7-4 OPTIONS

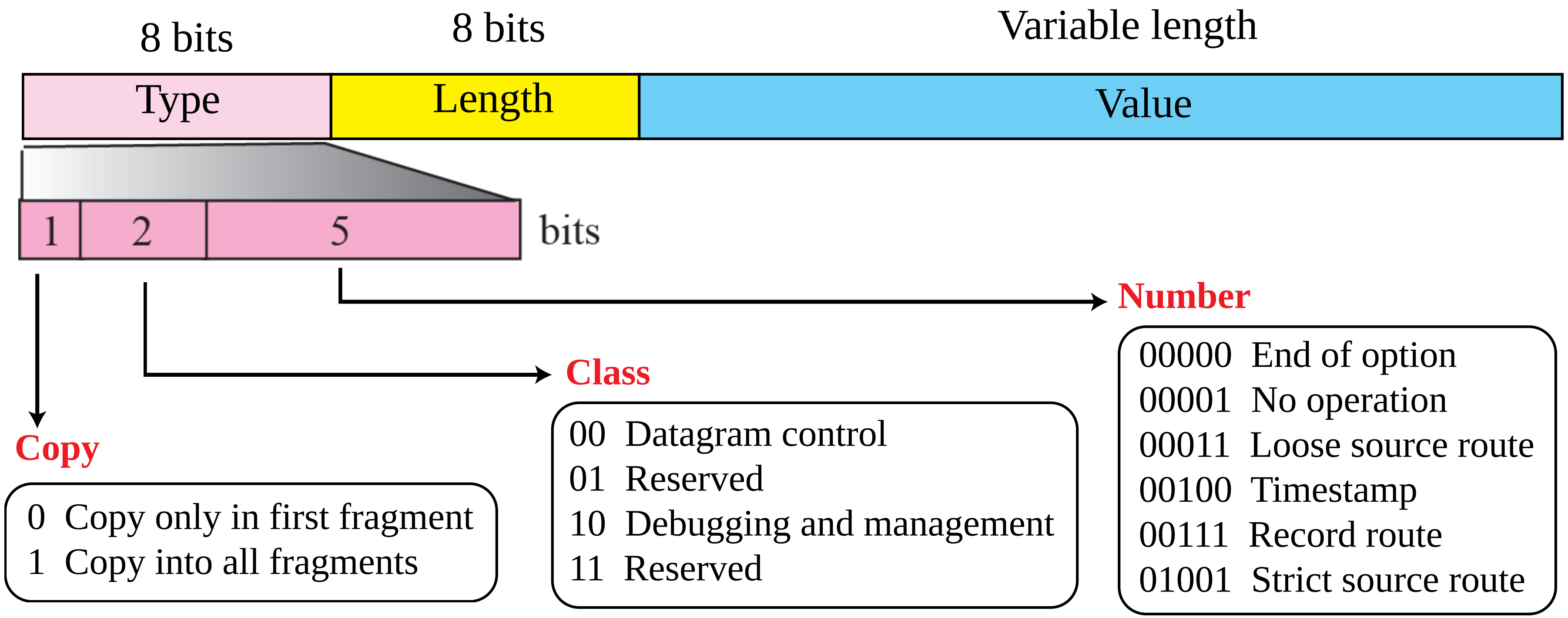
The header of the IP datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section. The variable part comprises the options, which can be a maximum of 40 bytes.

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*



0 Copy only in first fragment
1 Copy into all fragments

00 Datagram control
01 Reserved
10 Debugging and management
11 Reserved

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

Copy: This 1-bit subfield controls the presence of the option in fragmentation

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.

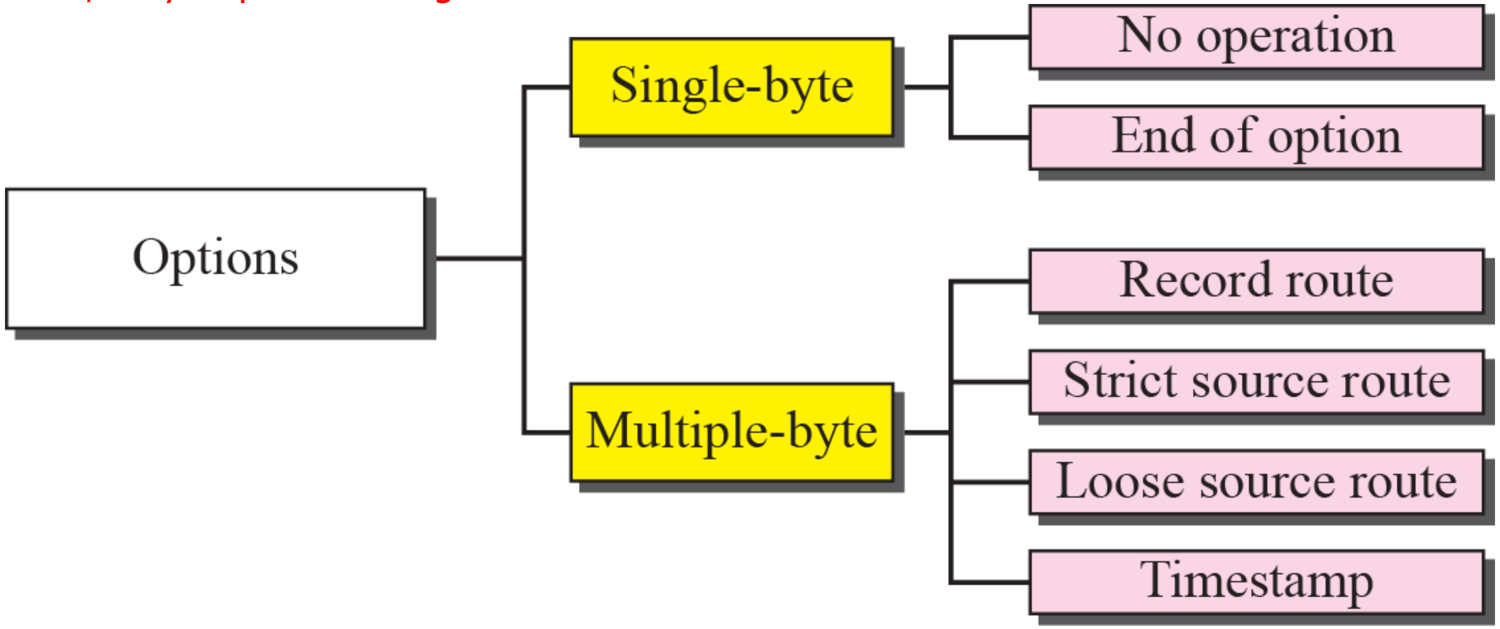
Number: This 5-bit subfield defines the type of option. Although 5 bits can define up to 32 different types, currently, only six options are being used.

Figure 7.10 : *Option format*

- **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.

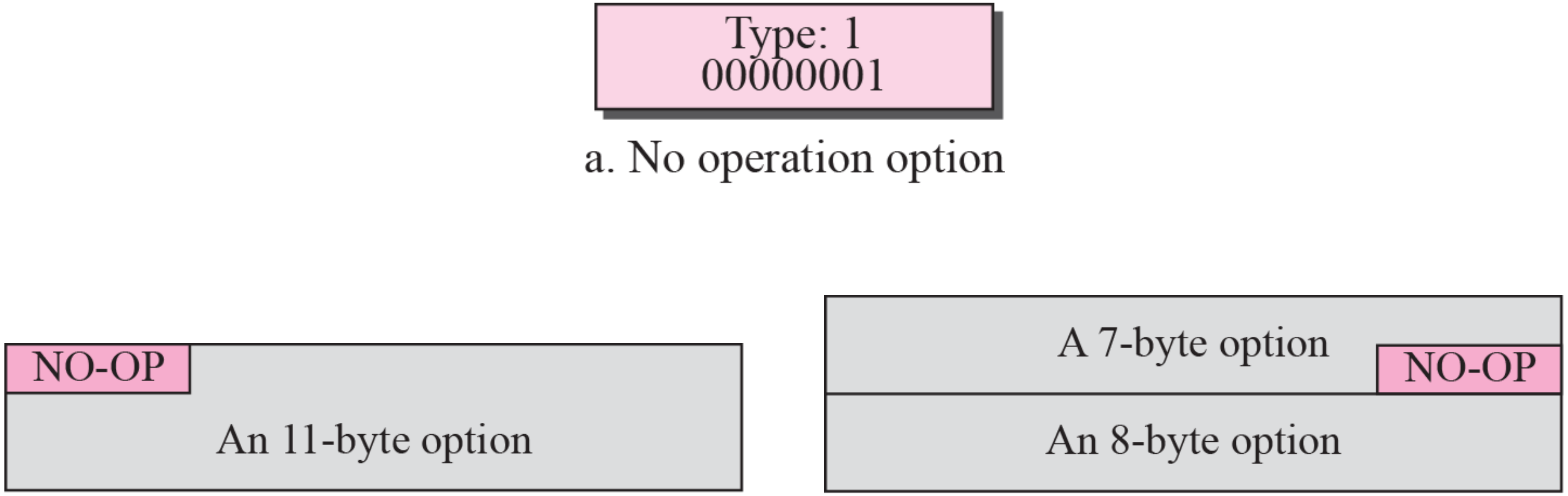
Figure 7.11 *Categories of options*

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 .



A no-operation option is a 1-byte option used as a filler between options. it can be used to align the next option on a 16-bit or 32-bit boundary

Figure 7.12 *No operation option*



a. No operation option

b. Used to align beginning of an option

c. Used to align the next option

Figure 7.13 *End-of-option option*

An end-of-option 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 data.

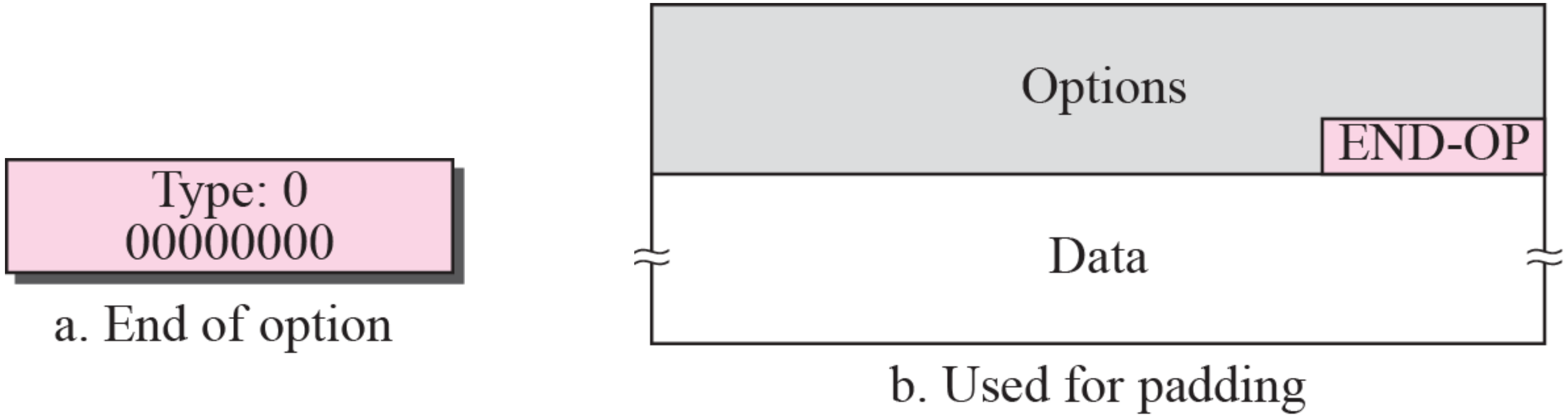
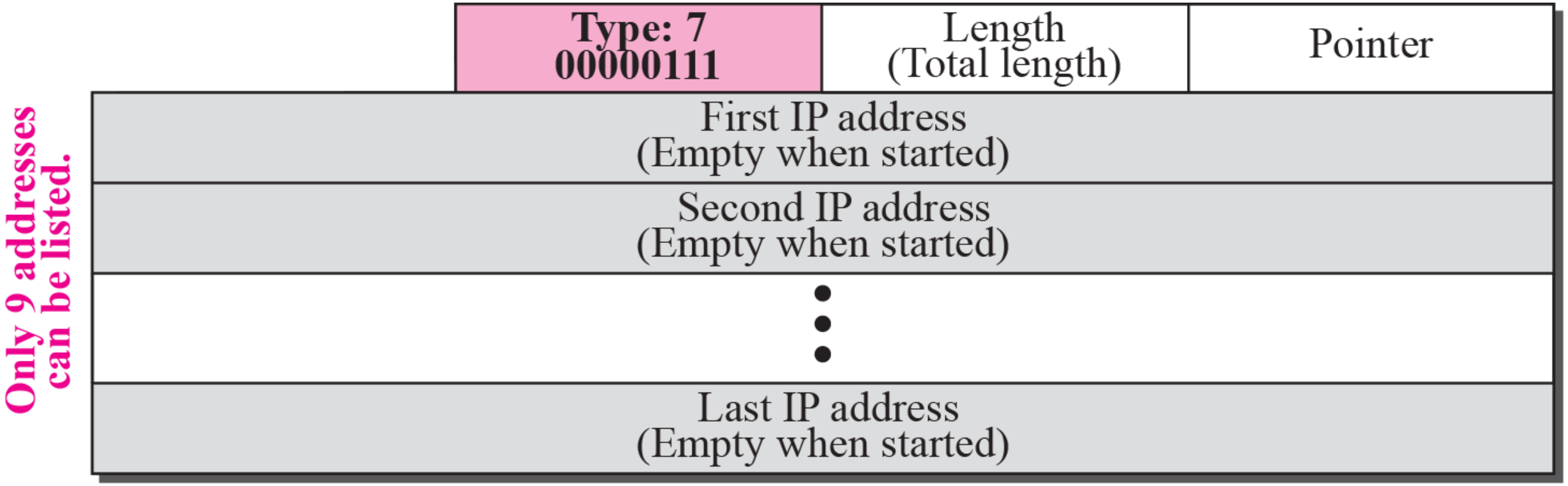


Figure 7.14 *Record-route option*

A record-route option: is used to record the internet routers that handle the datagram. It can list up to nine router IP addresses since the maximum size of the header is 60 bytes, The source creates placeholder fields in the option to be filled by the visited routers.



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*

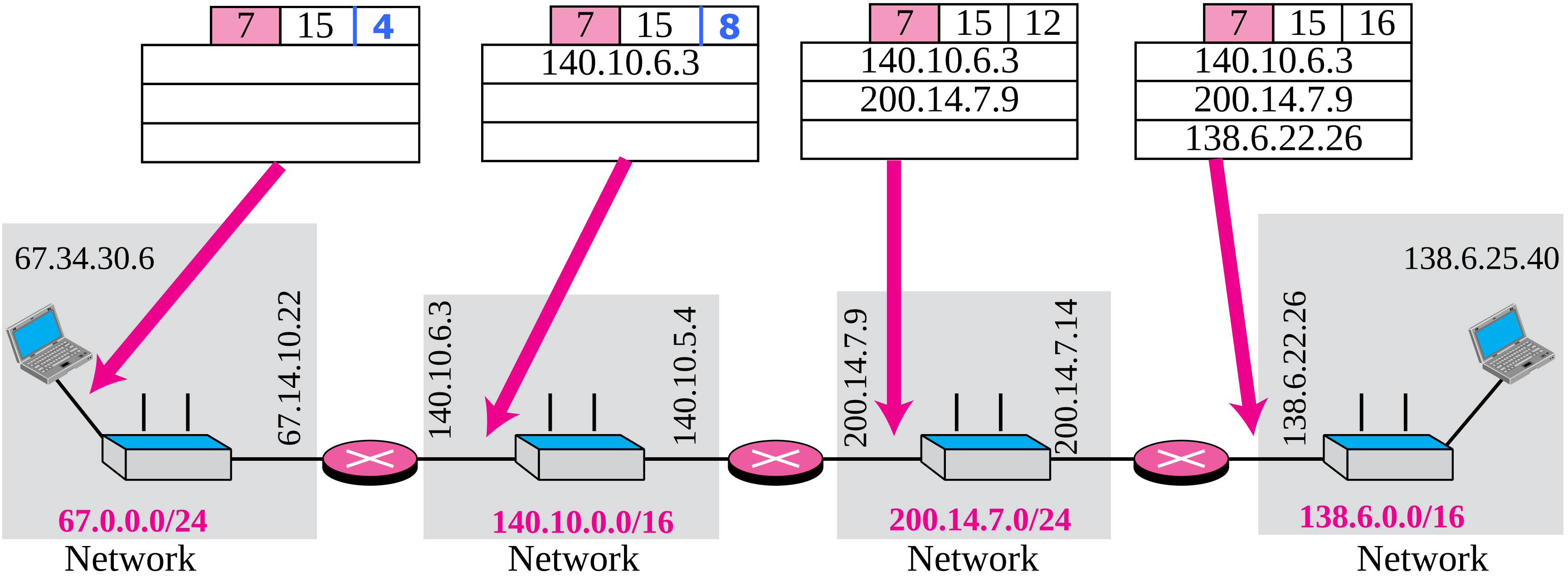


Figure 7.16 *Strict-source-route option*

A strict-source-route option: is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput. Alternatively, it may choose a route that is safer or more reliable for the sender’s purpose.

If the datagram visits a router that is not on the list, the datagram is discarded and an error message is issued. And If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded and an error message issued.

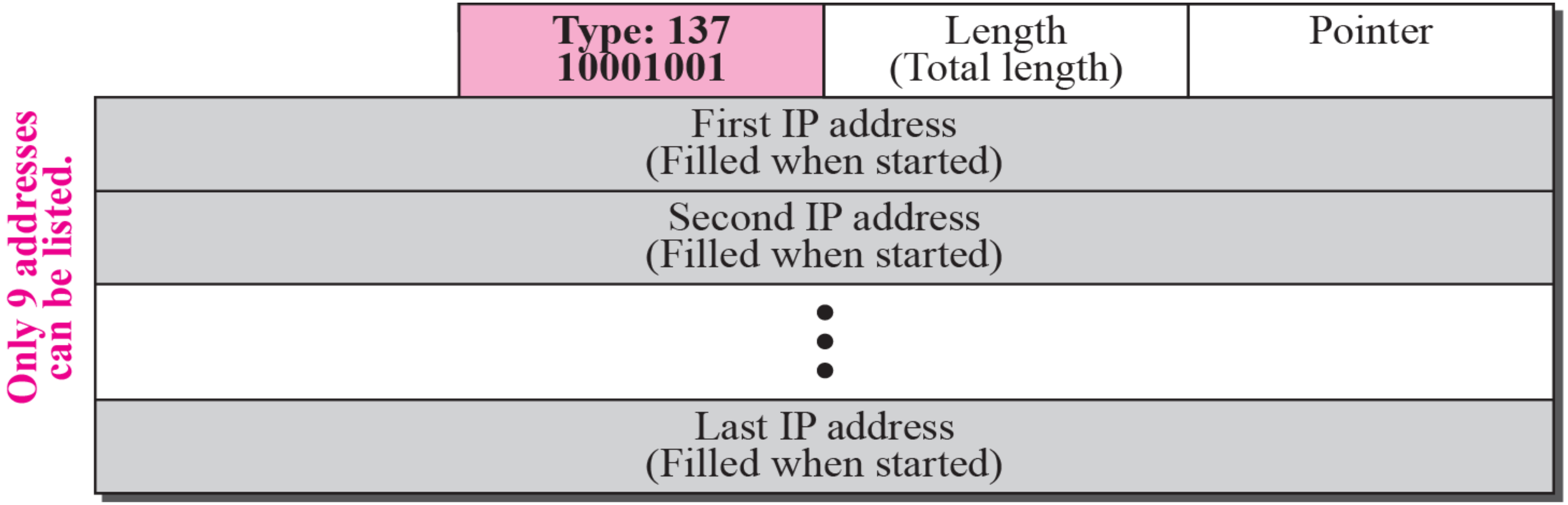
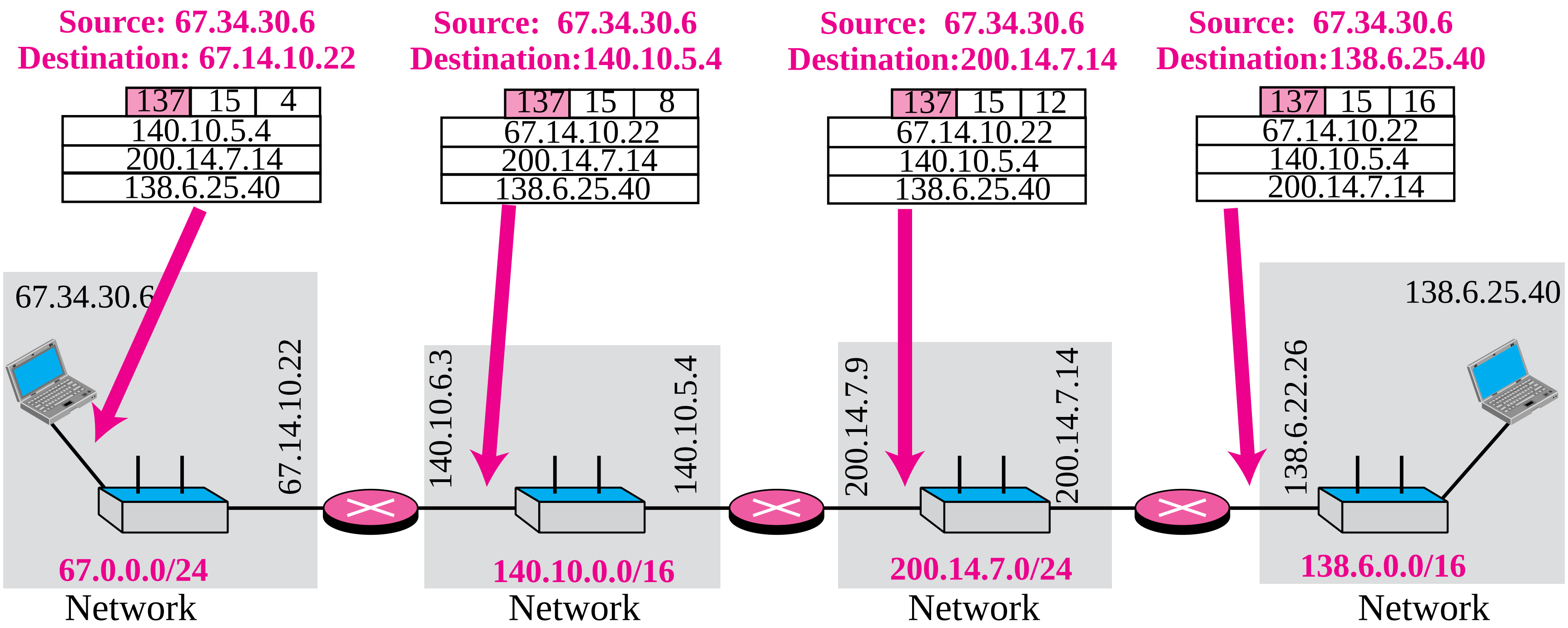


Figure 7.17 *Strict-source-route option*



Source: 67.34.30.6
Destination: 67.14.10.22

137	15	4
140.10.5.4		
200.14.7.14		
138.6.25.40		

Source: 67.34.30.6
Destination:140.10.5.4

137	15	8
67.14.10.22		
200.14.7.14		
138.6.25.40		

Source: 67.34.30.6
Destination:200.14.7.14

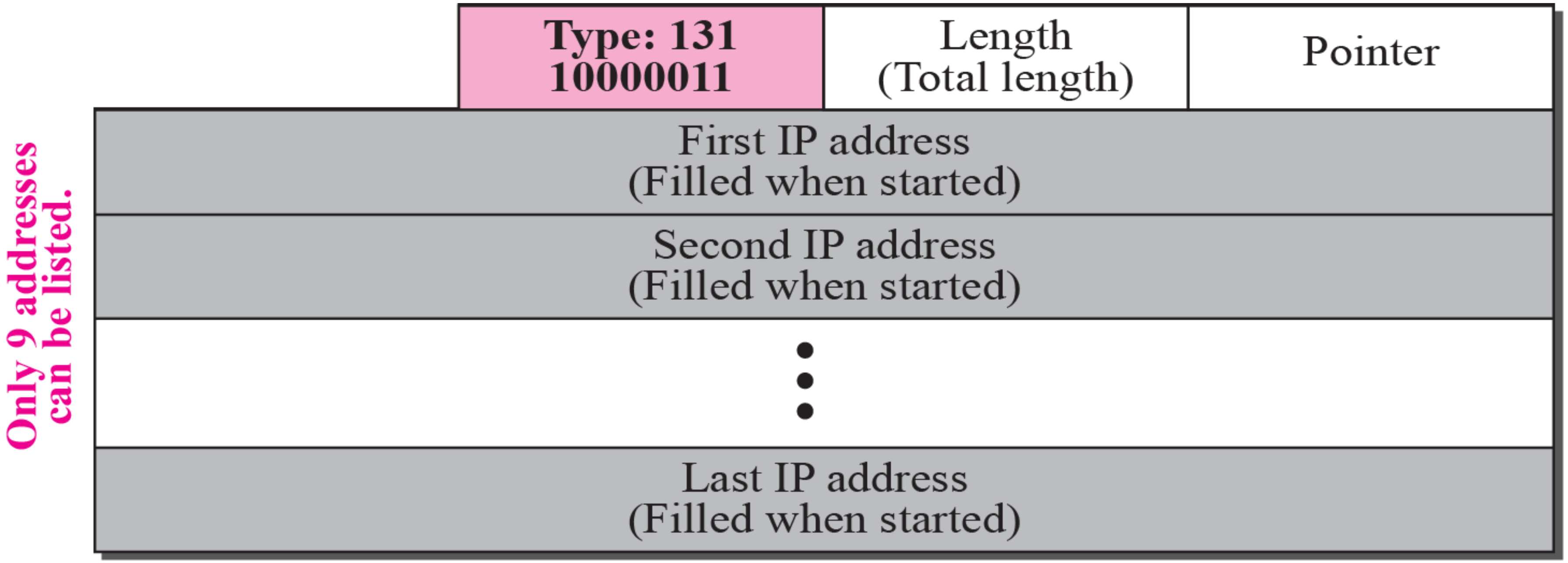
137	15	12
67.14.10.22		
140.10.5.4		
138.6.25.40		

Source: 67.34.30.6
Destination:138.6.25.40

137	15	16
67.14.10.22		
140.10.5.4		
200.14.7.14		

Figure 7.18 *Loose-source-route option*

A loose-source-route option: is similar to the strict source route, but it is more relaxed. Each router in the list must be visited, but the datagram can visit other routers as well.



Type: 131
10000011

Length
(Total length)

Pointer

First IP address
(Filled when started)

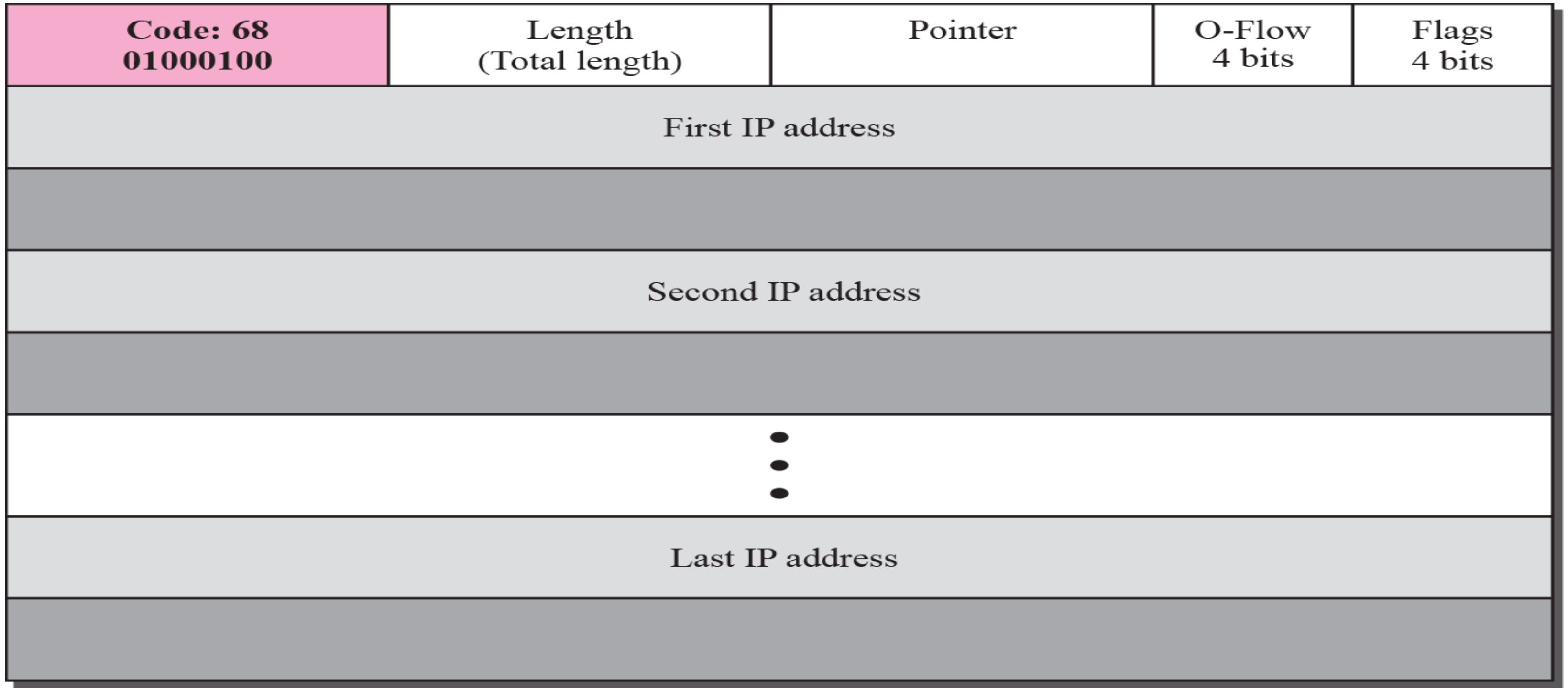
Second IP address
(Filled when started)

Last IP address
(Filled when started)

Only 9 addresses can be listed.

Figure 7.19 *Time-stamp option*

A timestamp option: is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal Time. Knowing the time a datagram is processed can help users and managers track the behaviour of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another.



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*

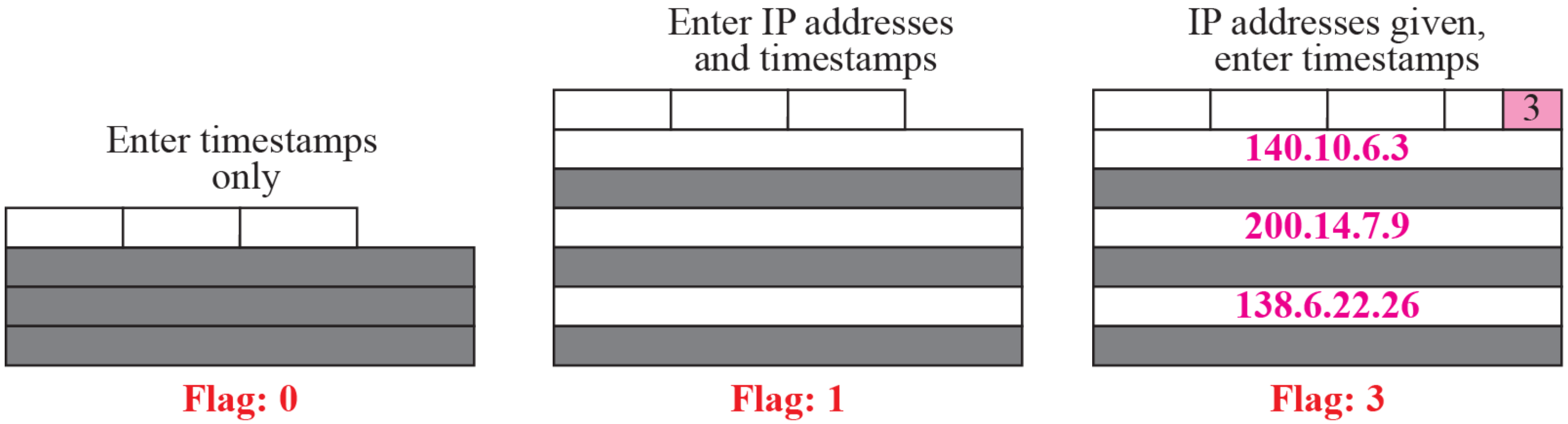
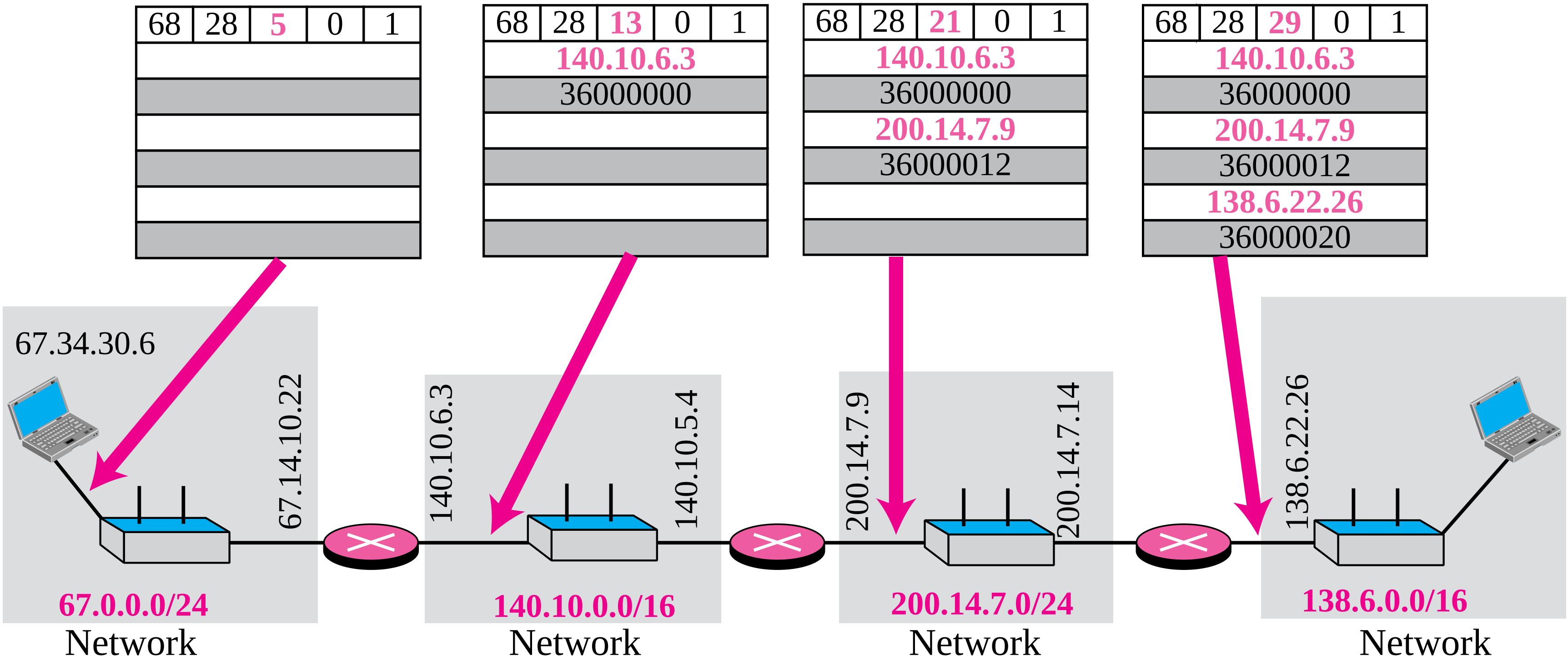


Figure 7.21 *Timestamp concept*



68	28	5	0	1

68	28	13	0	1
140.10.6.3				
36000000				

68	28	21	0	1
140.10.6.3				
36000000				
200.14.7.9				
36000012				

68	28	29	0	1
140.10.6.3				
36000000				
200.14.7.9				
36000012				
138.6.22.26				
36000020				

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.

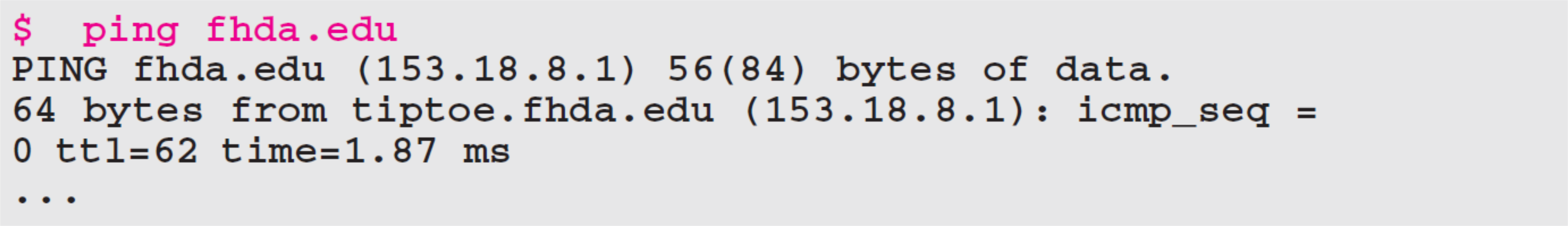
d. Strict source route: type is 10001001; datagram control.

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

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

Example 7.12

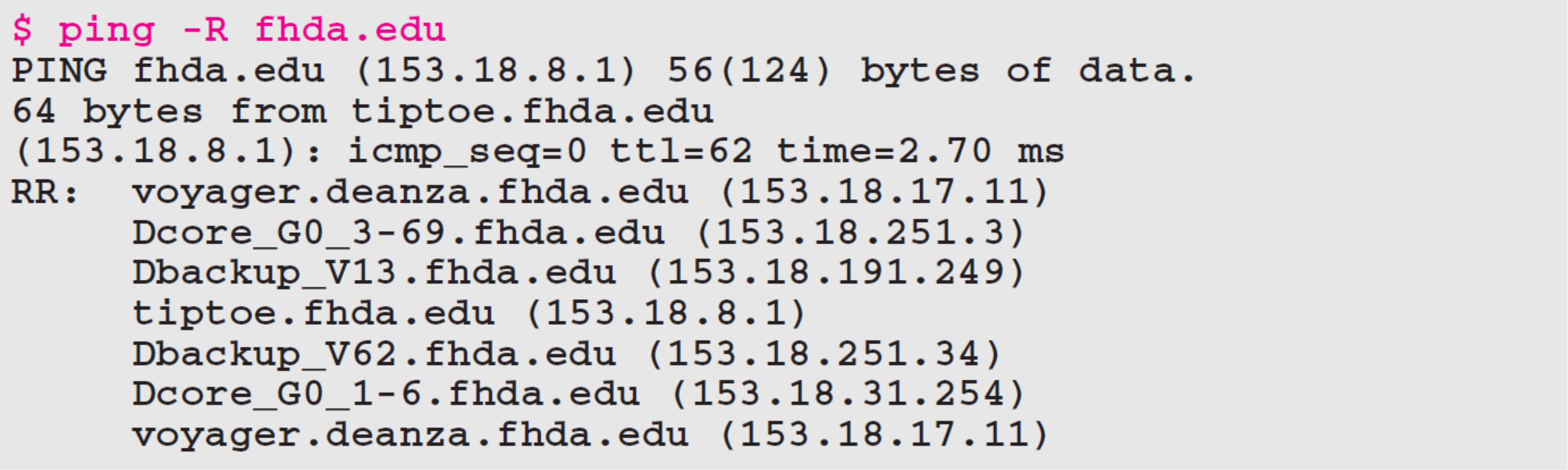
One of the utilities available in UNIX to check the traveling of the IP packets is ping. In the next chapter, we talk about the ping program in more detail. In this example, we want to show how to use the program to see if a host is available. We ping a server at De Anza College named fhda.edu. The result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

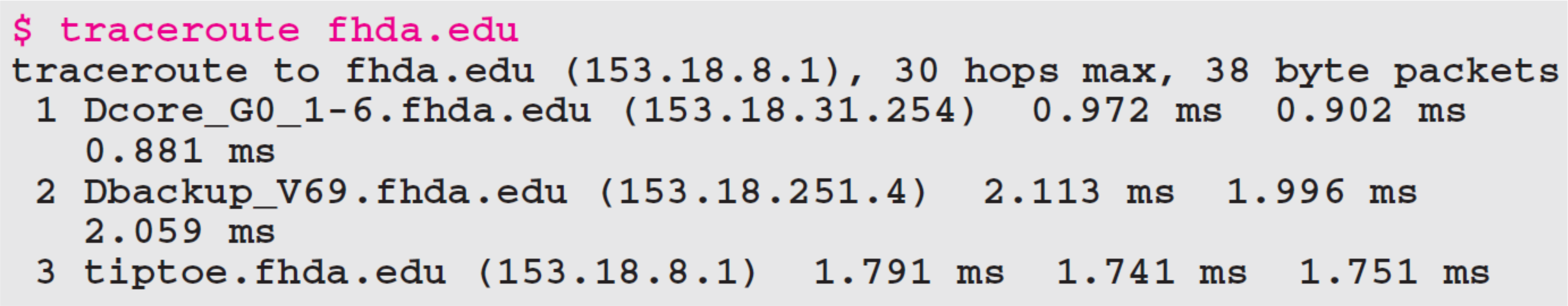
We can also use the ping utility with the -R option to implement the record route option. The result shows the interfaces and IP addresses.



```
$ 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

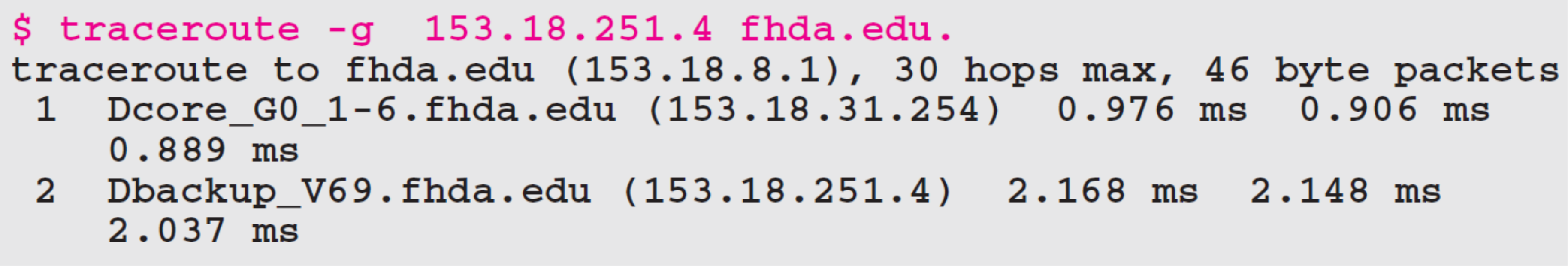
The traceroute utility can also be used to keep track of the route of a packet. The result shows the three routers visited.



```
$ 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

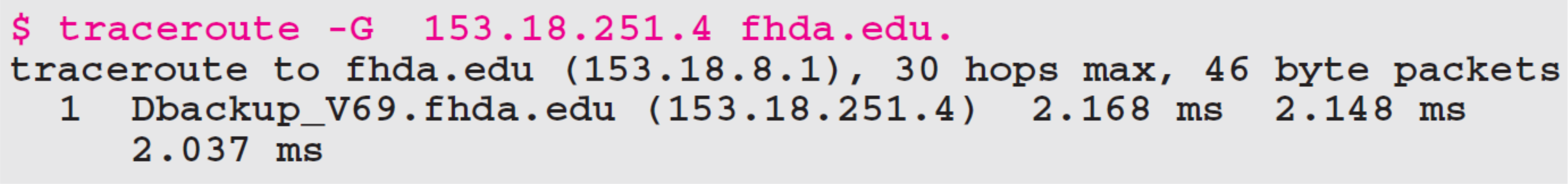
The traceroute program can be used to implement loose source routing. The -g option allows us to define the routers to be visited, from the source to destination. The following shows how we can send a packet to the fhda.edu server with the requirement that the packet visit the router 153.18.251.4.



```
$ 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*

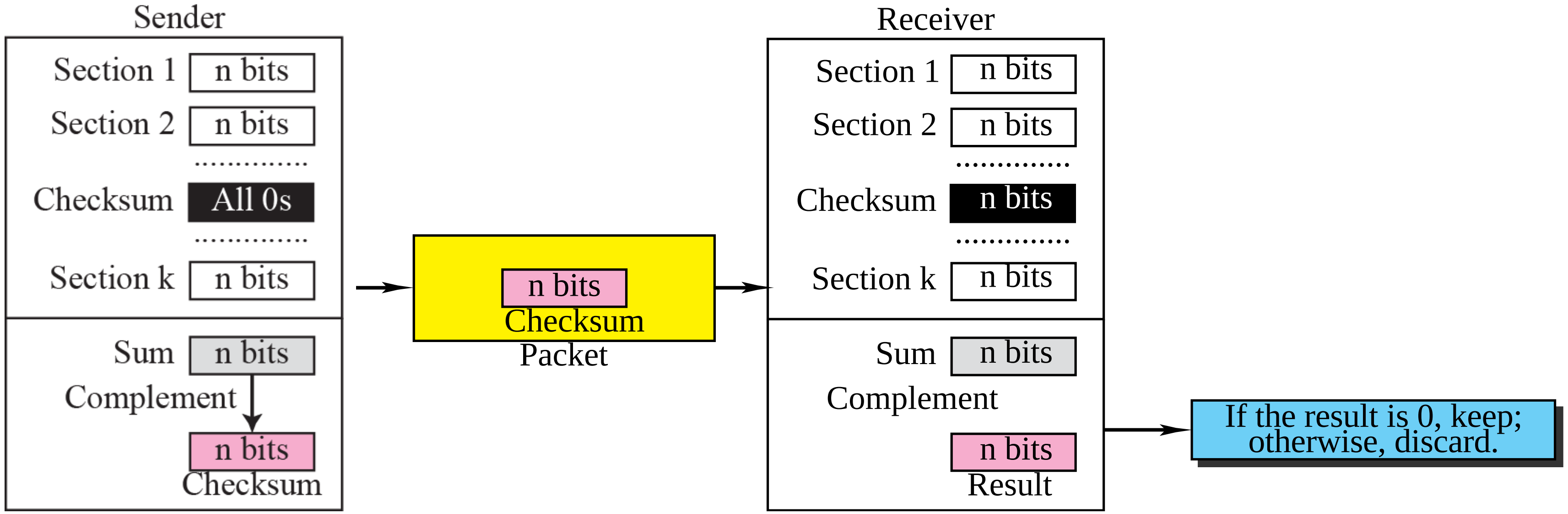
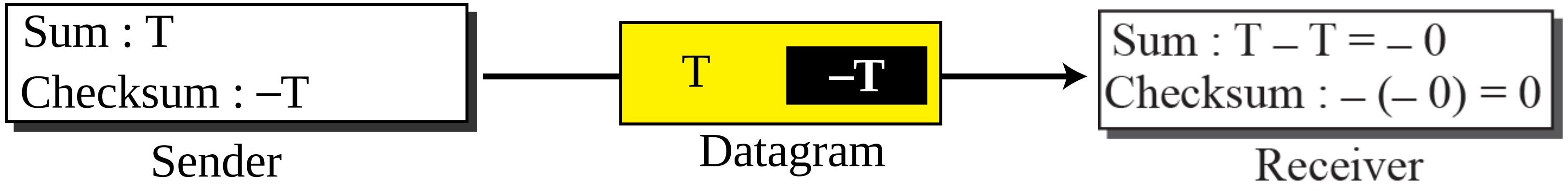


Figure 7.23 *Checksum in one's complement arithmetic*



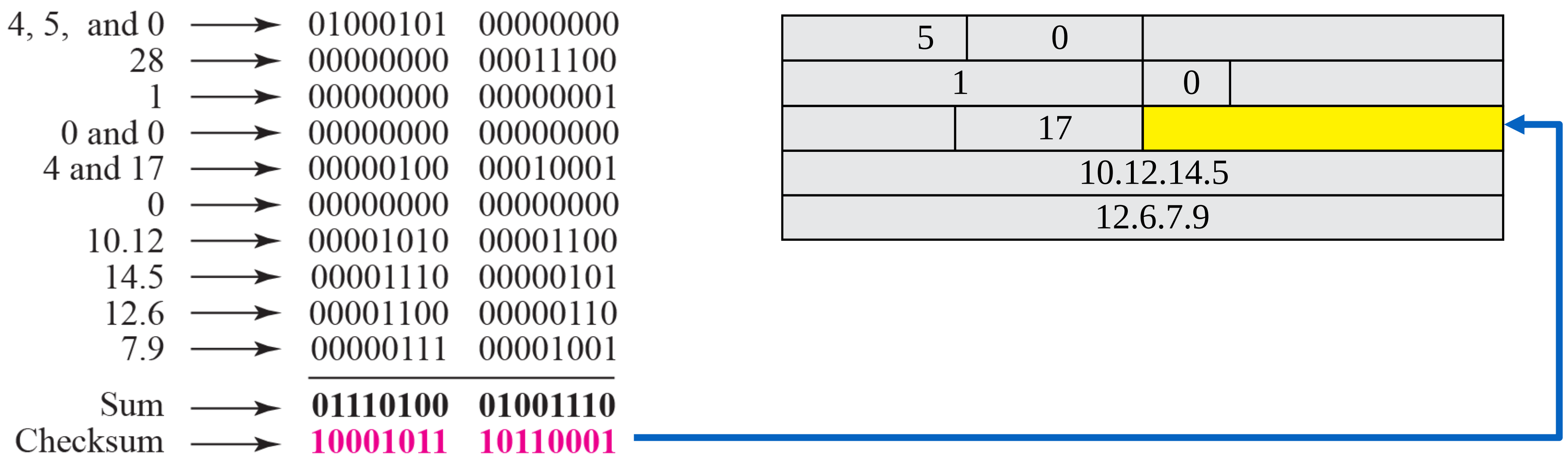
Note

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**