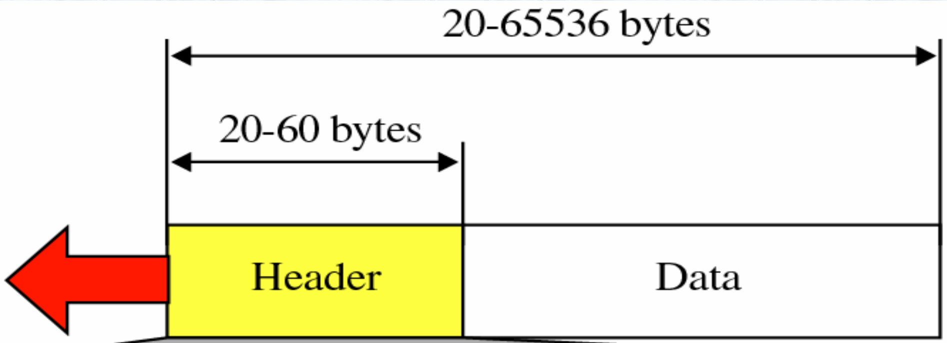
Chapter 8

Internet Protocol (IP)

CONTENTS

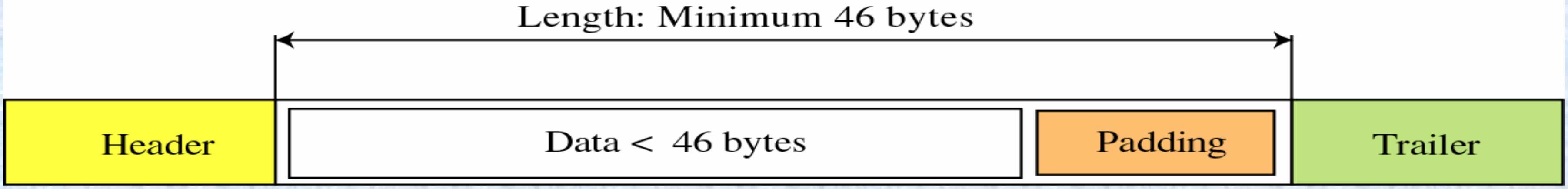
- **DATAGRAM**
- **FRAGMENTATION**
- **OPTIONS**
- **CHECKSUM**
- **IP PACKAGE**



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			
Option			

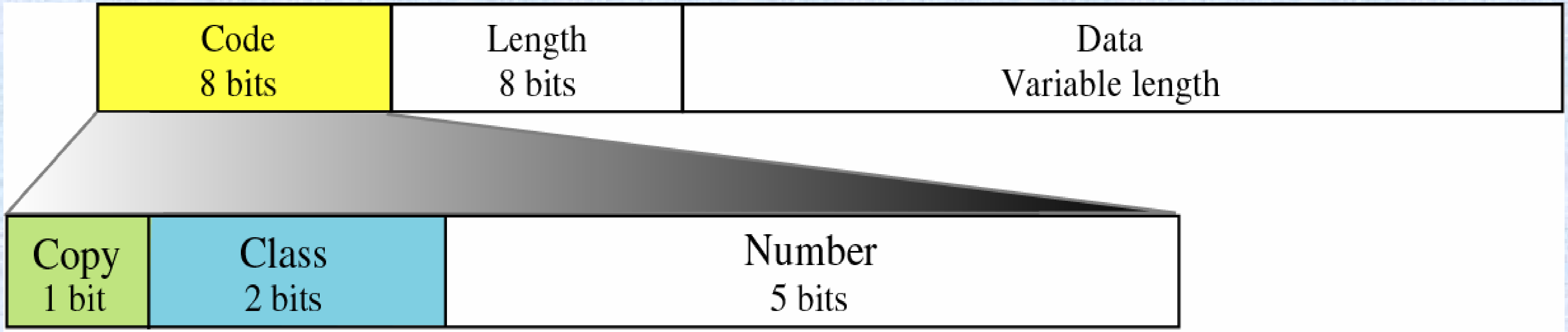
Fragmentation-related →

Fixed values



8.3 OPTIONS

- Used for network testing and debugging
- Each option follows the TLV (Type-Length-Value) format



Code 8 bits | Length 8 bits | Data Variable length

Copy 1 bit | Class 2 bits | Number 5 bits

Copy

0	Copy only in first fragment
1	Copy into all fragments

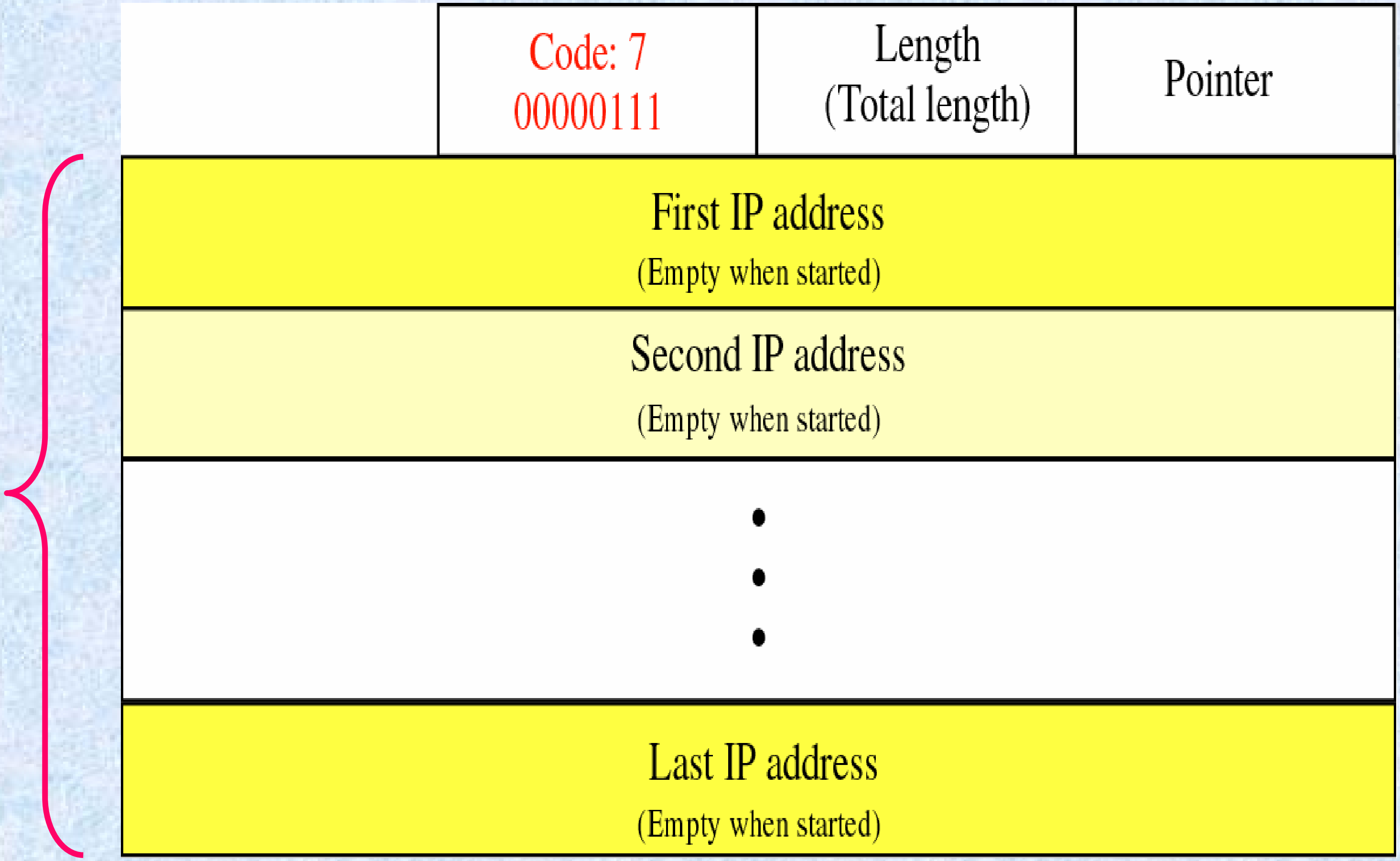
Class

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

Record route option
How did it arrive?



Code: 7
00000111

Length
(Total length)

Pointer

First IP address
(Empty when started)

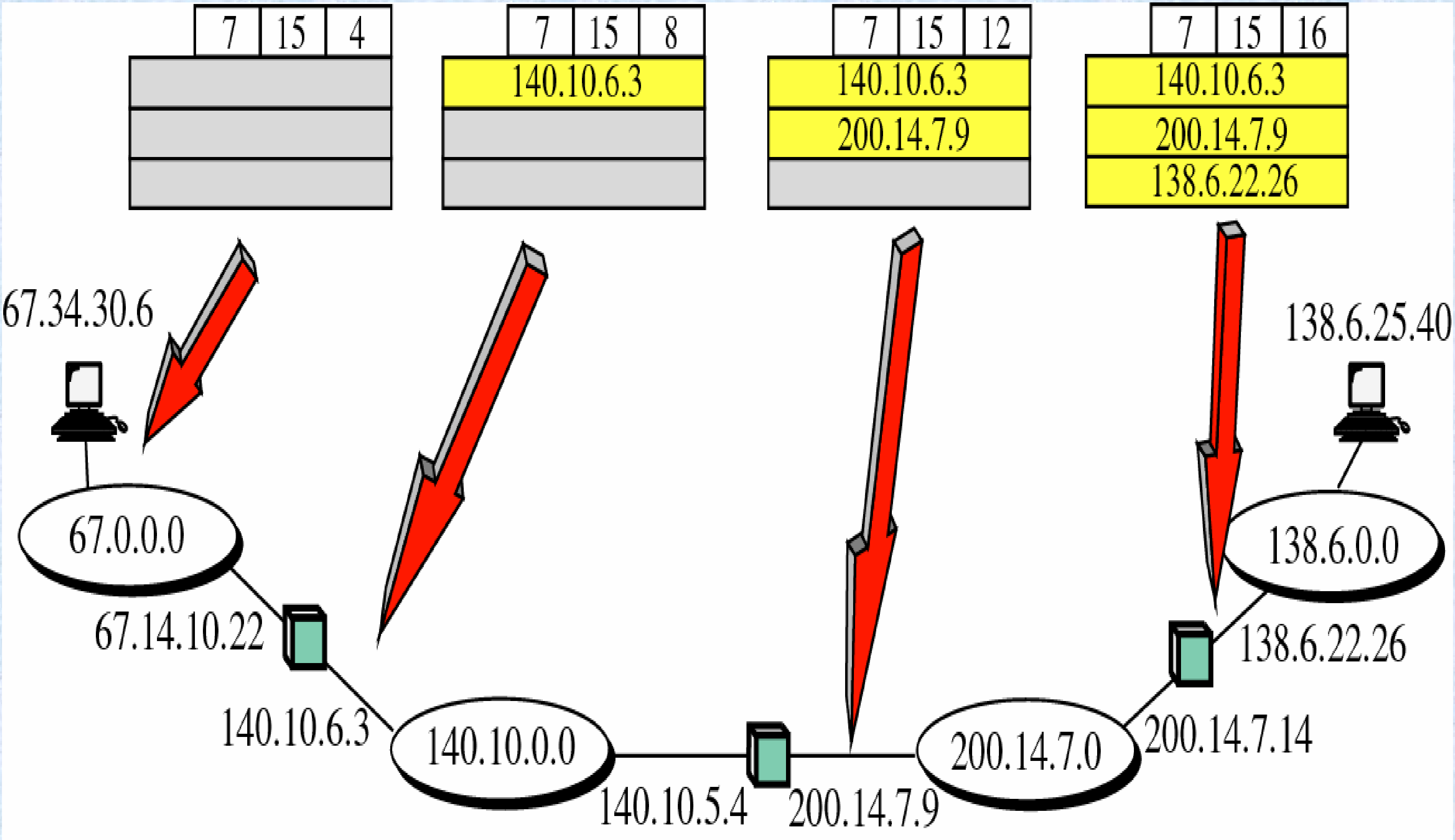
Second IP address
(Empty when started)

•
•
•

Last IP address
(Empty when started)

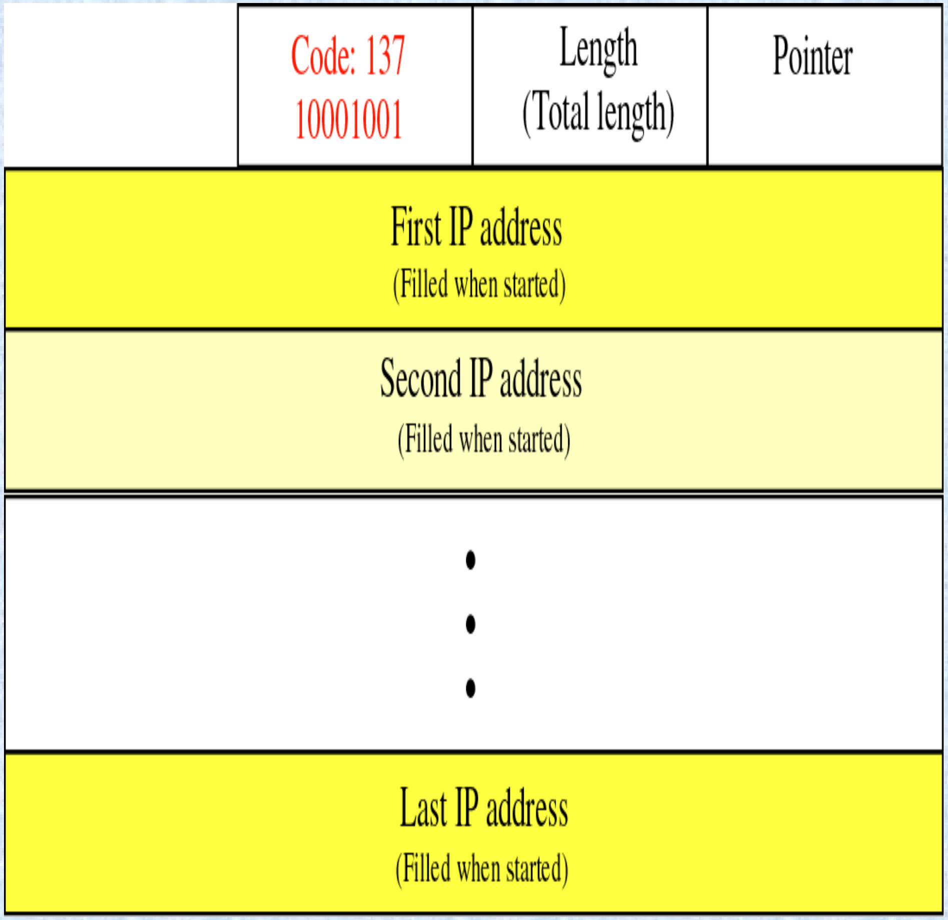
Up to 9 placeholders created by Source host

Record route concept



Strict source route option
How should it travel?

- Route strictly dictated by Source
  - For performance, reliability, testing, ..etc.
- All pre-specified routers MUST be visited, and nothing else.
  - Otherwise: discard datagram and send error message.



Code: 137
10001001

Length
(Total length)

Pointer

First IP address
(Filled when started)

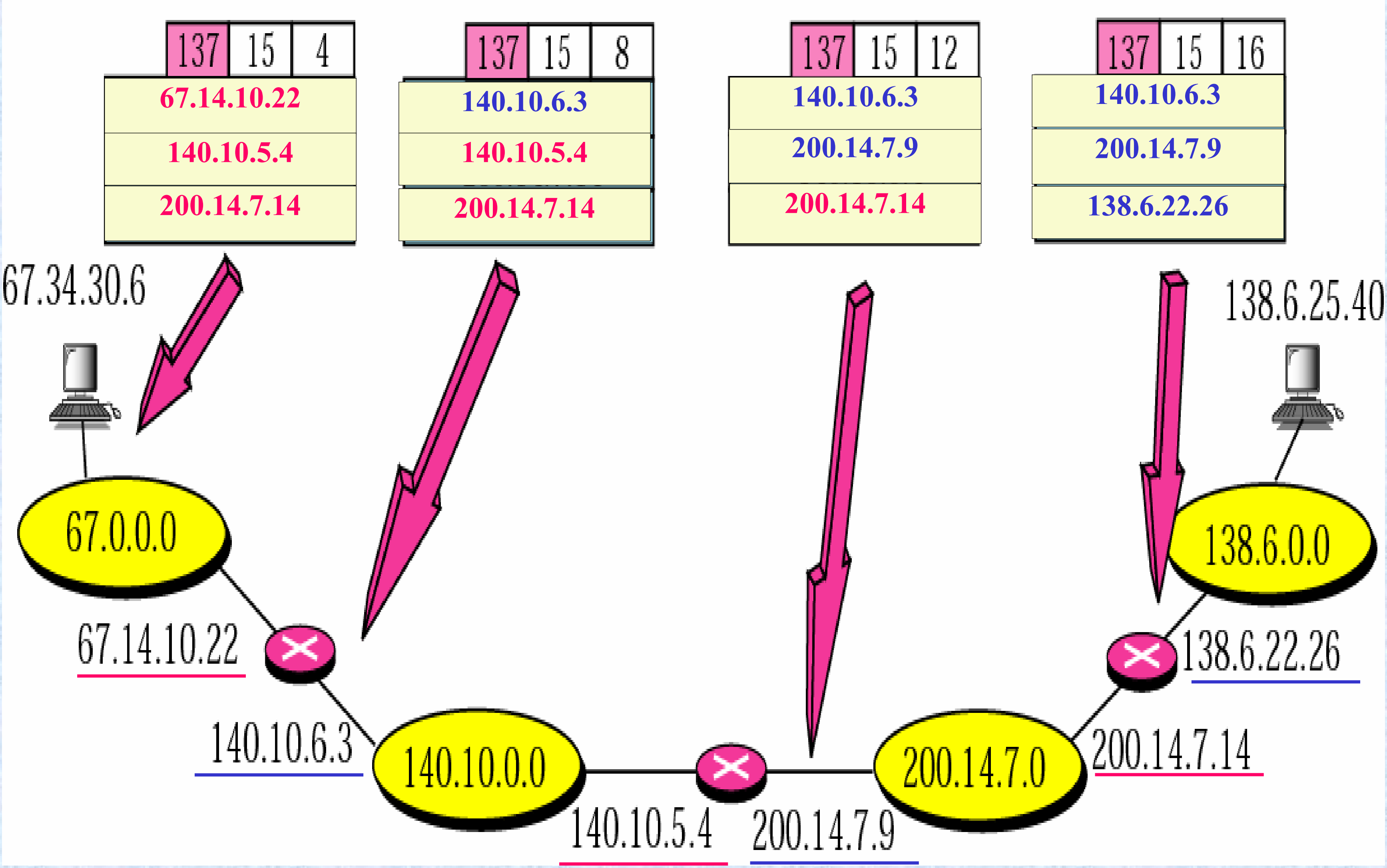
Second IP address
(Filled when started)

•
•
•

Last IP address
(Filled when started)

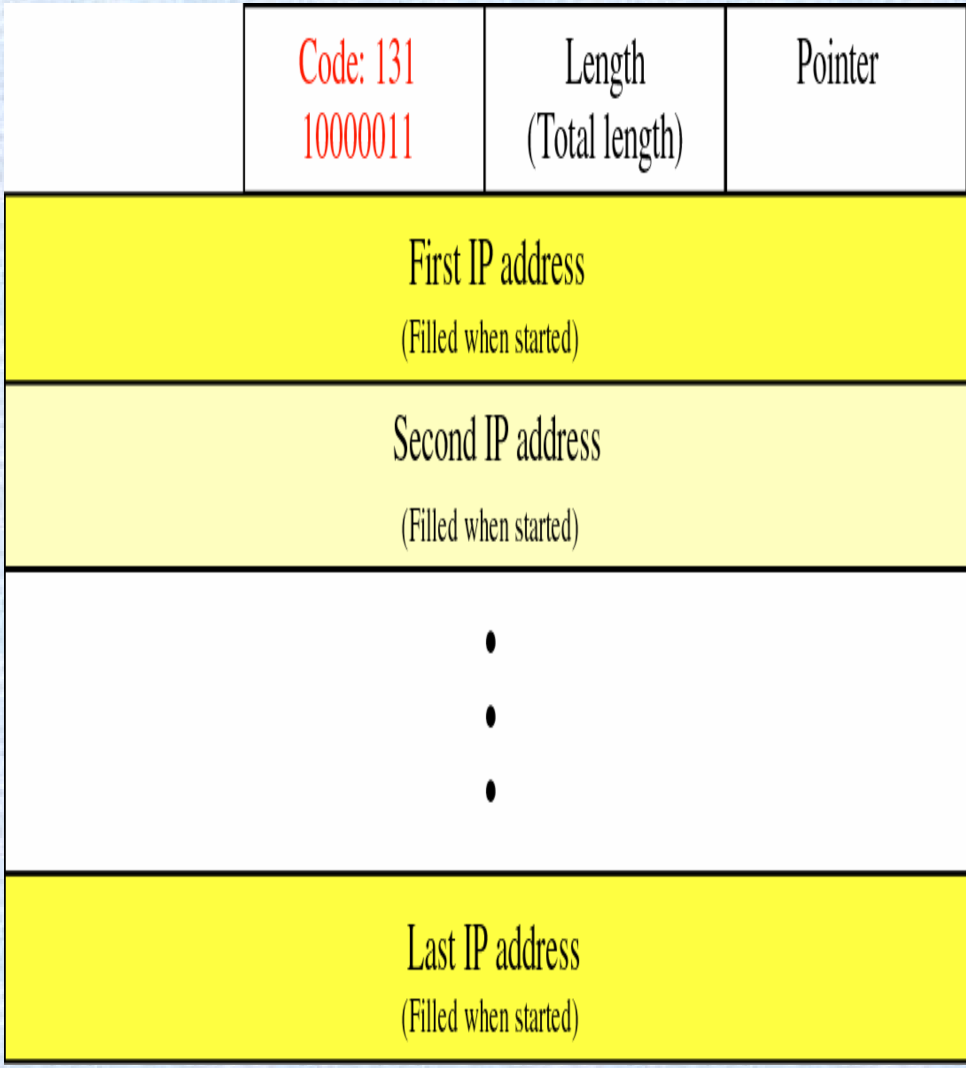
Strict source route example

Figure after corrections



Loose source route option

- Minimum portion of Route is dictated by Source
- All pre-specified routers MUST be visited, and possibly others as well.
  - Otherwise: discard datagram and send error message.



Code: 131
10000011

Length
(Total length)

Pointer

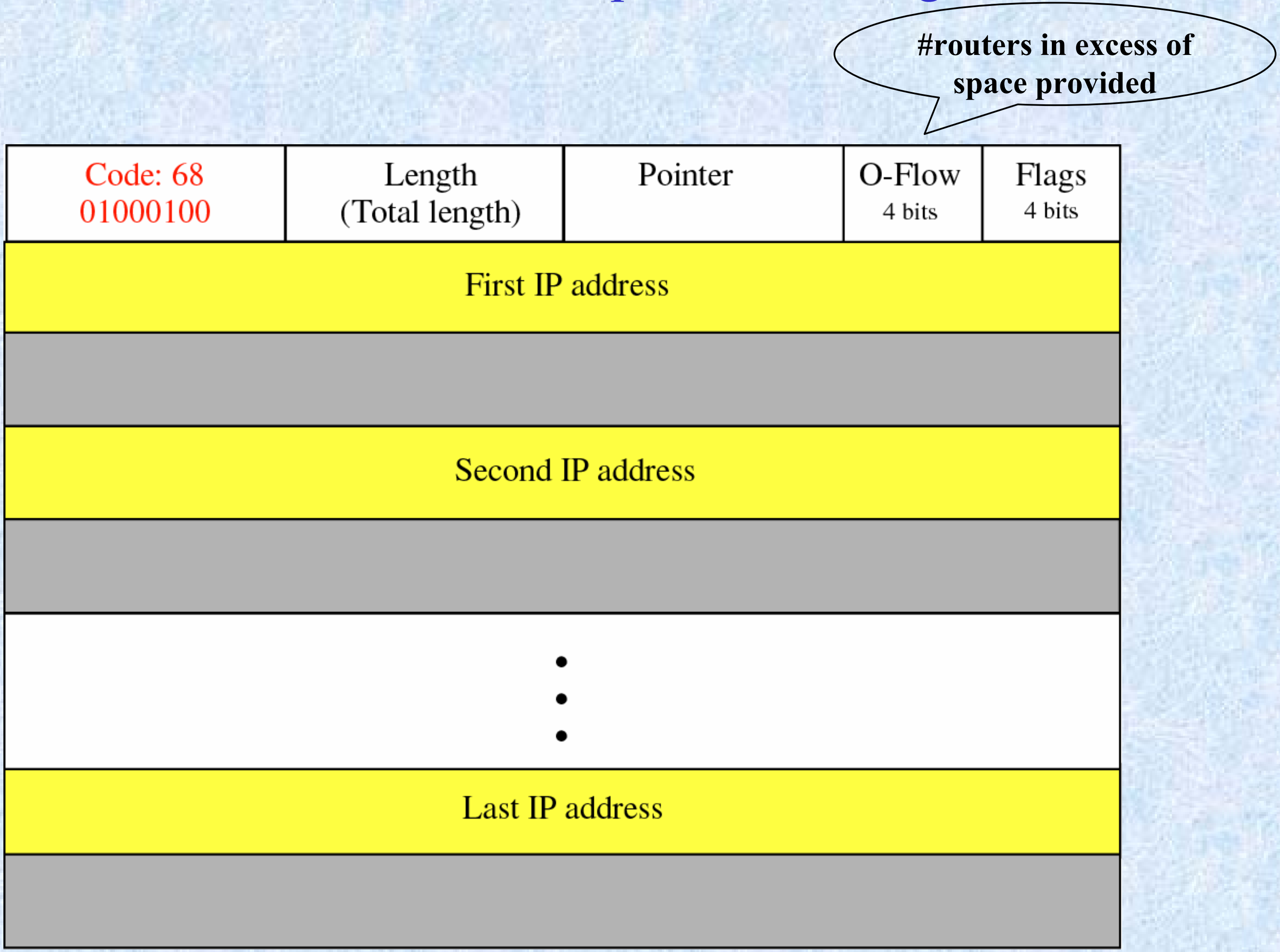
First IP address
(Filled when started)

Second IP address
(Filled when started)

•
•
•

Last IP address
(Filled when started)

Timestamp option
When did each visited router process a datagram?



#routers in excess of space provided

Code: 68
01000100

Length
(Total length)

Pointer

O-Flow
4 bits

Flags
4 bits

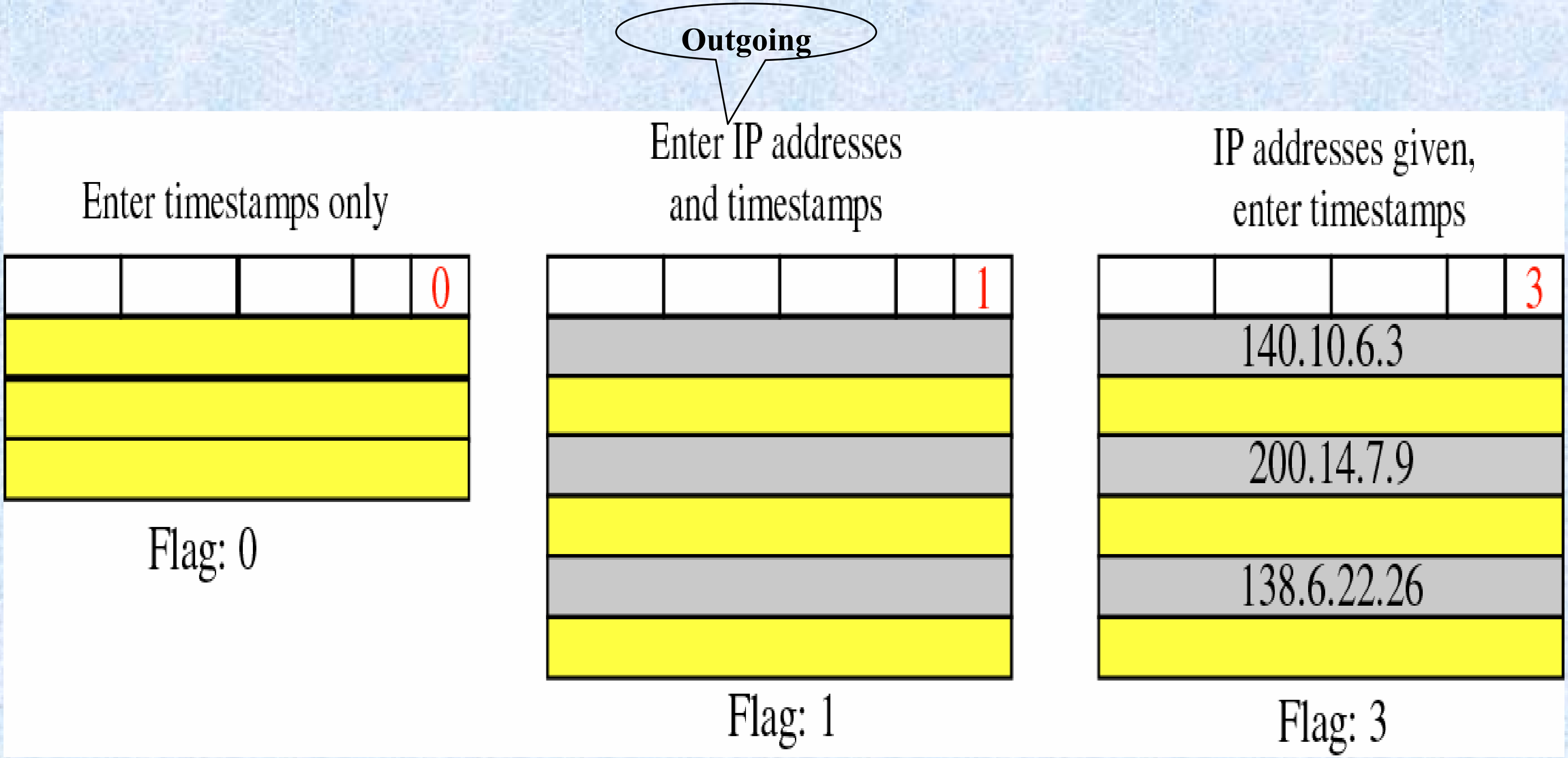
First IP address

Second IP address

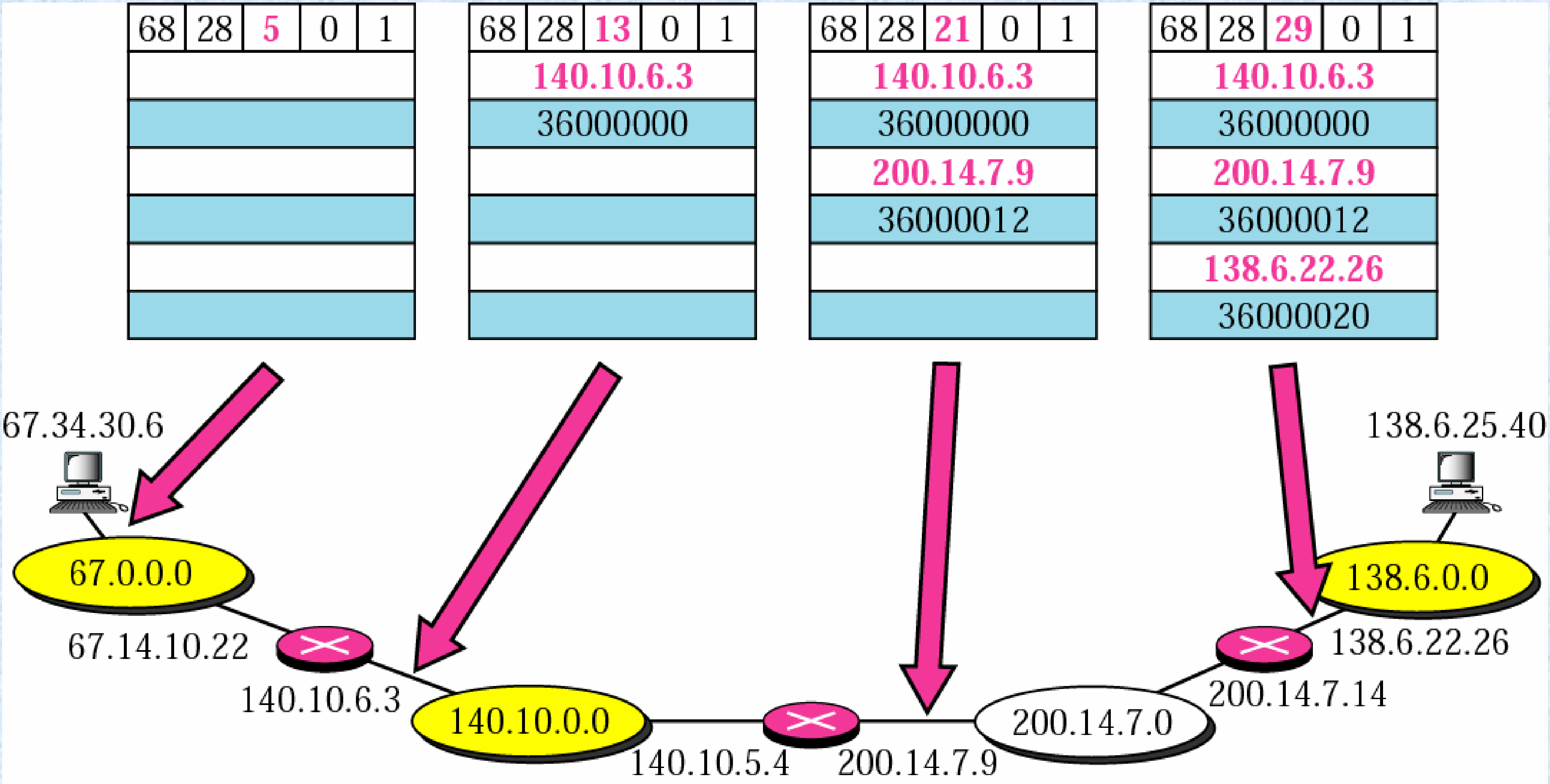
•
•
•

Last IP address

Use of flag in timestamp



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 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.

No operation:	Code is **0**0000001; no copy.
End of option:	Code is **0**0000000; no copy.
Record route:	Code is **0**0000111; no copy.
Strict source route:	Code is **1**0001001; copied.
Loose source route:	Code is **1**0000011; copied.
Timestamp:	Code is **0**1000100; no copy.

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.

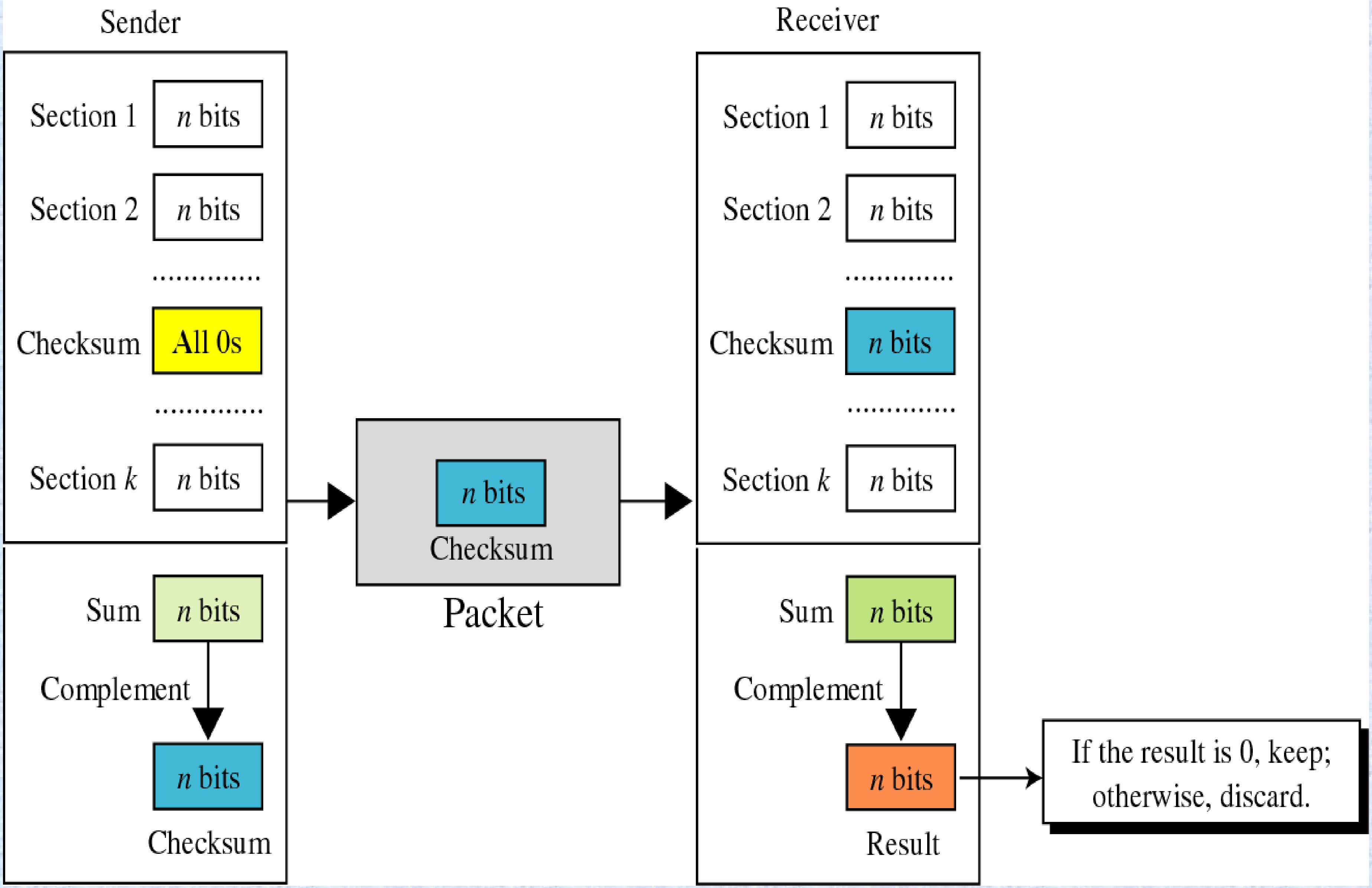
No operation:	Code is 00000001; control.
End of option:	Code is 00000000; control.
Record route:	Code is 00000111; control.
Strict source route:	Code is 10001001; control.
Loose source route:	Code is 10000011; control.
Timestamp:	Code is 01000100; debugging

8.4 CHECKSUM

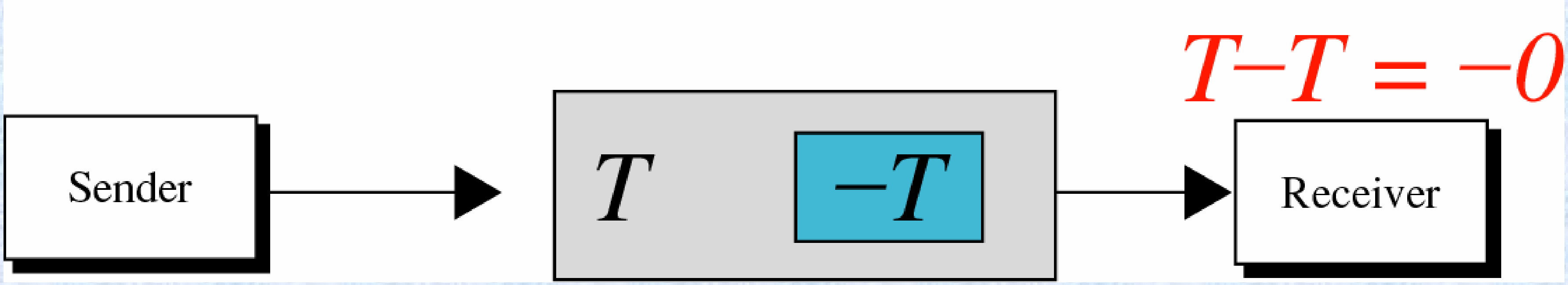
To create the checksum the sender does the following:

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

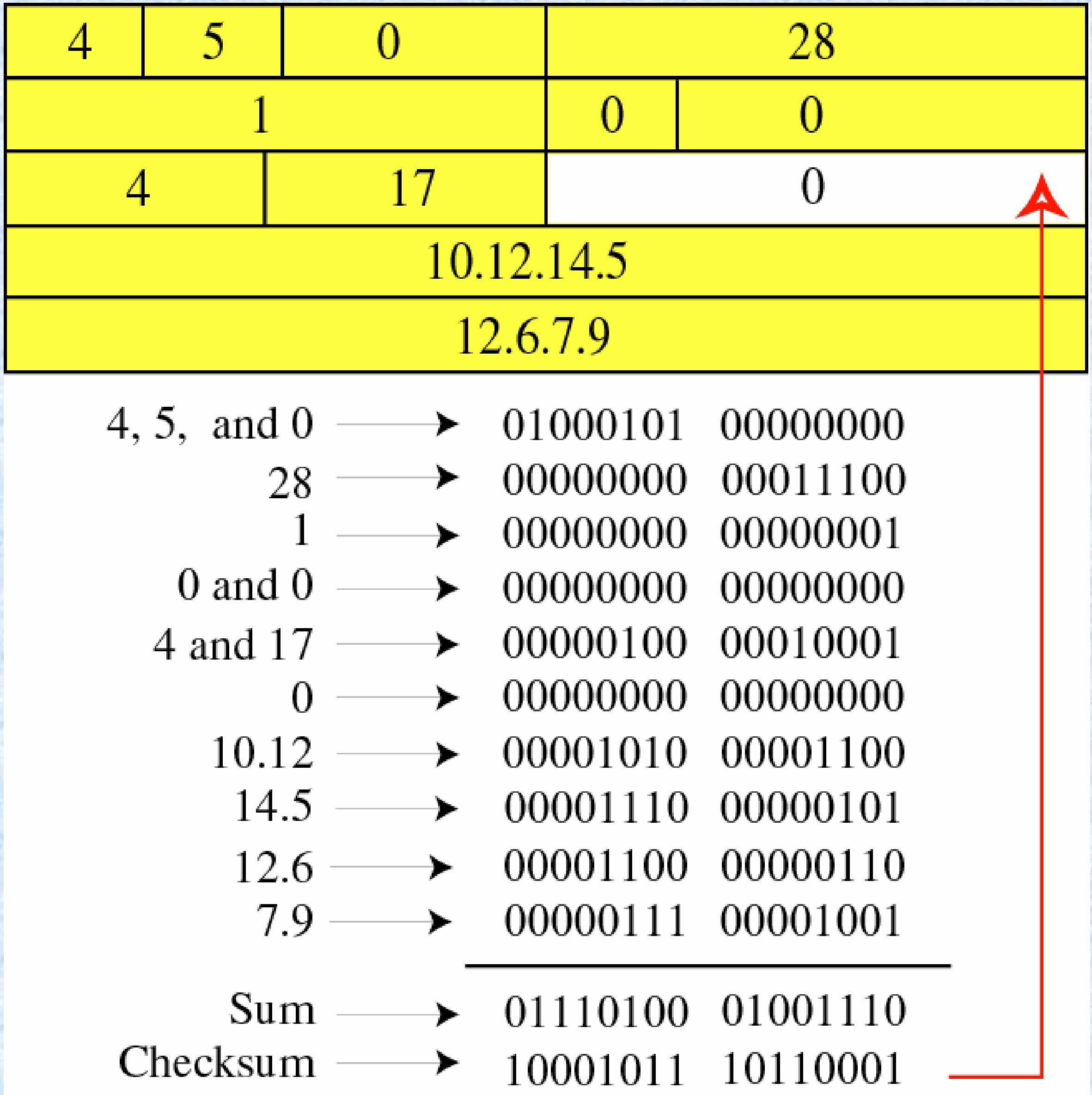
Checksum concept



Checksum in one's complement arithmetic



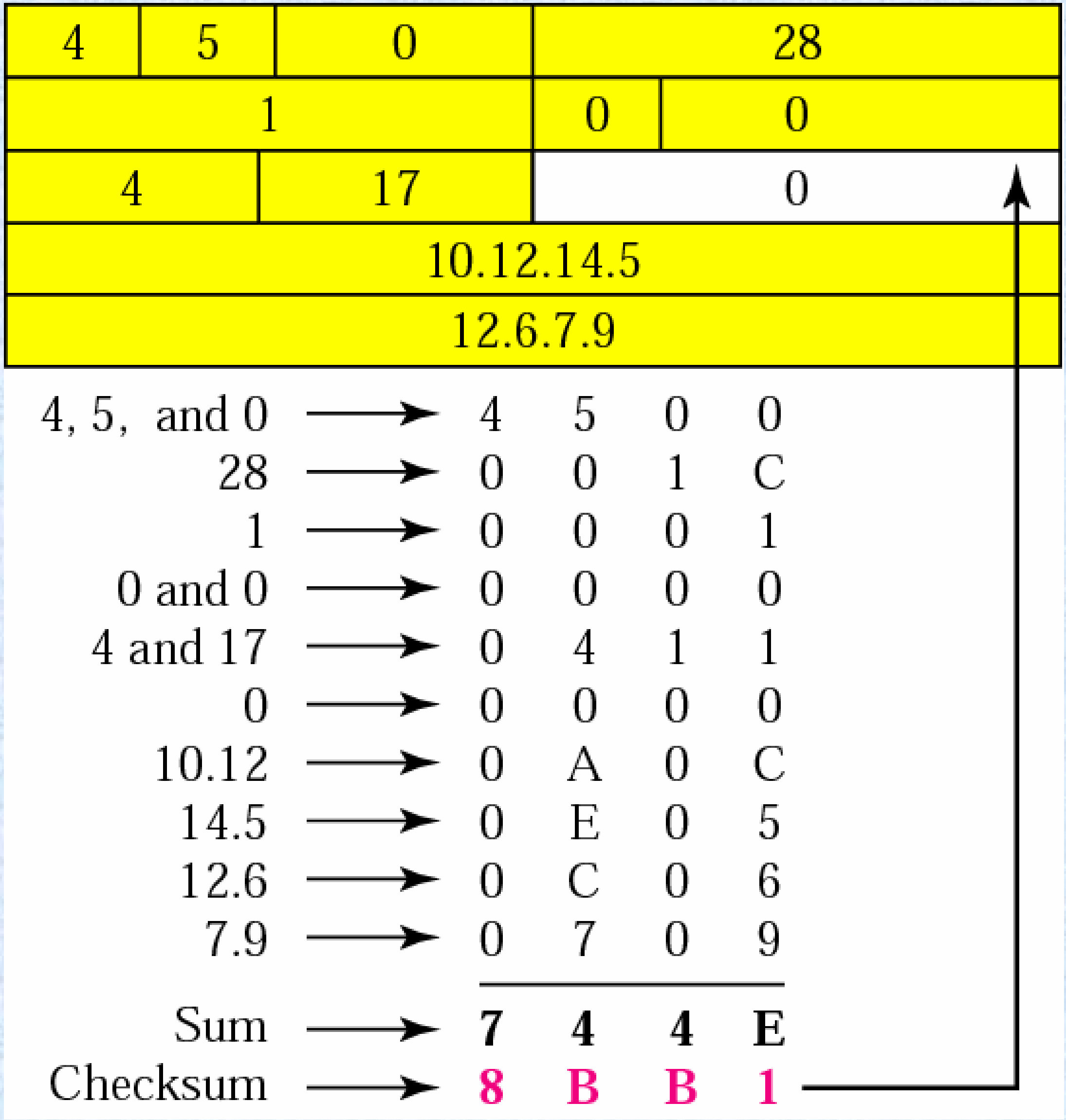
Example of checksum calculation in binary



4	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

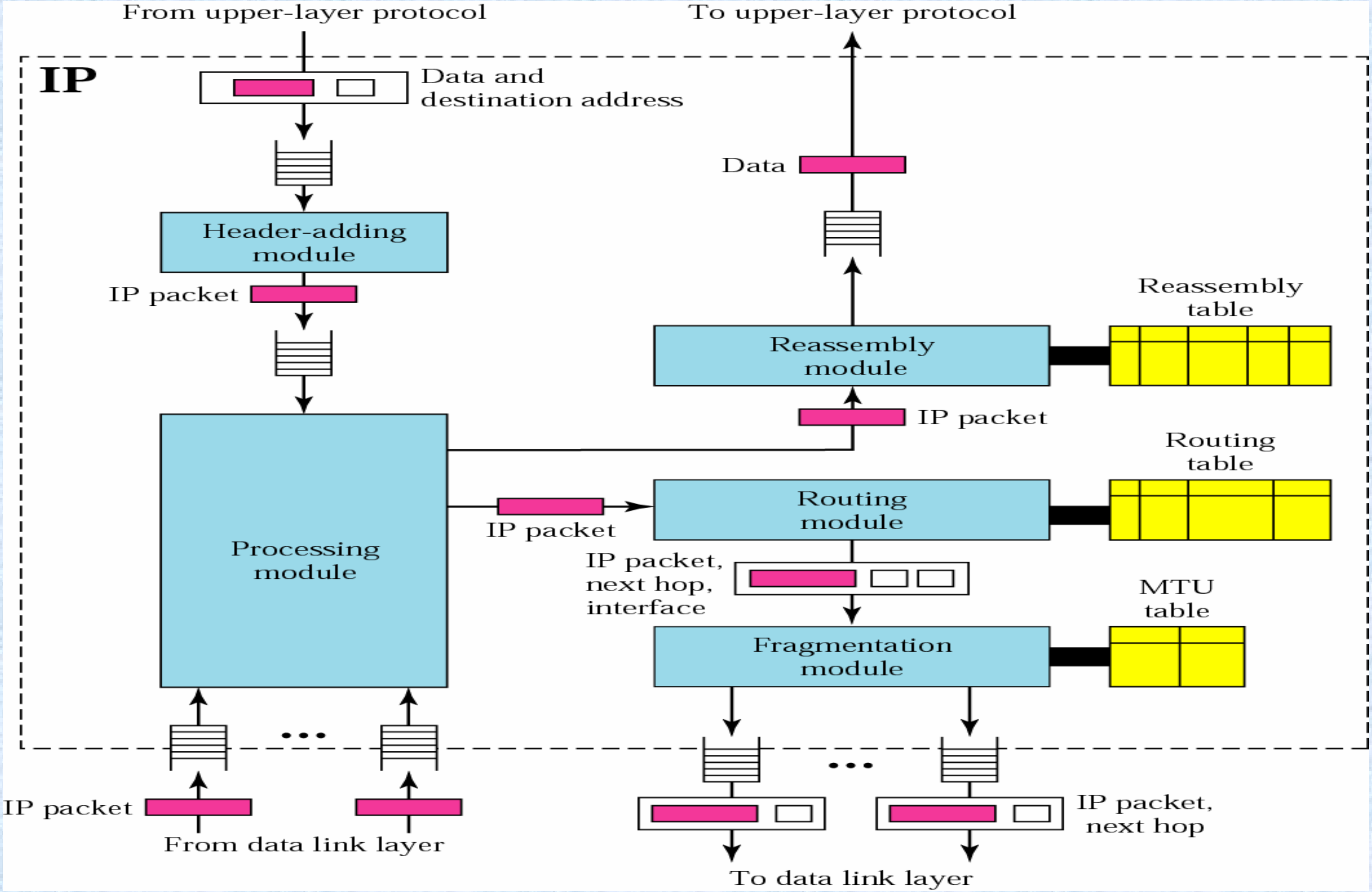
Example of checksum calculation in hexadecimal



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**

8.5 IP PACKAGE



MTU table

Interface Number	MTU
....	....
....	....
....	....

Reassembly table

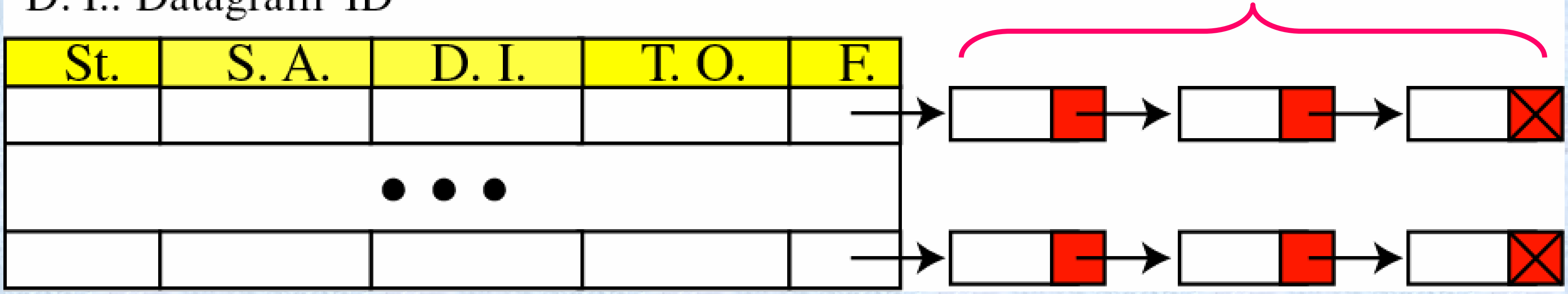
St.: State

S. A.: Source address

D. I.: Datagram ID

T. O.: Time-out

F.: Fragments



Sorted Linked-List of fragments of the same datagram

St.	S. A.	D. I.	T. O.	F.
• • •				