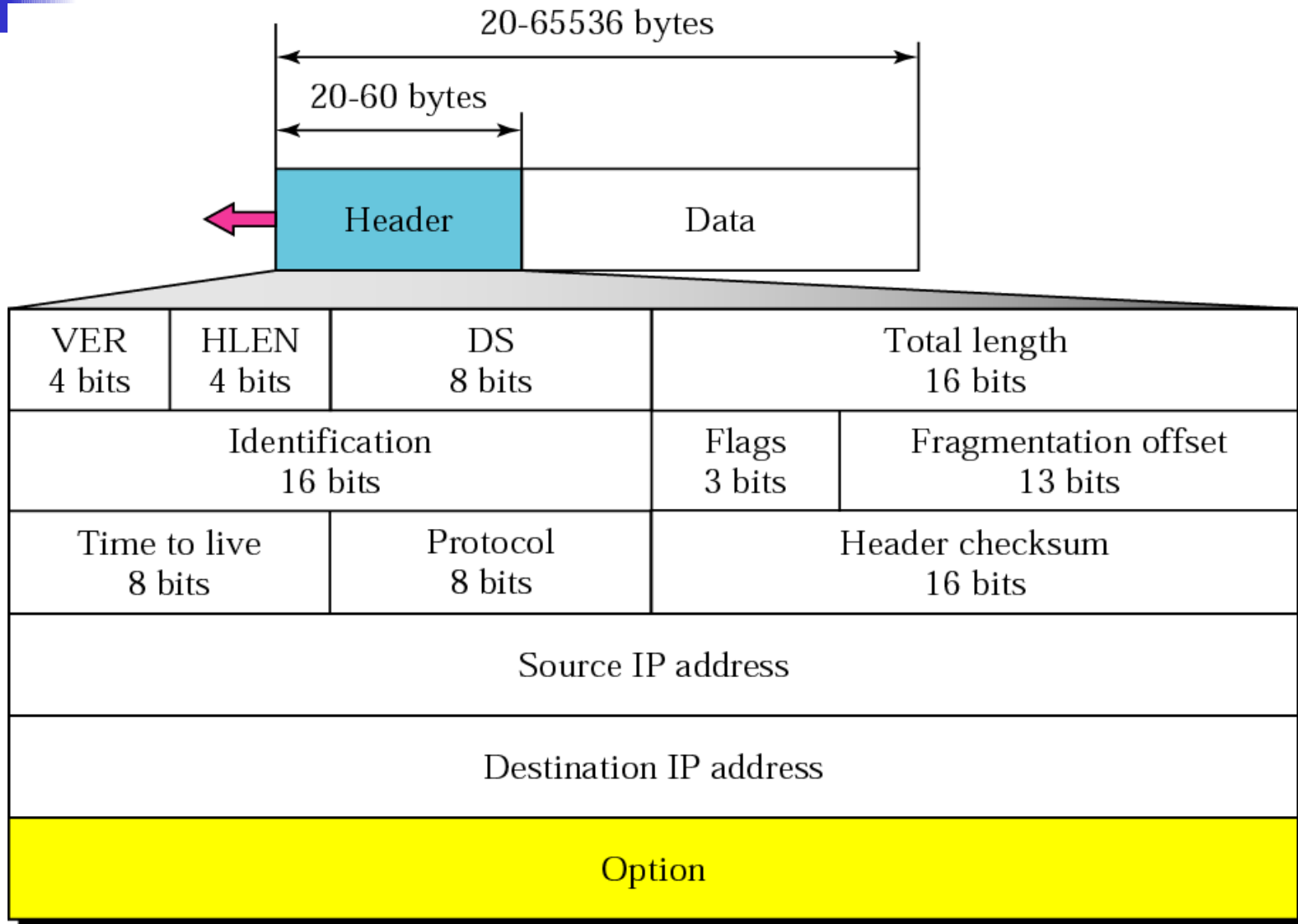
Each datagram handled differently and follow a different route to destination.

IP relies on higher level protocols to resolve problems.

8.1 DATAGRAM

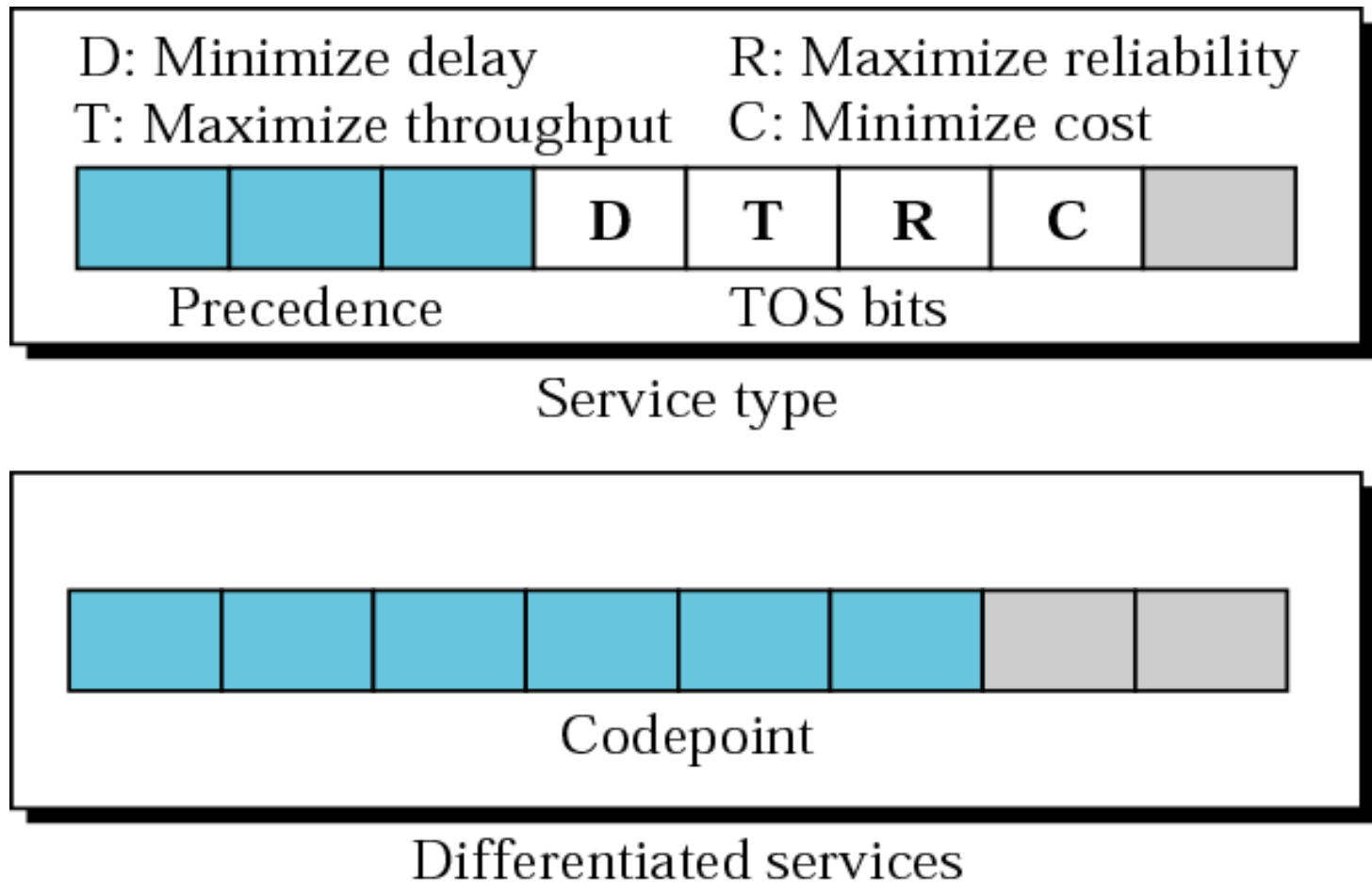
A packet in the IP layer is called a datagram, a variable-length packet consisting of two parts: header and data. The header is 20 to 60 bytes in length and contains information essential to routing and delivery.

Figure 8.2 *IP datagram*



- **Version (VER)** – 4 bit field to specify IP Version.
- **Header Length (HLEN)** – 4 bit field defines total length of header. (From 5 to 15).
- **Differentiated Services (DS)** – 8 bit field. Previously called service type. See Fig.

Figure 8.3 *Service type or differentiated services*



- **Precedence** – 3 bit subfield ranging from 0 to 7. Defines priority of datagram in case of congestion. Lowest priority datagrams discarded first.
- **TOS bits** – 4 bit subfield. Each bit has special meaning. Only one bit can have value of 1 in each datagram.
- Bit patterns are shown in fig.



Note:

***The precedence subfield was designed,
but never used in version 4.***

Table 8.1 Types of service

<i>TOS Bits</i>	<i>Description</i>
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

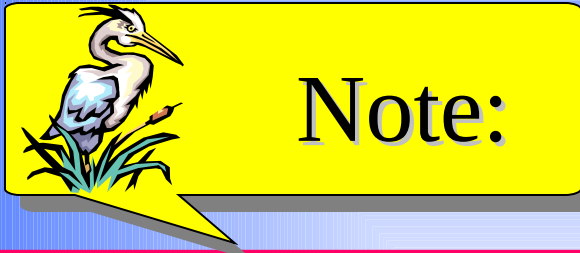
Table 8.2 Default types of service

<i>Protocol</i>	<i>TOS Bits</i>	<i>Description</i>
ICMP	0000	Normal
BOOTP	0000	Normal
NNTP	0001	Minimize cost
IGP	0010	Maximize reliability
SNMP	0010	Maximize reliability
TELNET	1000	Minimize delay
FTP (data)	0100	Maximize throughput
FTP (control)	1000	Minimize delay
TFTP	1000	Minimize delay
SMTP (command)	1000	Minimize delay
SMTP (data)	0100	Maximize throughput
DNS (UDP query)	1000	Minimize delay
DNS (TCP query)	0000	Normal
DNS (zone)	0100	Maximize throughput

- First 6 bits make up codepoint, last 2 bits are not used.
- When 3 right most bits are 0's, 3 left most bits are interpreted same as precedence bits.
- When 3 right most bits are not 0's, 6 bits defines 64 services.
- See Fig.

Table 8.3 *Values for codepoints*

<i>Category</i>	<i>Codepoint</i>	<i>Assigning Authority</i>
1	XXXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental



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



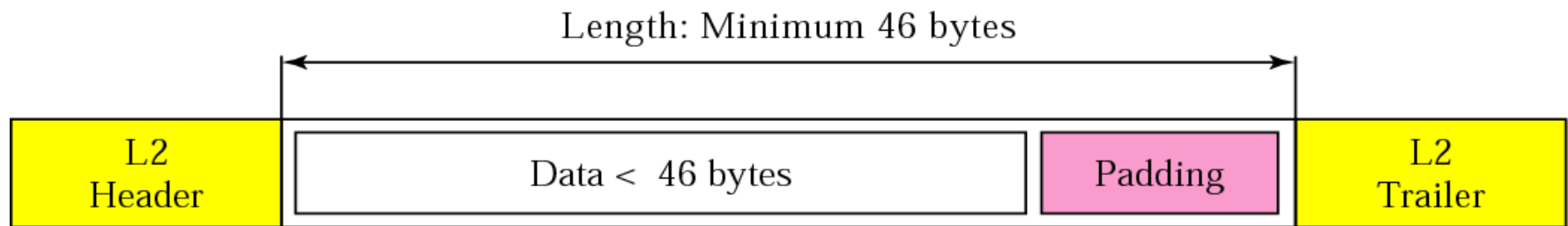
Total Length

- 16-bit field defines total length (Header plus data)

Length of data = total length – header length.

- Header length = $HLEN * 4$.
- As field length is 16-bit, IP datagram is limited to 65,535 bytes.
- Some networks can not handle such a large frames, so we need to fragment the datagram.

Figure 8.4 *Encapsulation of a small datagram in an Ethernet frame*





Other Fields

- **Identification** – Used in fragmentation.
- **Flags** – Used in fragmentation.
- **Fragmentation Offset** – used in fragmentation.
- **Time to Live (TTL)** – Used to hold timestamp decremented by each visited router. For this, all machines should be synchronized and must know how long it takes for datagram for delivery.
 - This field limits the lifetime of a datagram.
 - Used to limit the journey of a packet.
- **Protocol** – 8-bit field. Defines higher level protocol which used IP service. Specifies final destination of protocol to which IP datagram should be delivered.

Figure 8.5 *Multiplexing*

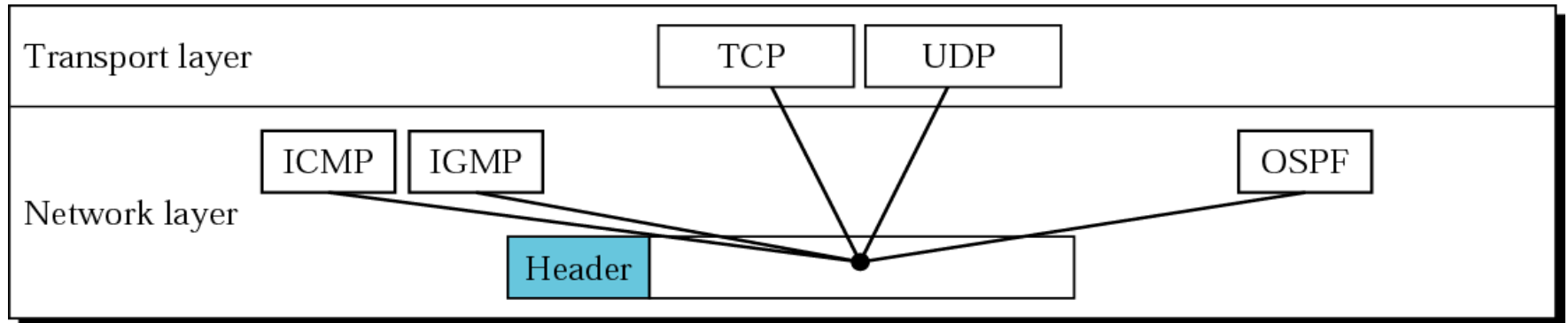


Table 8.4 Protocols

<i>Value</i>	<i>Protocol</i>
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF



Other Fields

- Checksum
- Source Address – 32-bit field defines IP of source. Must remain unchanged.
- Destination Address – 32-bit field defines IP of Destination.



Example 1

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

 ***01000010***

The receiver discards the packet. Why?



Example 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 header length; which means ($2 \times 4 = 8$), which is wrong. The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.



Example 2

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



Example 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 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?



Example 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 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?***



Example 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 8.4).

8.2 FRAGMENTATION

The format and size of a frame depend on the protocol used by the physical network. A datagram may have to be fragmented to fit the protocol regulations.

The topics discussed in this section:

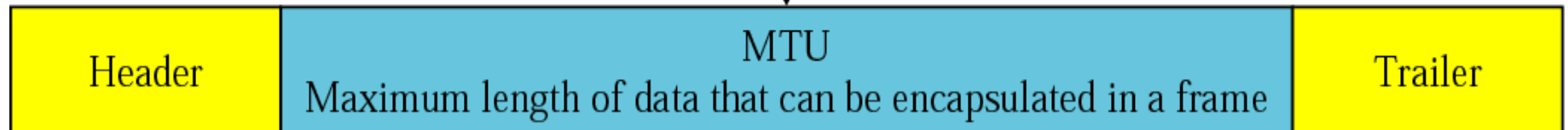
Maximum Transfer Unit (MTU)

Fields Related to Fragmentation

- Each data link layer protocol has its own format.
- Total size of Datagram must be less than maximum size (MTU)
- MTU defined by hardware and software restrictions used in network.
- Value of MTU differs from one physical network to other



MTU



Frame

Table 8.5 MTUs for some networks

<i>Protocol</i>	<i>MTU</i>
Hyperchannel	65,535
Token Ring (16 Mbps)	17,914
Token Ring (4 Mbps)	4,464
FDDI	4,352
Ethernet	1,500
X.25	576
PPP	296



FRAGMENTATION

- IP Allows max. 64K packet size.
- For other physical networks, we must divide the datagram to make it possible to pass through these networks..
- Each fragment has its own header with most of the field repeated but some changed.
- A datagram may be fragmented several times before it reaches to its destination.
- Fragmentation by source host or by router but reassembly is done only by dst. Host.
- Host/Router must change 3 fields. Flags, Fragmentation offset and total length.

- **Identification** : 16-bit field. Identifies a datagram which leaves a source. IP uses counter to label the datagrams.
- When fragmented, value in identification field is copied into all fragments.
- Identification helps in reassembly of datagram.
- **Flags**: 3-bit field. 1st bit is reserved. 2nd bit is do-not-fragment bit. If set to 1, machine should not fragment the datagram. 3rd bit, is more fragment. If set to 1, means datagram is not the last fragment.




Figure 8.7 *Flags field*

D: Do not fragment
M: More fragments





FRAGMENTATION OFFSET

- 13-bit field. Shows relative position of this fragment with respect to whole datagram.
- Value of offset measured in units of 8 bytes.
- Size of each fragment is such that first byte number is divisible by 8.
- Value of more bit set except the last.
- First fragment has offset field value of zero.
- Divide length of first fragment by 8. Second fragment has offset value equal to result.
- Continue the process. Last fragment has more bit as 0.

Figure 8.8 *Fragmentation example*

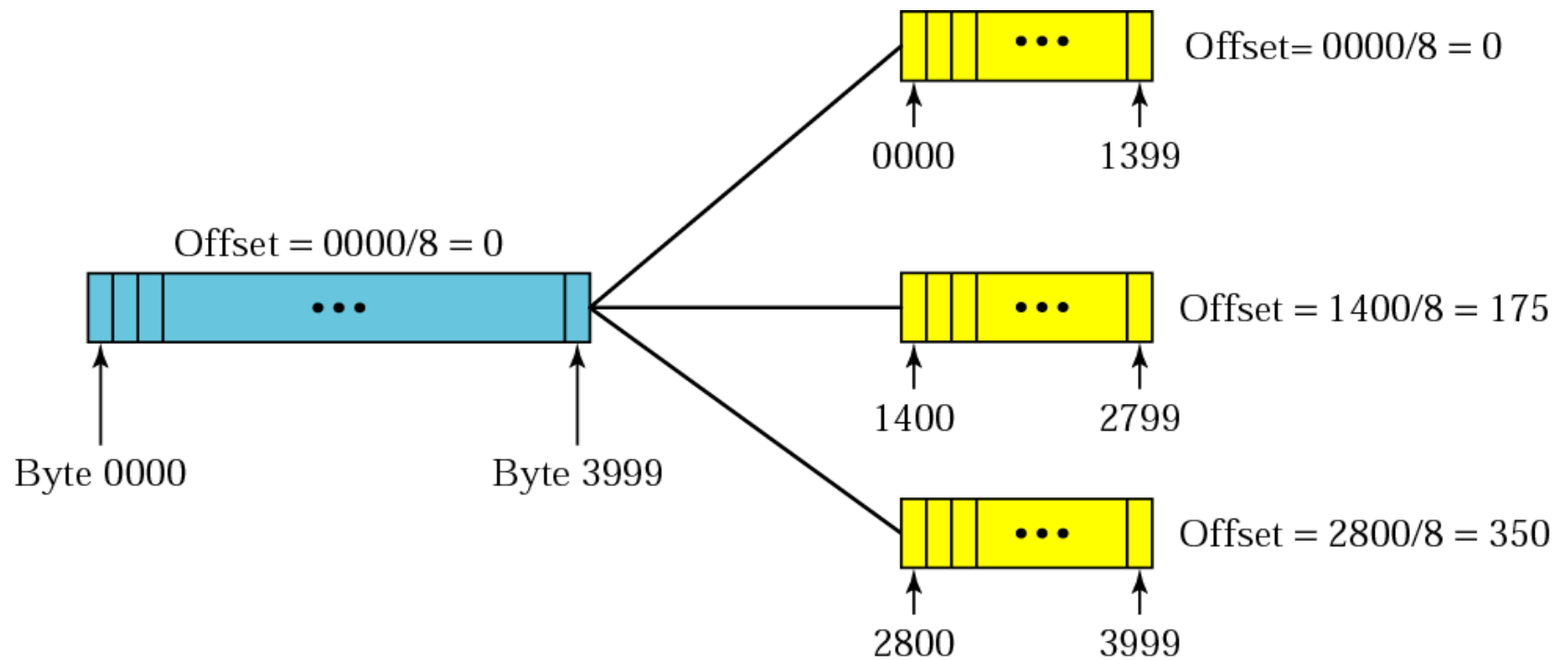
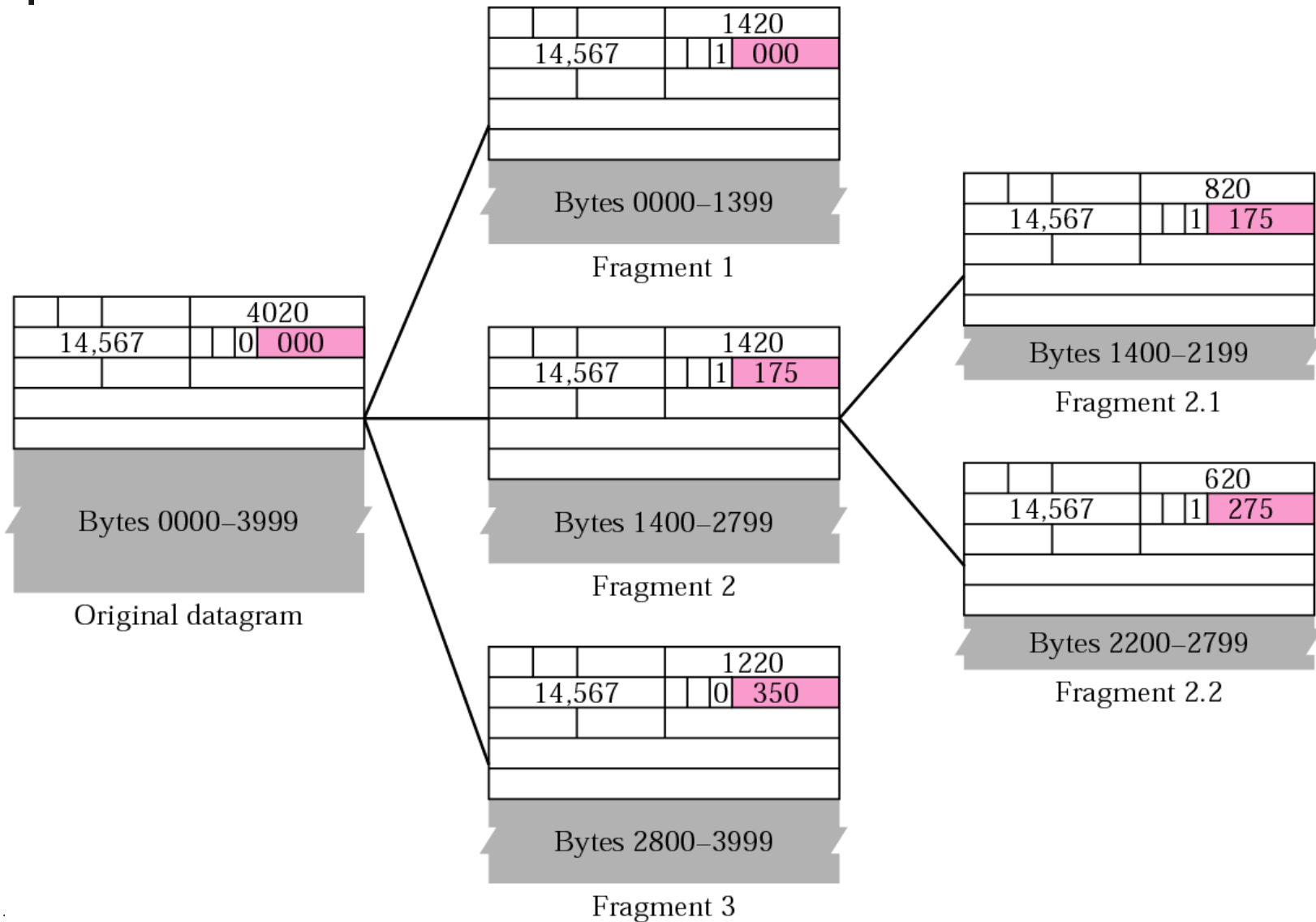


Figure 8.9 *Detailed fragmentation example*





Example 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?



Example 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 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?



Example 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

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?



Example 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 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?



Example 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 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?



Example 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**.*

8.3 OPTIONS

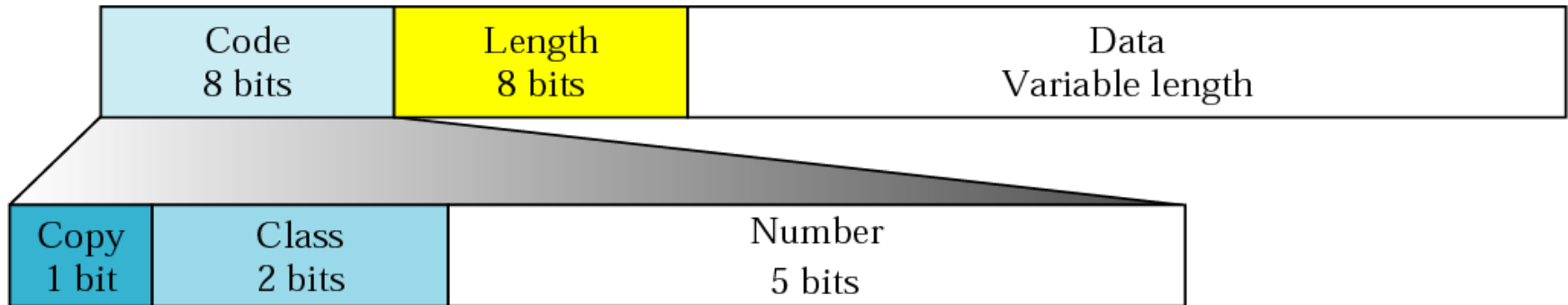
The header of the IP datagram is made of two parts: a fixed part and a variable part. The variable part comprises the options that can be a maximum of 40 bytes.

The topics discussed in this section include:

Format

Option Types

Figure 8.10 *Option format*



Copy

0 Copy only in first fragment

1 Copy into all fragments

Class

00 Datagram control

01 Reserved

10 Debugging and management

11 Reserved

Number

00000 End of option

00001 No operation

00011 Loose source route

00100 Timestamp

00111 Record route

01001 Strict source route

- Code Field- 8 bits long. Contains copy, class and number.
- Copy**: Controls presence of option in fragmentation. If 0, copy options only to first fragment. If 1, copy options to all fragments.
- Class**:
 - 00= Option used for datagram control.
 - 10= Option used for debugging and management.
 - 01 & 11 not defined.
- Number**: (5-bit field), Currently 6 types are in use.



OPTION FORMAT

- **Length**: 8-bit field. Gives total length of option including code field and length field itself.
- **Data**: Not present in all option types.

Figure 8.11 *Categories of options*

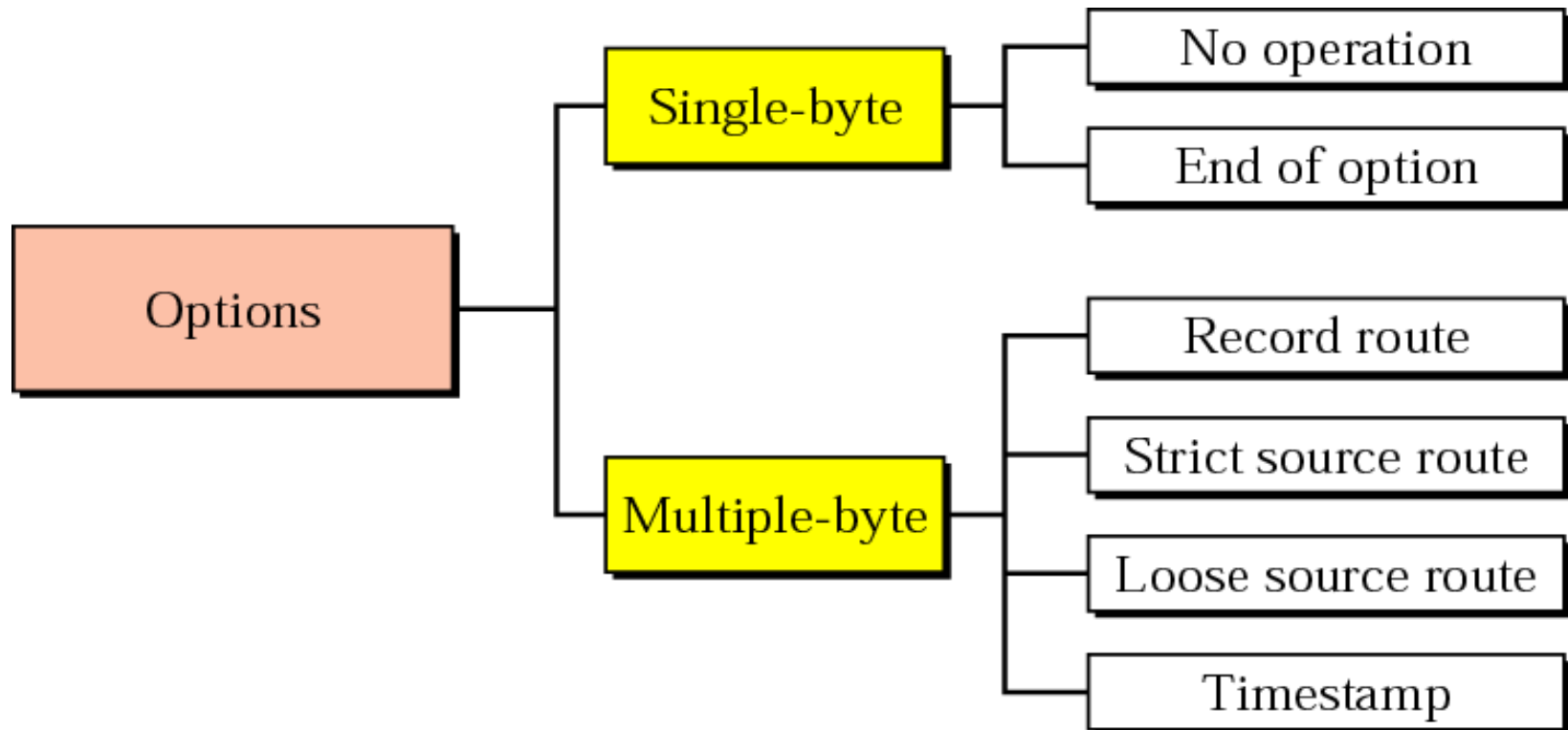
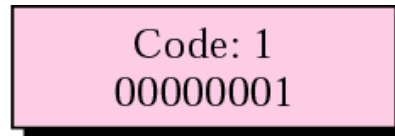
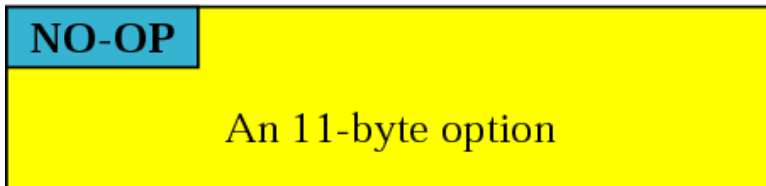


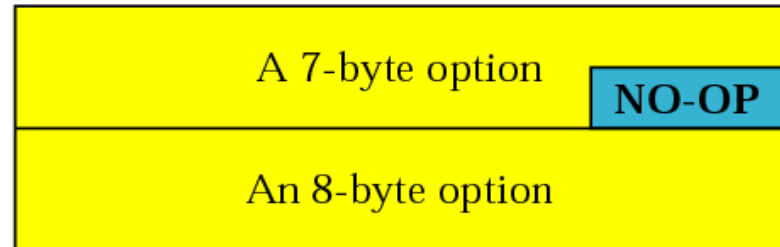
Figure 8.12 *No operation option*



a. No operation option

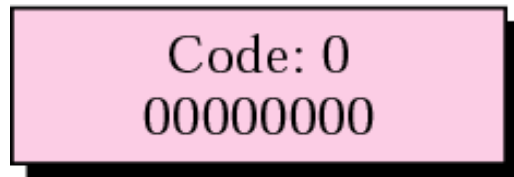


b. Used to align beginning of an option

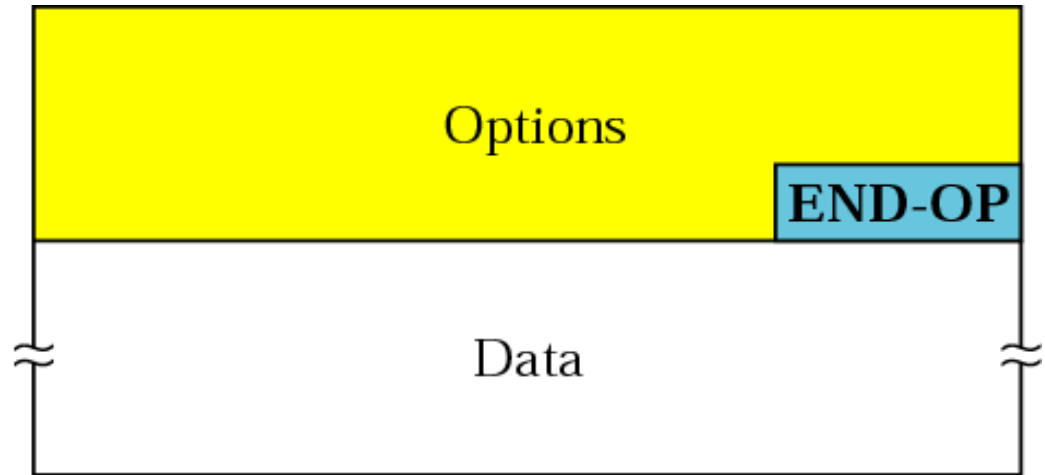


c. Used to align the next option

Figure 8.13 *End of option option*



a. End of option



b. Used for padding

To record Internet routers that handle the datagram.

Maximum 9 router IP addresses are allowed.

Pointer Field points to first available entry. (Default value is 4)

Each router adds the IP address of its interface from which the datagram is leaving.

Router then increments the value of pointer by 4.

See fig.

Figure 8.14 *Record route option*

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

Figure 8.15 *Record route concept*

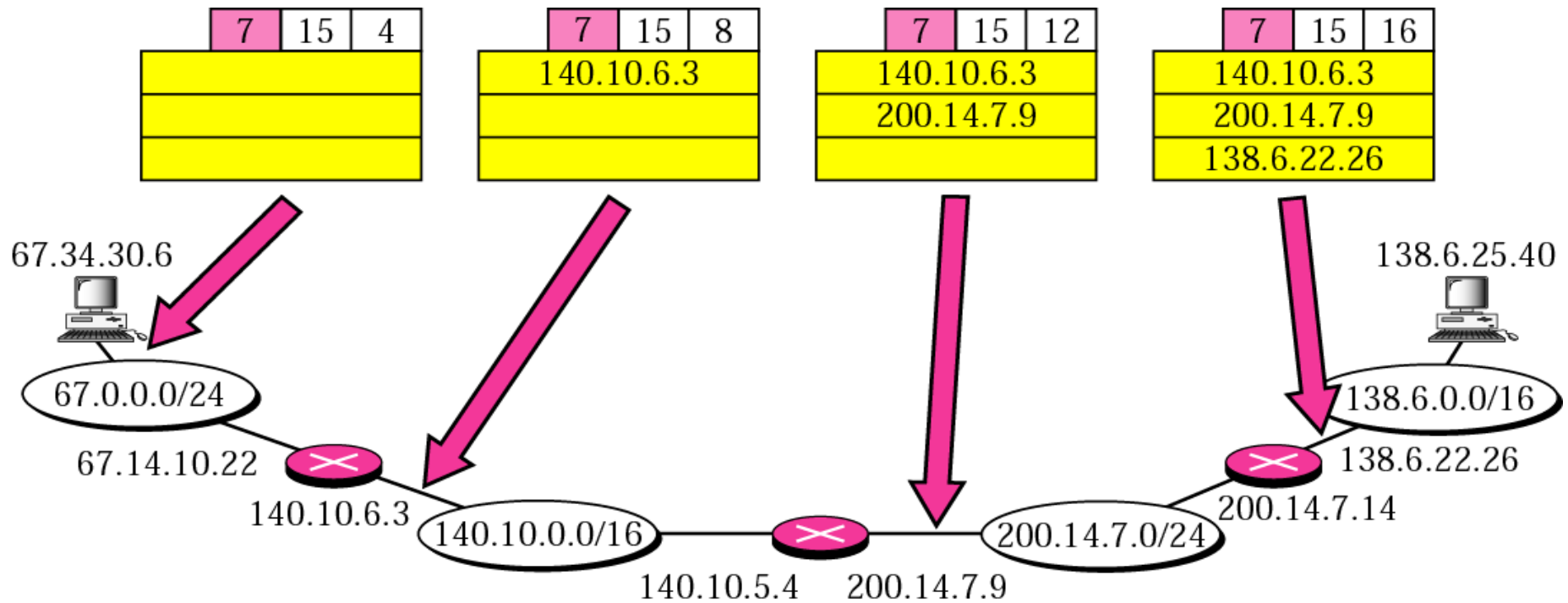
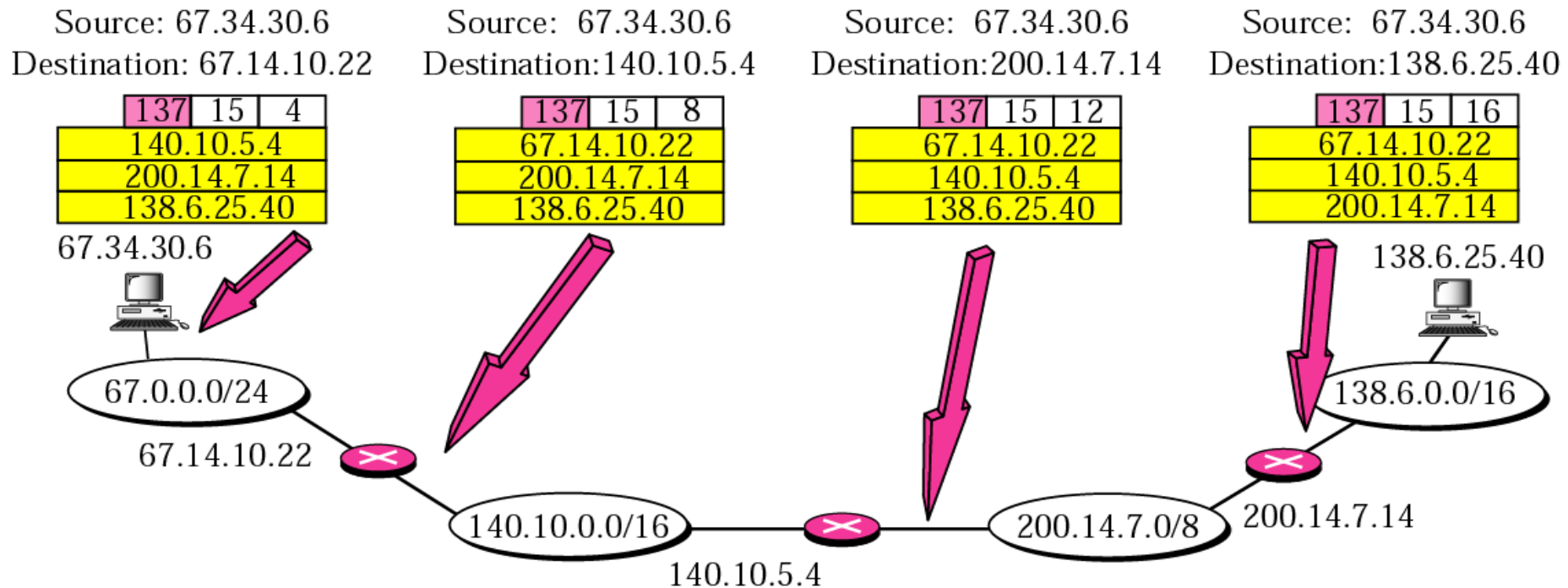


Figure 8.16 *Strict source route option*

Code: 137 10001001	Length (Total length)	Pointer
First IP address (Filled when started)		
Second IP address (Filled when started)		
• • •		
Last IP address (Filled when started)		

Figure 8.17 *Strict source route concept*






Figure 8.18 *Loose source route option*

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

- Used to record the time of datagram processing by a router.
- Expressed in milliseconds from midnight.
- Used to track the behavior of routers in the internet.
- Fig shows format of timestamp option.
- O-flow: Records number of routers that could not add their Timestamps.

Figure 8.19 *Timestamp option*

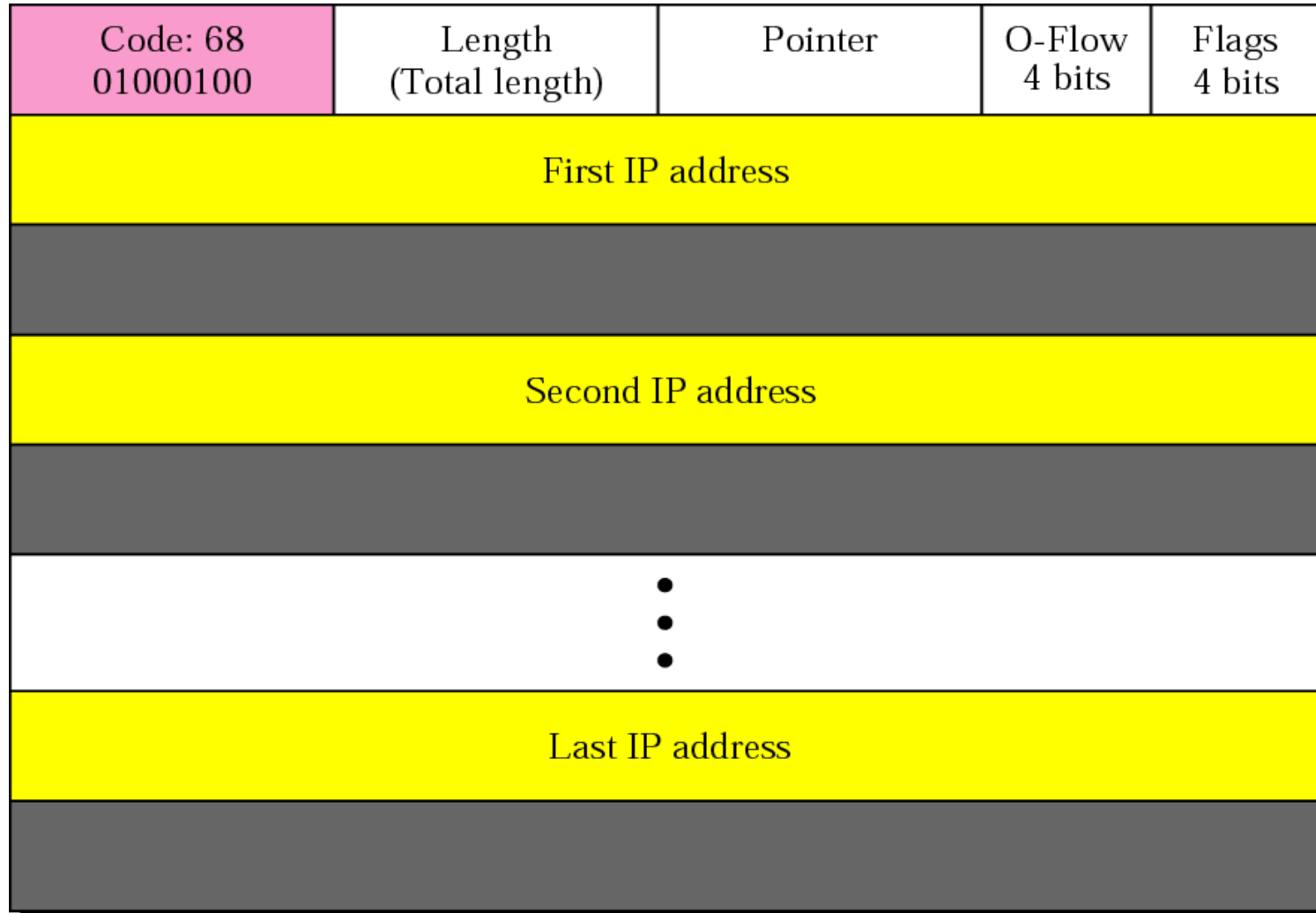


Figure 8.20 *Use of flag in timestamp*

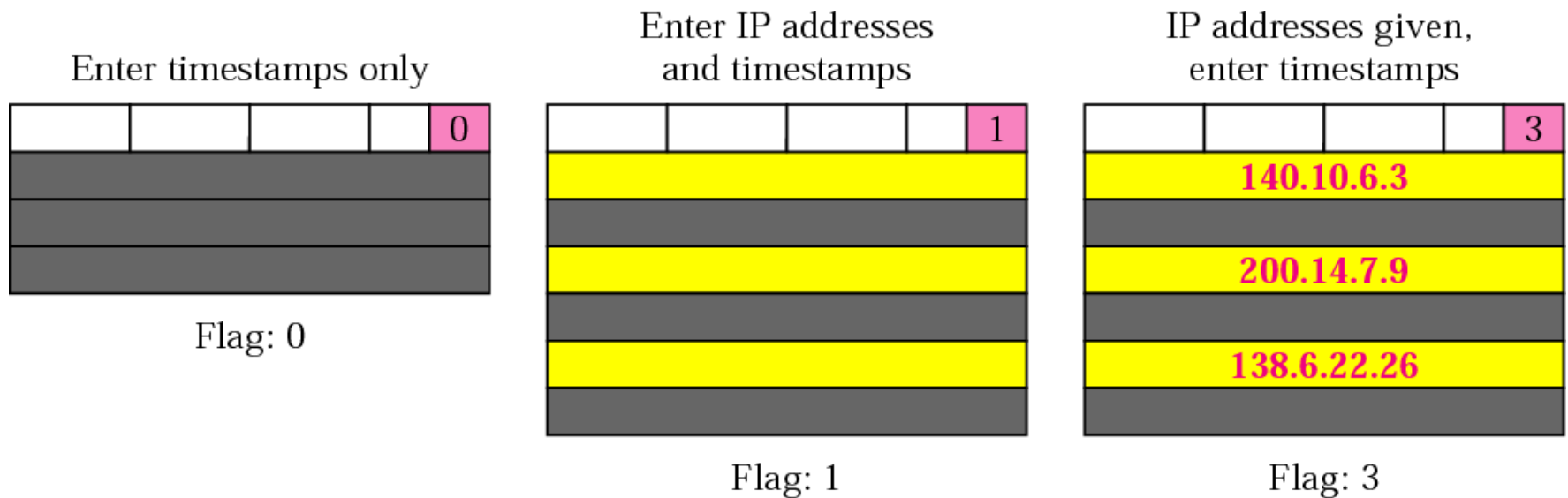
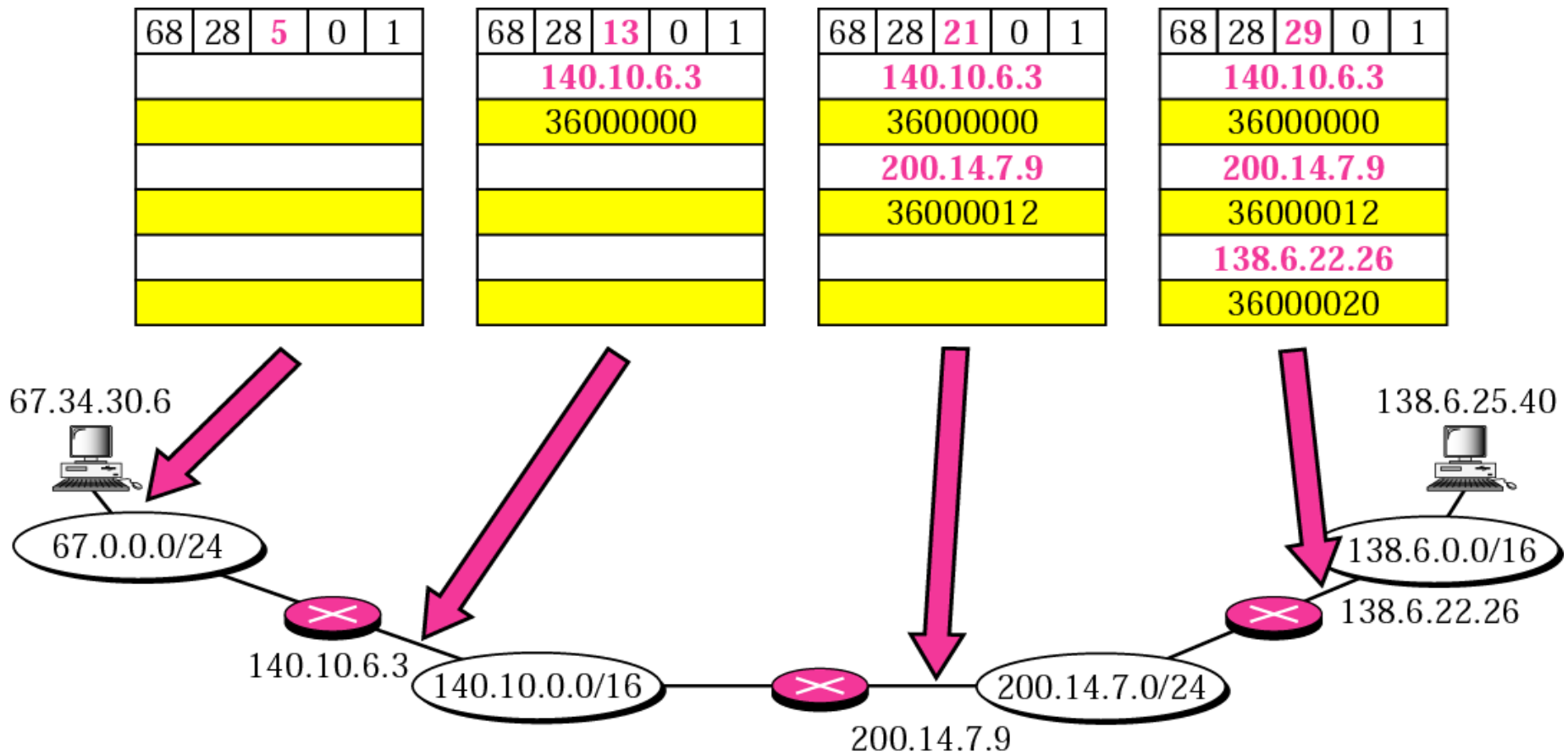
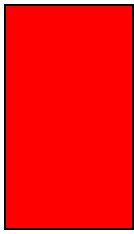


Figure 8.21 *Timestamp concept*





Example 10

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



Example 10

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

Solution

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

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



Example 11

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

Solution

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

- a. No operation: Code is 0**00**00001; datagram control.*
- b. End of option: Code is 0**00**00000; datagram control.*
- c. Record route: Code is 0**00**00111; datagram control.*
- d. Strict source route: Code is 1**00**01001; datagram control.*
- e. Loose source route: Code is 1**00**00011; datagram control.*
- f. Time stamp: Code is 0**10**00100; debugging and management control.*



Example 12

*One of the utilities available in UNIX to check the travelling 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**.*

```
$ 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): ....
```

The result shows the IP address of the host and the number of bytes used.



Example 13

We can also use the **ping** utility with the **-R** option to implement the record route option.

```
$ 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)
```

The result shows the interfaces and IP addresses.



Example 14

*The **traceroute** utility can also be used to keep track of the route of a packet.*

```
$ 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
```

The result shows the three routers visited.



Example 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 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, not any other one.

```
$ 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
```

8.4 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 topics discussed in this section include:

Checksum Calculation at the Sender

Checksum Calculation at the Receiver

Checksum in the IP Packet



Note:

To create the checksum the sender does the following:

- The packet is divided into k sections, each of n bits.***
- All sections are added together using 1's complement arithmetic.***
- The final result is complemented to make the checksum.***

Figure 8.22 *Checksum concept*

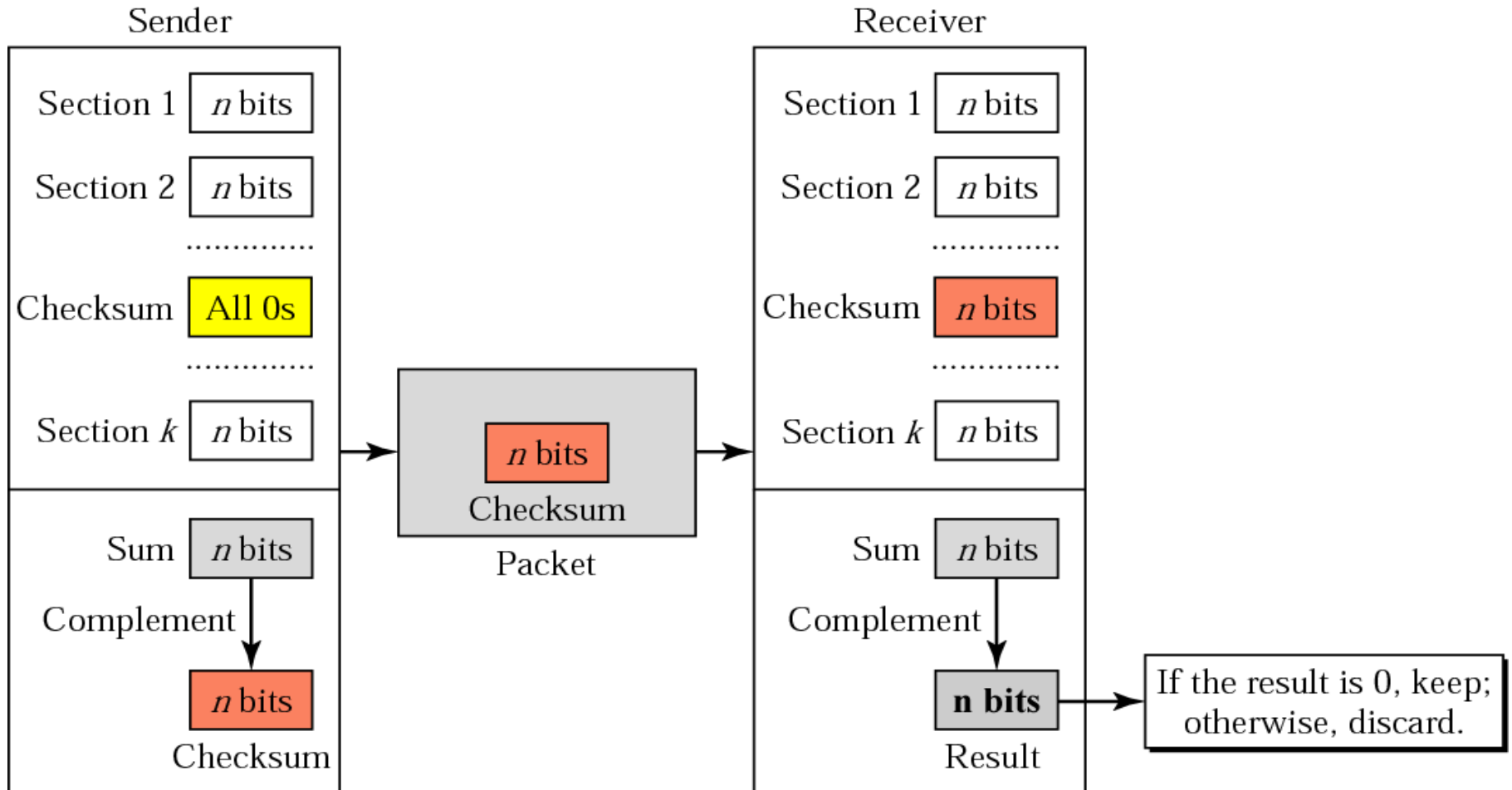
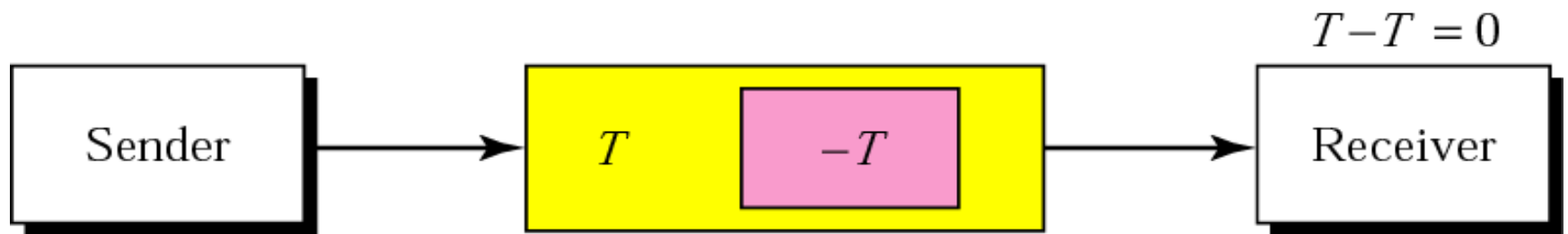
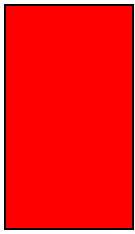


Figure 8.23 *Checksum in one's complement arithmetic*





Example 17

Figure 8.24 shows an example of a checksum calculation 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.

See Next Slide

Figure 8.24 *Example of checksum calculation in binary*

	5	0	28	
	1	0	0	
4	17	0		↑
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	
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	←



Example 18

Let us do the same example in hexadecimal. Each row has four hexadecimal digits. We calculate the sum first. Note that if an addition results in more than one hexadecimal digit, the right-most digit becomes the current-column digit and the rest are carried to other columns. From the sum, we make the checksum by complementing the sum. However, note that we subtract each digit from 15 in hexadecimal arithmetic (just as we subtract from 1 in binary arithmetic). This means the complement of E (14) is 1 and the complement of 4 is B (11). Figure 8.25 shows the calculation. Note that the result (8BB1) is exactly the same as in Example 17.

See Next Slide

Figure 8.25 *Example of checksum calculation in hexadecimal*

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

4, 5, and 0	→	4	5	0	0
28	→	0	0	1	C
1	→	0	0	0	1
0 and 0	→	0	0	0	0
4 and 17	→	0	4	1	1
0	→	0	0	0	0
10.12	→	0	A	0	C
14.5	→	0	E	0	5
12.6	→	0	C	0	6
7.9	→	0	7	0	9
Sum	→	7	4	4	E
Checksum	→	8	B	B	1



Note:

Check [Appendix C](#) for a detailed description of checksum calculation and the handling of carries.

8.5 IP PACKAGE

We give an example of a simplified IP software package to show its components and the relationships between the components. This IP package involves eight modules.

The topics discussed in this section include:

Header-Adding Module

Processing Module

Queues

Routing Table

Forwarding Module

MTU Table

Fragmentation Module

Reassembly Table

Reassembly Module

Figure 8.26 *IP components*

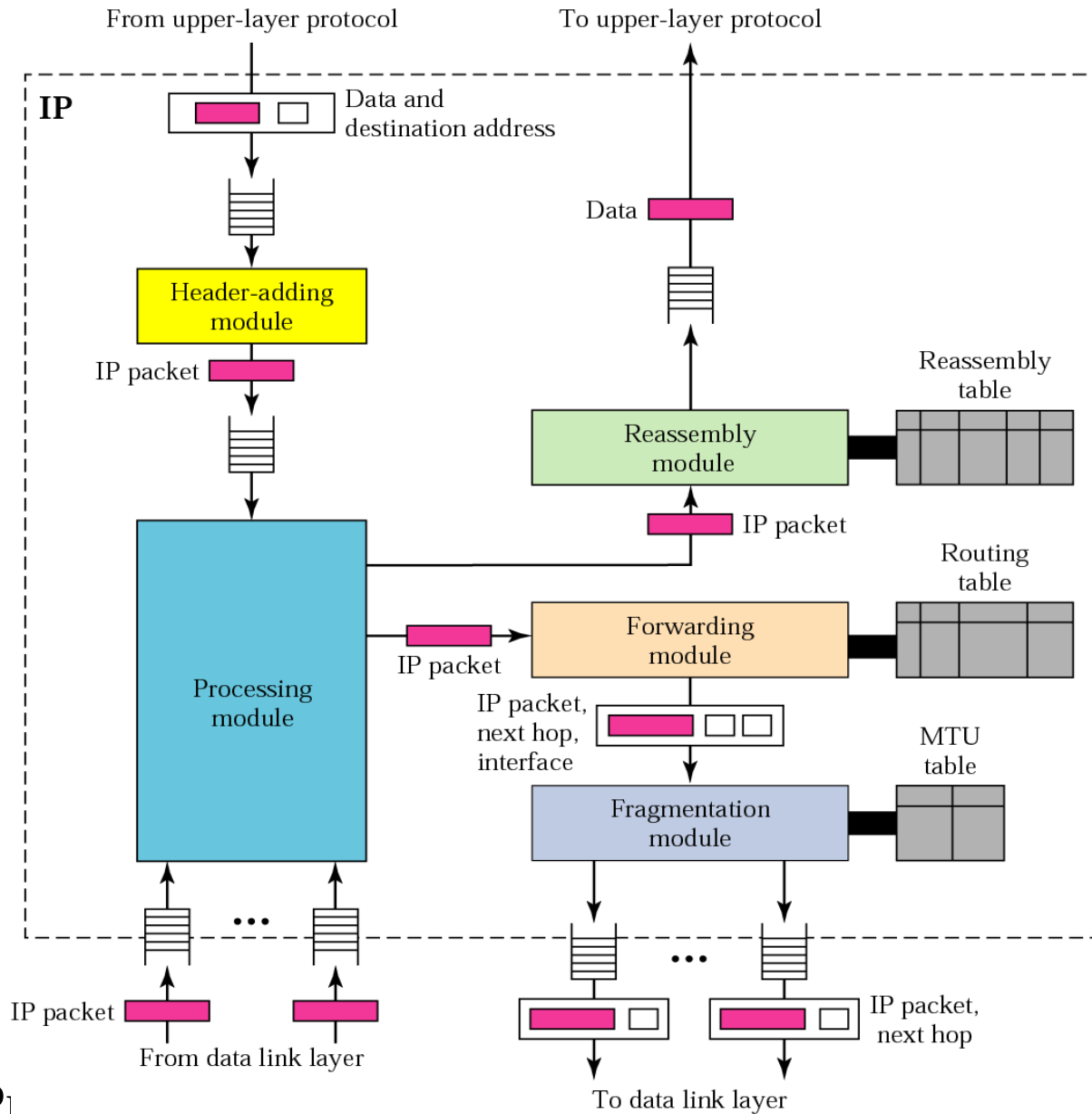




Figure 8.27 *MTU table*

Interface Number	MTU
.....

Figure 8.28 *Reassembly table*

St.: State

S. A.: Source address

T. O.: Time-out

D. I.: Datagram ID

F.: Fragments

