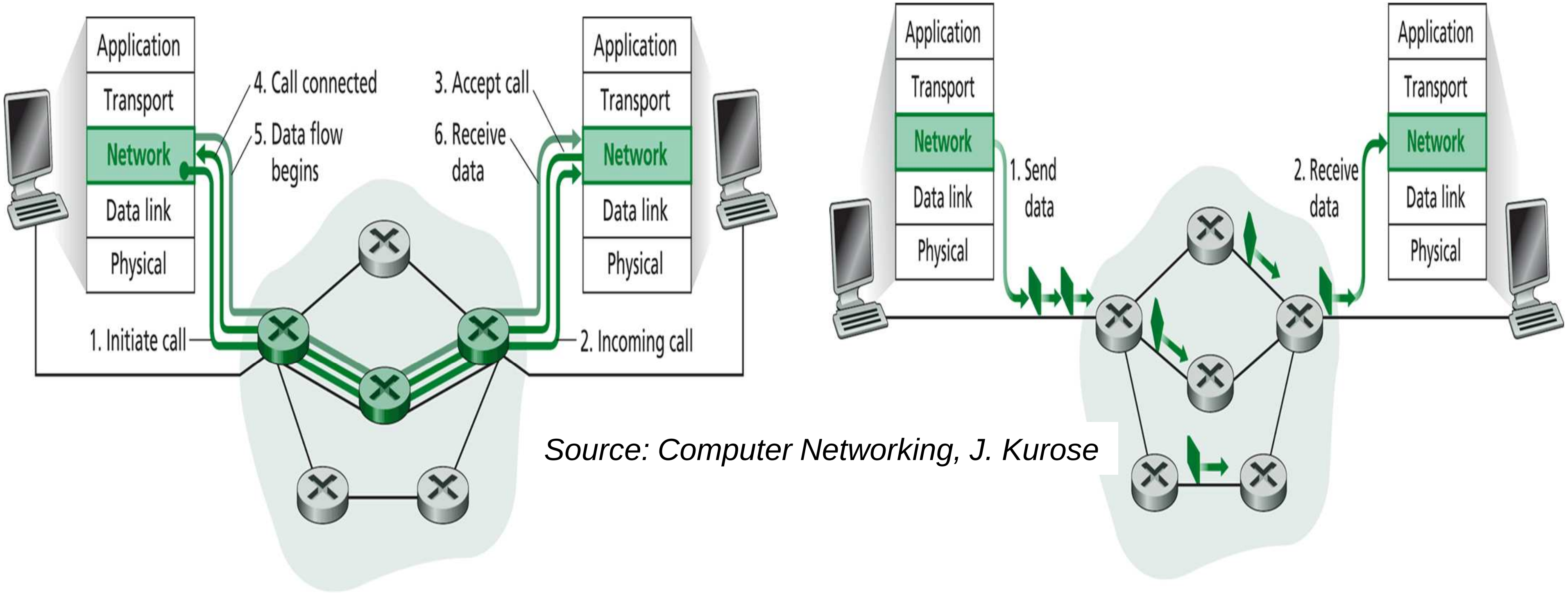
IP (Internet Protocol)

- Offered Services
- Packet Format
- ICMP

IP Communication Service

- *Connectionless*
  - *packet-oriented (or datagram) paradigm*
  - Two packets meant for the same destination may “be handled” in different ways (just like 2 letters in the postal system)
- Low Reliability
  - *Best-effort* delivery
  - Similar to the snail mail service

Packet vs Virtual Circuit



Source: Computer Networking, J. Kurose

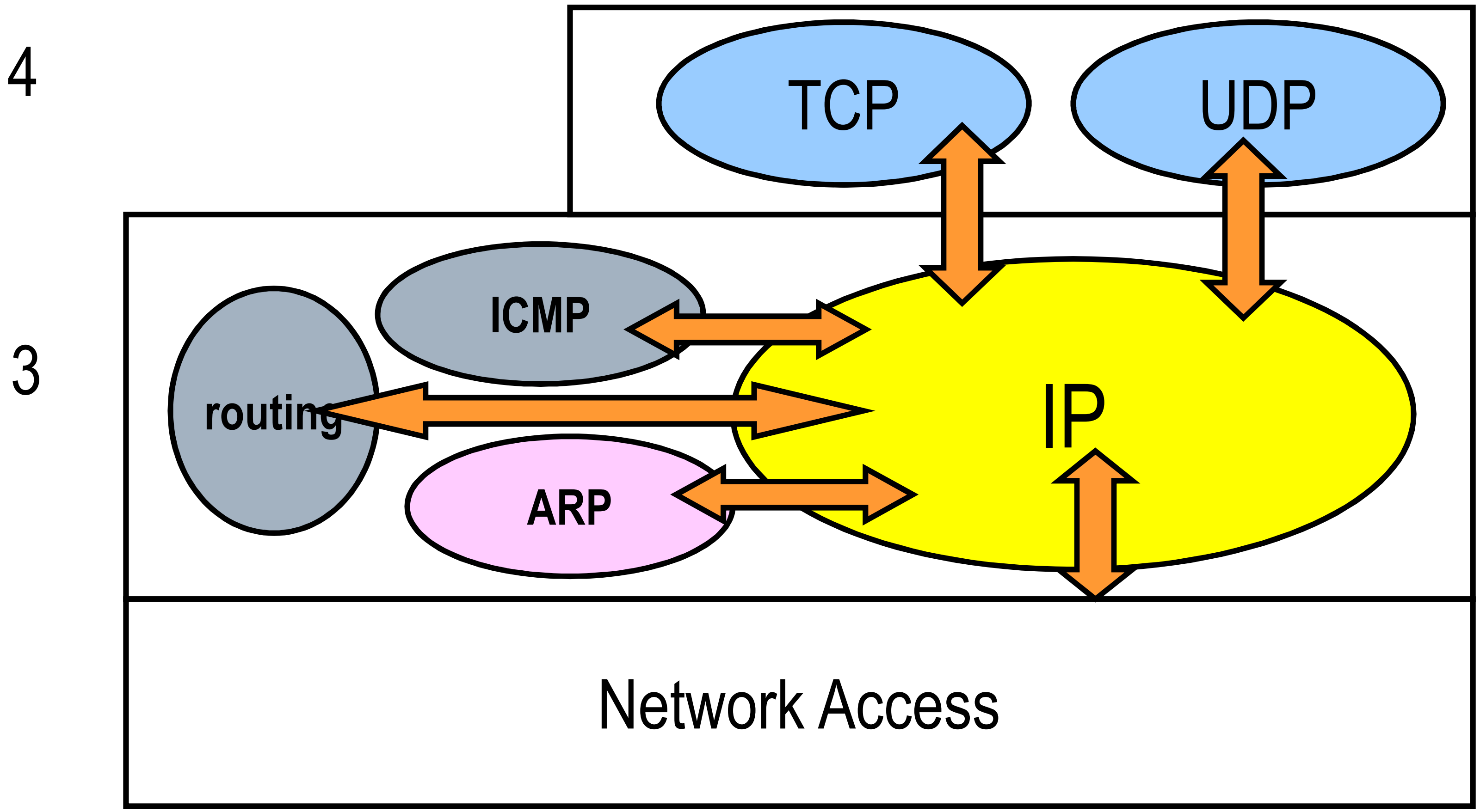
ATM, X25, Frame Relay

IP Approach

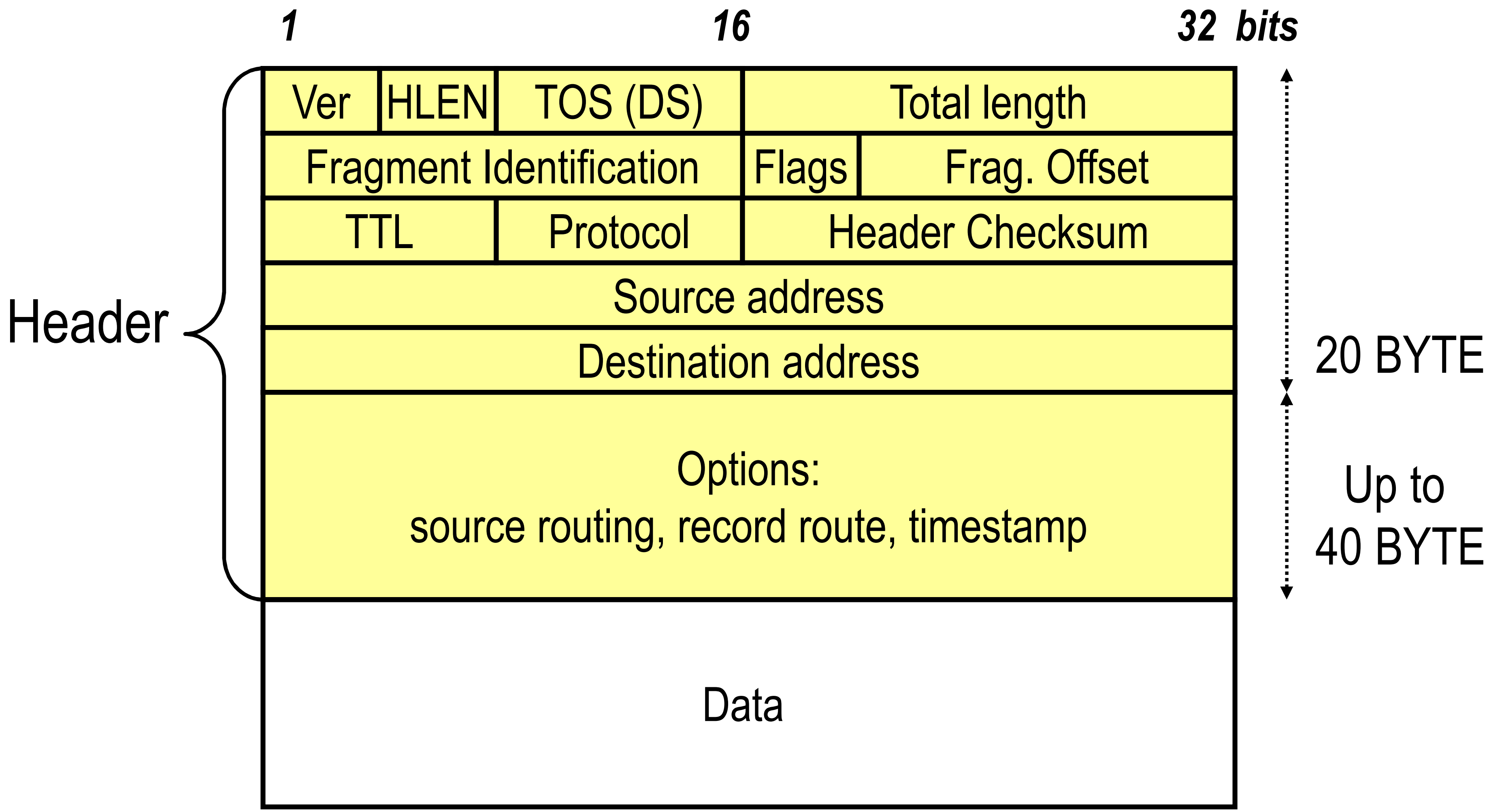
IP Services

- Addressing: to assign a unique and universally valid address
- Fragmentation/De-fragmentation: according to the network access requirements

IP Layer



IP Packet Format

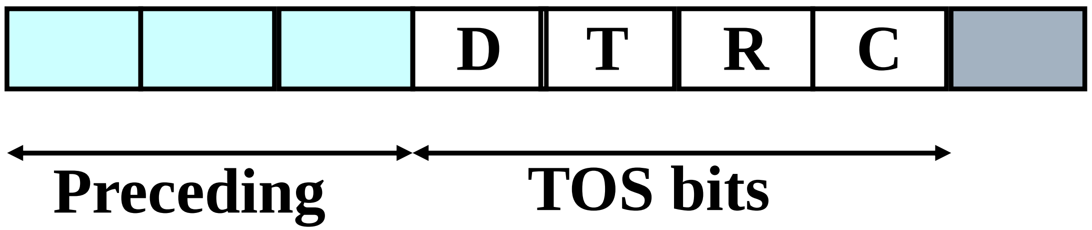


IP Header Fields

- Ver (4 bit):
  - *Version* of the protocol: IPv4, IPv6. If the router does not support the specified version the packet is dropped
- HLEN (4 bit)
  - *header length*: expressed in 32 bits words (max 64 byte)
- Total length (16 bit):
  - Measured in bytes: maximum length 2^{16}=65536; HLEN and Total length can be used to calculate the dimension of the payload (useful if lower layers implement padding)

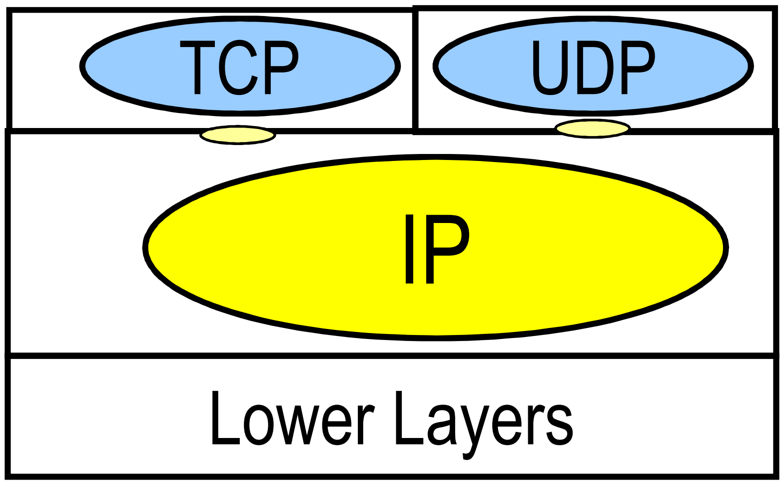
IP Header Fields

- TOS *type of service* (8 bit)
  - Recently changed into *Differentiated Services* field. Used to handle priorities in the router queues and to provide QoS



Protocol Field

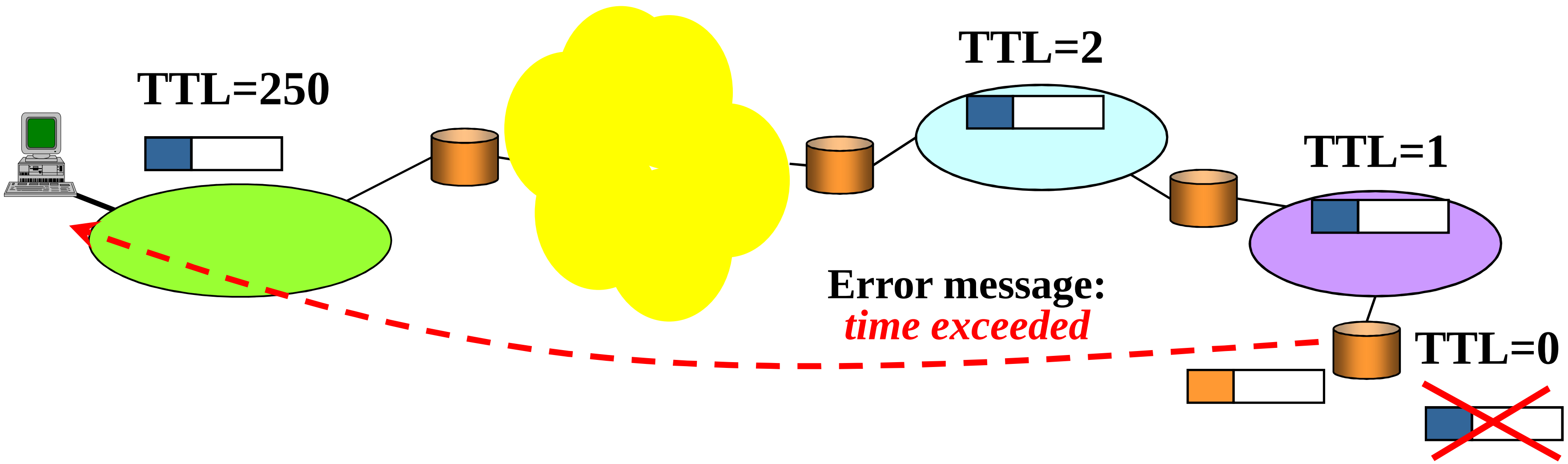
- Identifies the upper layer protocol
- Multiple upper layer protocols can use IP (multiplexing)
- The field identifies the SAP (*Service Access Point*) between IP and the upper layer protocol



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

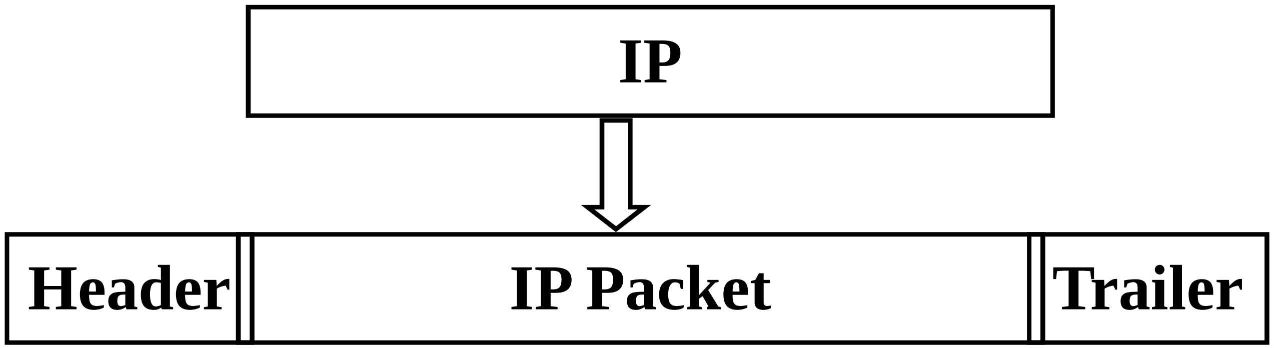
Time To Live (TTL)

- *TTL* is set by the source of a packet and is decremented by each router managing the packet
- If the TTL goes to zero before reaching the destination an error message towards the source is generated
- *Time-out* on packet validity



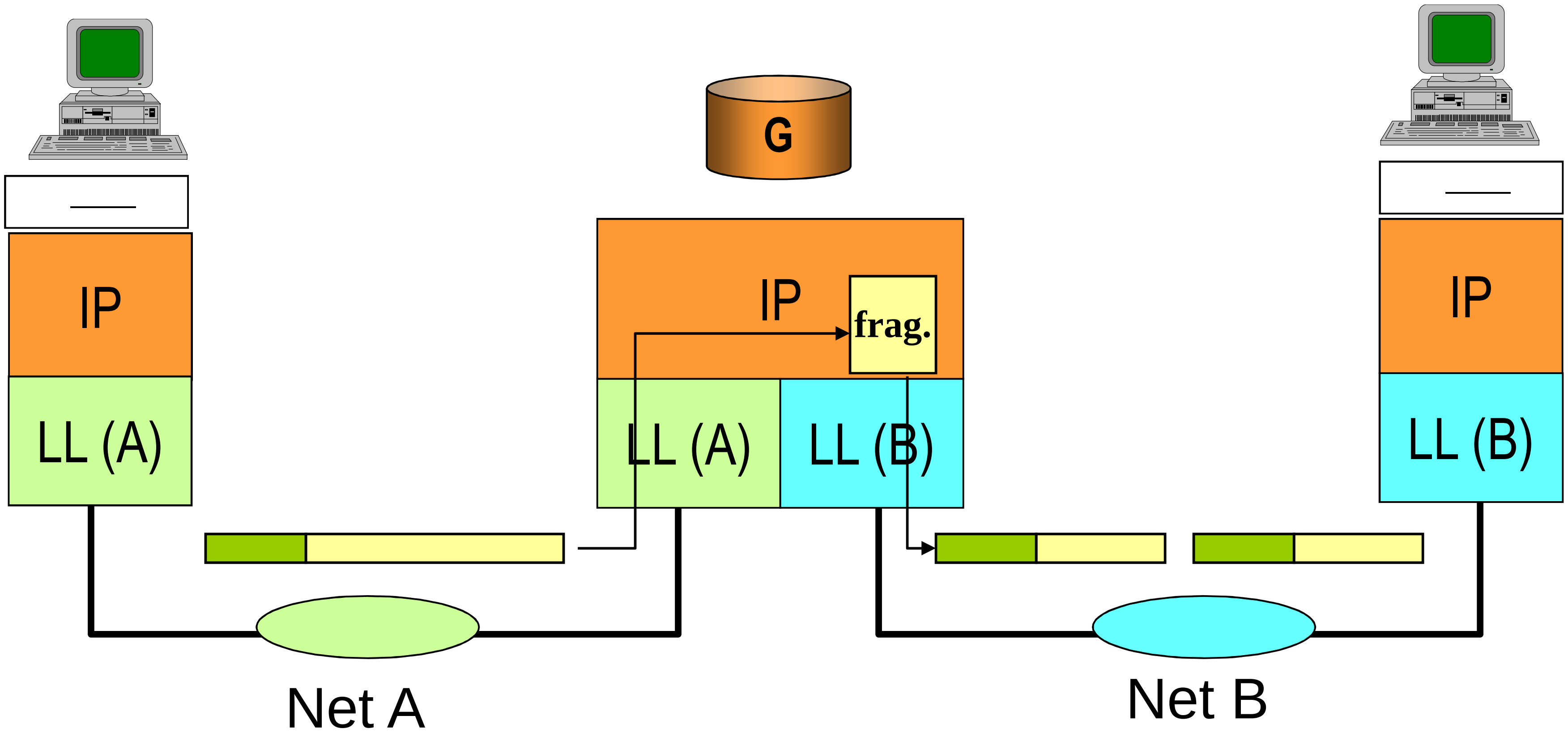
Fragmentation (1)

- Many Network Access Protocols require a maximum dimension of the frames (*Maximum Transfer Unit, MTU*) which is much lower than the IP packet maximum length (65536 bytes)



Protocol	MTU (byte)
Token Ring 16Mb/s	17914
FDDI	4352
Ethernet	1500

Fragmentation (2)



Fragmentation (3)

- Before passing down the packet IP splits it up into fragments with the corresponding headers
- A fragment can be further fragmented along its path
- The fragments are composed at the destination only (different fragments of the same packet may follow different paths)
- Fields *Identification, Flags and Frag. Offset* handle the fragmentation process

Fields for Fragmentation (1)

- Identification (16 bits)
  - Identifies all the fragments of the same packet. Chosen by the first fragmenting entity
- Frag. Offset (13 bits)
  - The bytes of the original packet are numbered from 0 to the packet length. *Frag. Offset* gives the number of the first byte in the fragment (counted as words of 8 bytes each)
  - *example:* a packet has 2000 bytes and is fragmented into two chunks of 1000 bytes; the first fragment has Frag *Offset* equal to 0, the second equal to 1000/8

Fields for Fragmentation (2)

- Flags

-	D	M

  - bit M (*More*) is set to 0 in the last fragment
  - bit D (*Do not fragment*) is set to 1 to switch off the fragmentation
    - In this case, if fragmentation is required the packet is dropped and an error message is generated

Fragmentation in practice

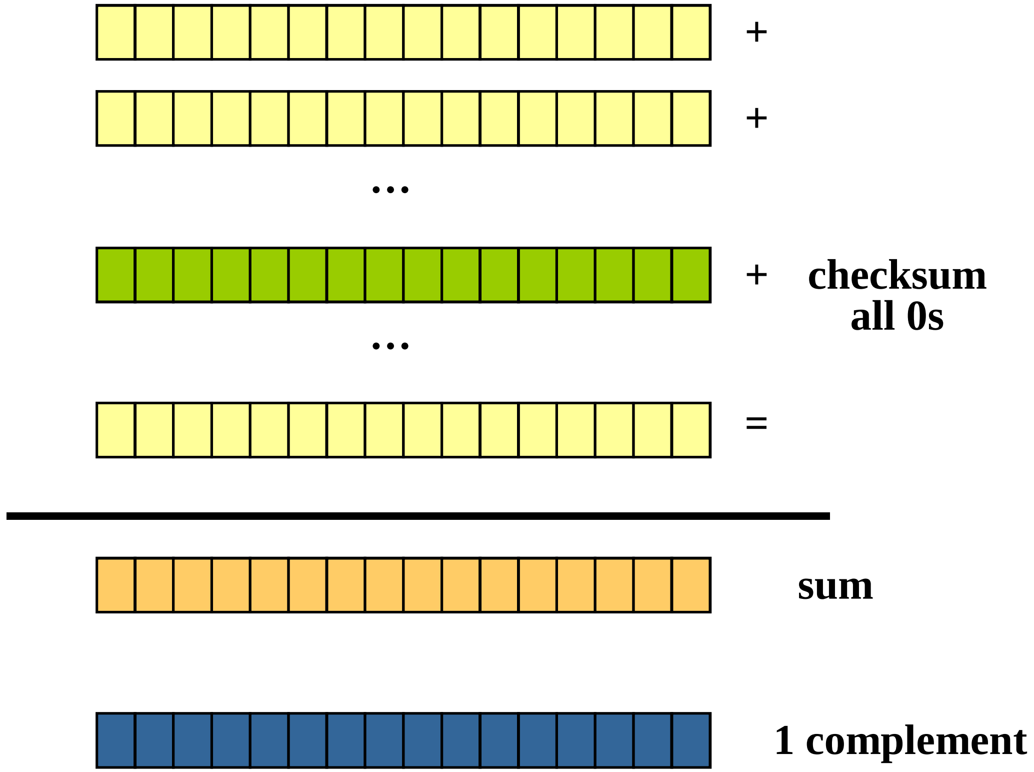
- High processing overhead due to fragmentation
- Fragmentation is skipped whenever possible. Hard limitation on the packet length.
- Underlying technologies can handle frames (MTU) of 576 bytes at least
- The transport layer segments length is set to 536 byte (+20byte TCP + 20byte IP)
- The most of application layer software works with message length in the range 512-536 byte.

Checksum: integrity check

- Redundant information in the IP header for error control
- The *checksum* field is computed by the transmitter (16 bit) and inserted into the header
- The receiver repeats the same computation on the received packet (*checksum field included*)
- If the result is positive it processes the packet otherwise it drops it

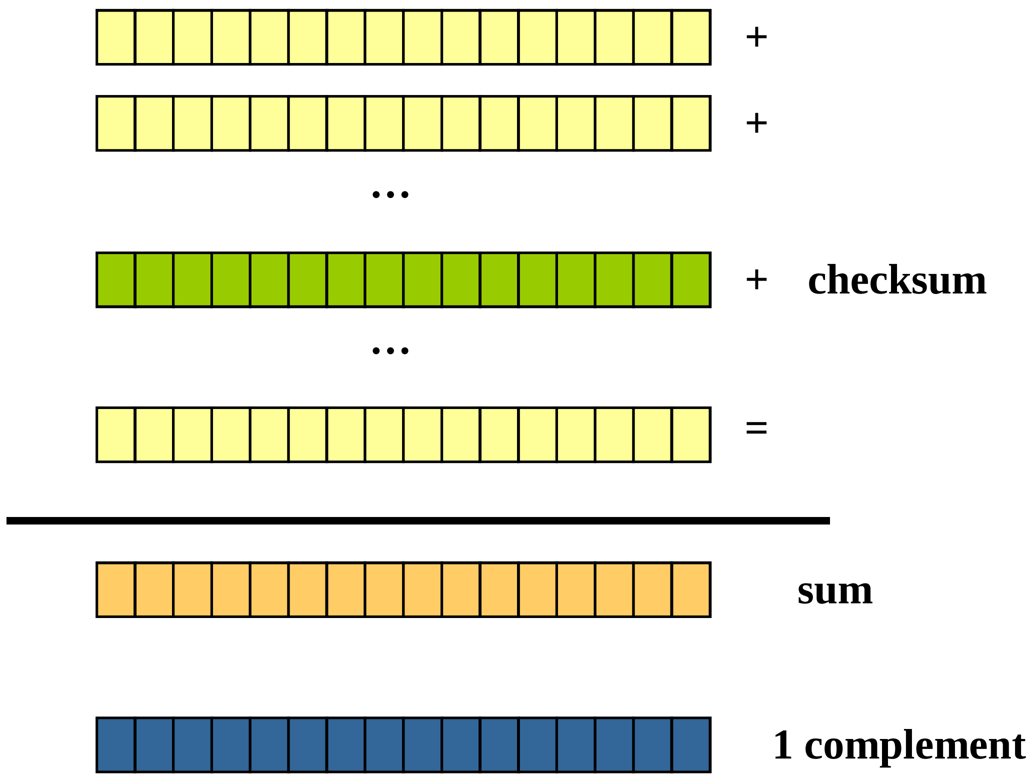
Checksum: transmitter's side

- The header is divided into 16 bits chunks
- The *Checksum field is set to 0*
- All the chunks are summed up
- The 1-complement of the result is inserted in the checksum field



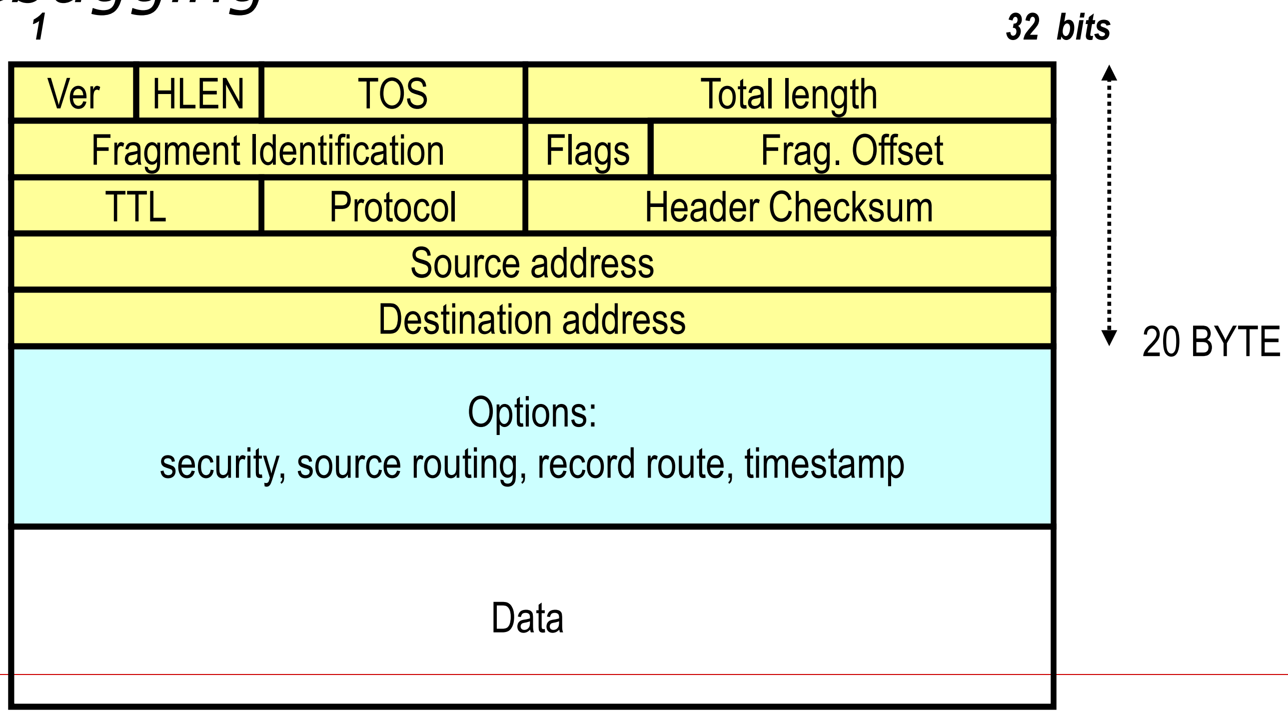
Checksum: receiver's side

- The header is divided into 16 bits chunks
- All the chunks are summed up
- The 1-complement of the result is taken
  - If all 0s the packet is processed
  - Otherwise is dropped

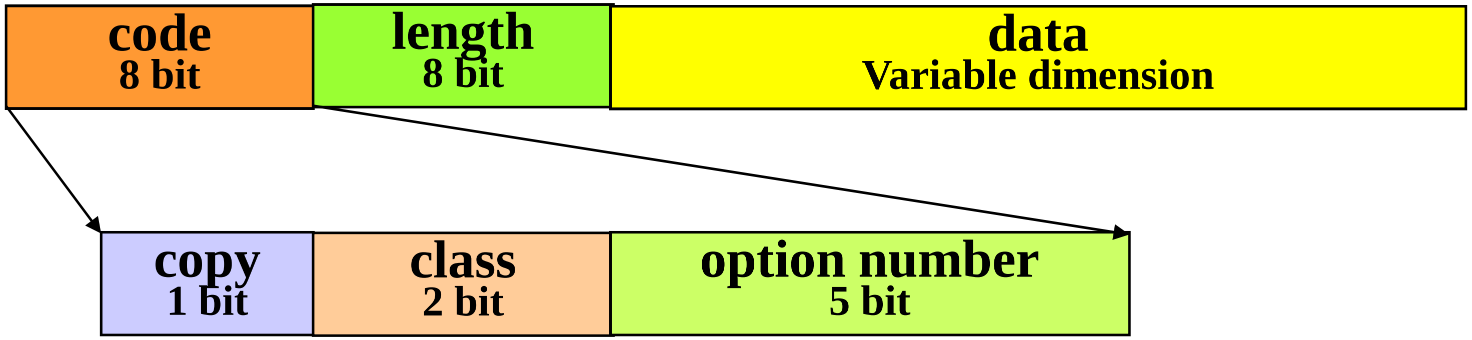


IP Options

- The first 20 bytes of the header are mandatory
- Optional fields may increase the packet length up to 60 byte
- Options are used:
  - *Testing*
  - *Debugging*



IP Options



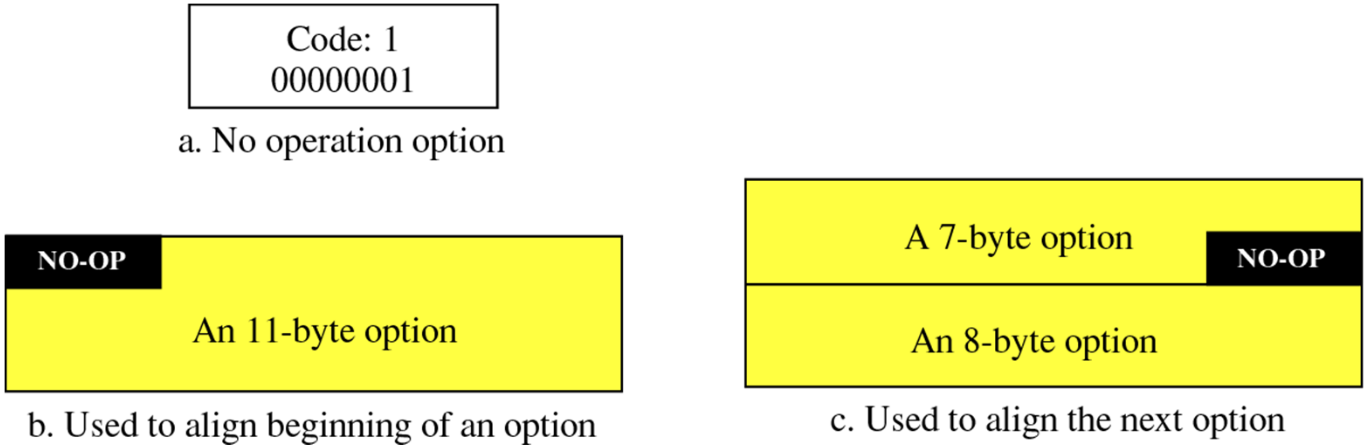
Copy:
- 0 option is copied in the first fragment only
- 1 option is copied in all the fragments

Class:
- 00 fragment control
- 10 management and debugging

Option number:
- 00000 end of option (1 byte)
- 00001 no operation (1 byte)
- 00011 loose source route
- 00100 time stamp
- 00111 record route
- 01001 strict source route

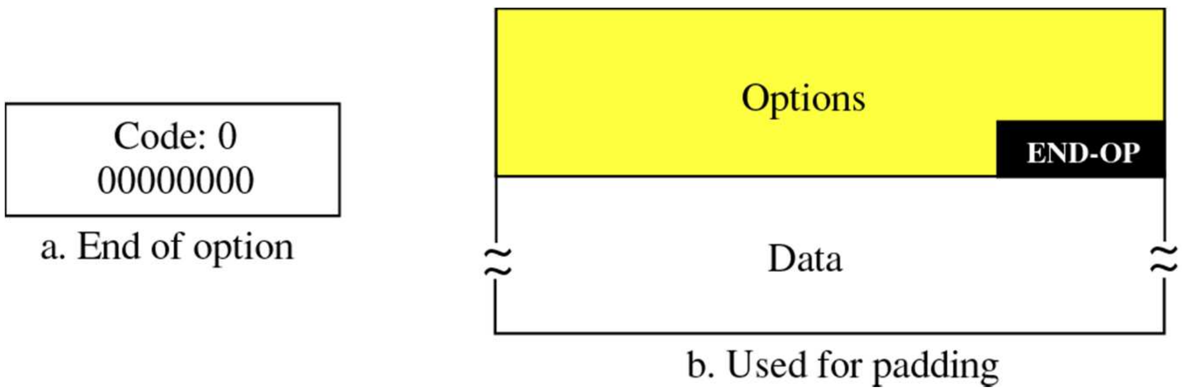
Followed by Data field

End Of Option **and** ***No operation***

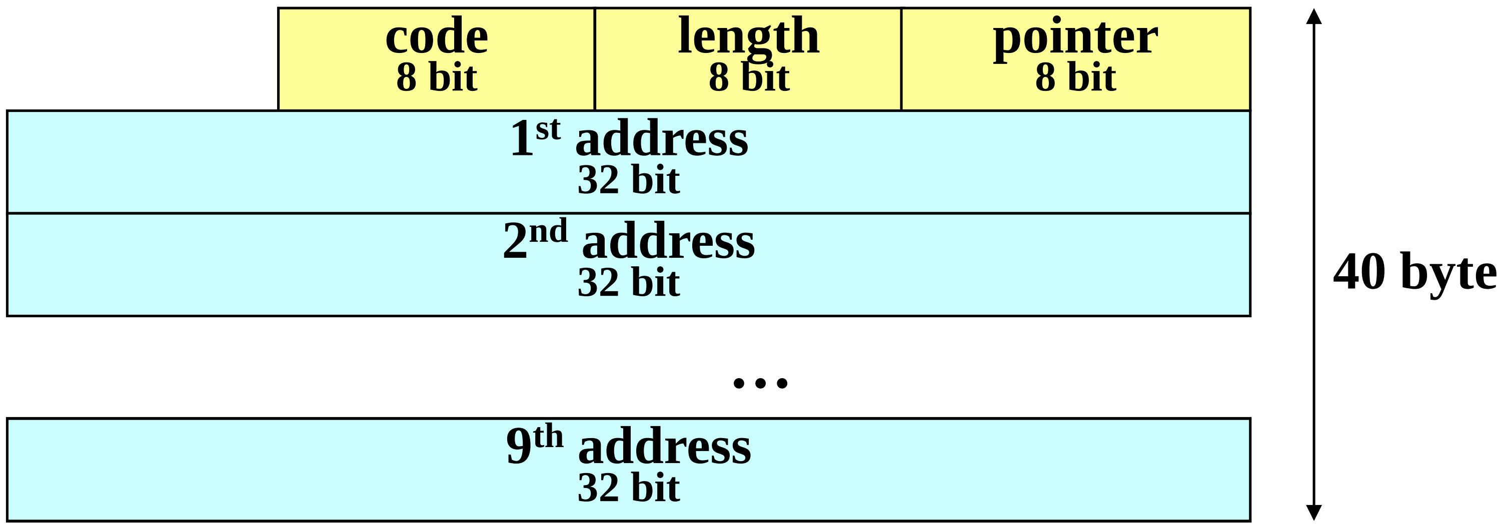


Source: *TCP/IP Protocol Suite, B. Forouzan.*

- Used for *padding*
- Data-less



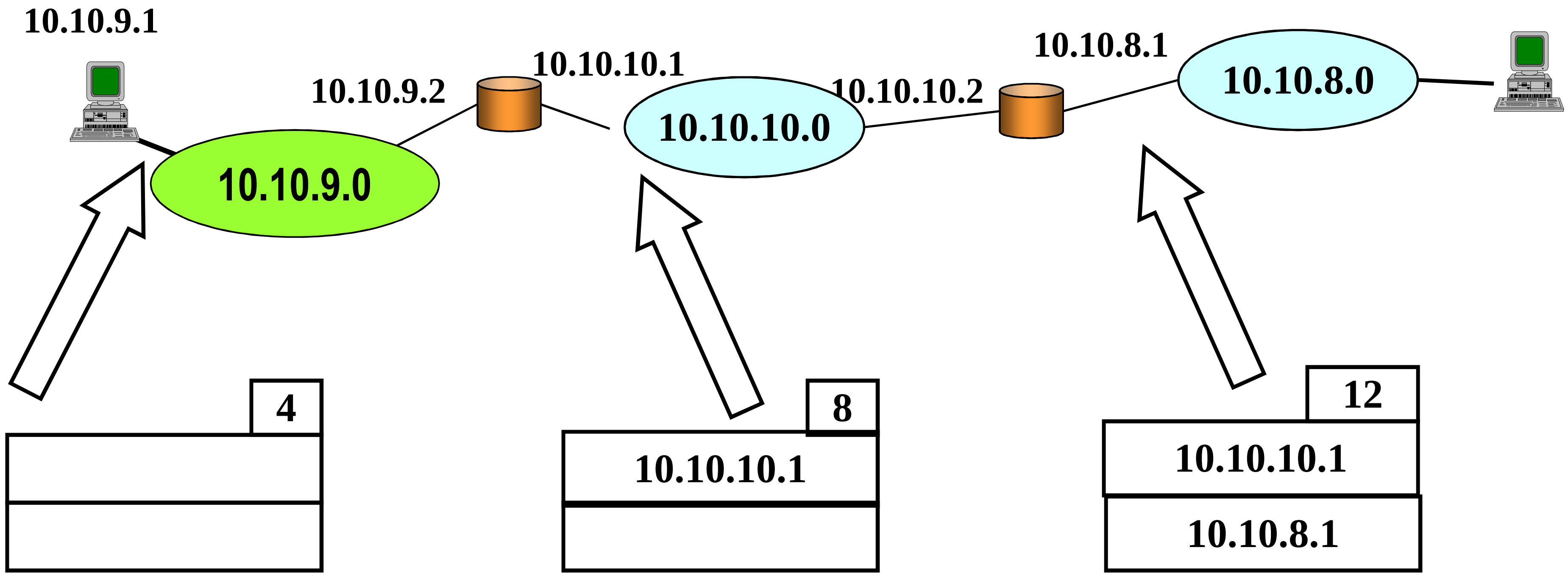
Record Route (1)



- To record the route followed by the IP packet
- *pointer* to identify the first free byte in the *address* list
- For each visited router its IP address is stored in the corresponding *address* field and the *pointer* is incremented by 4

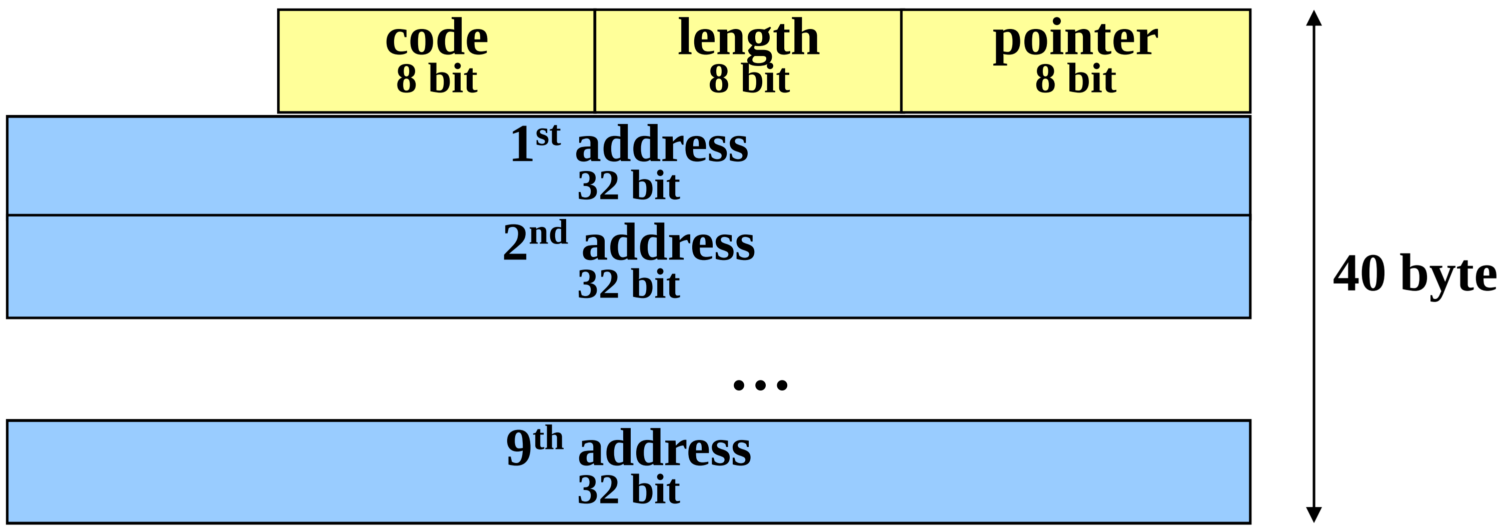
Record Route (2)

- *Example:*



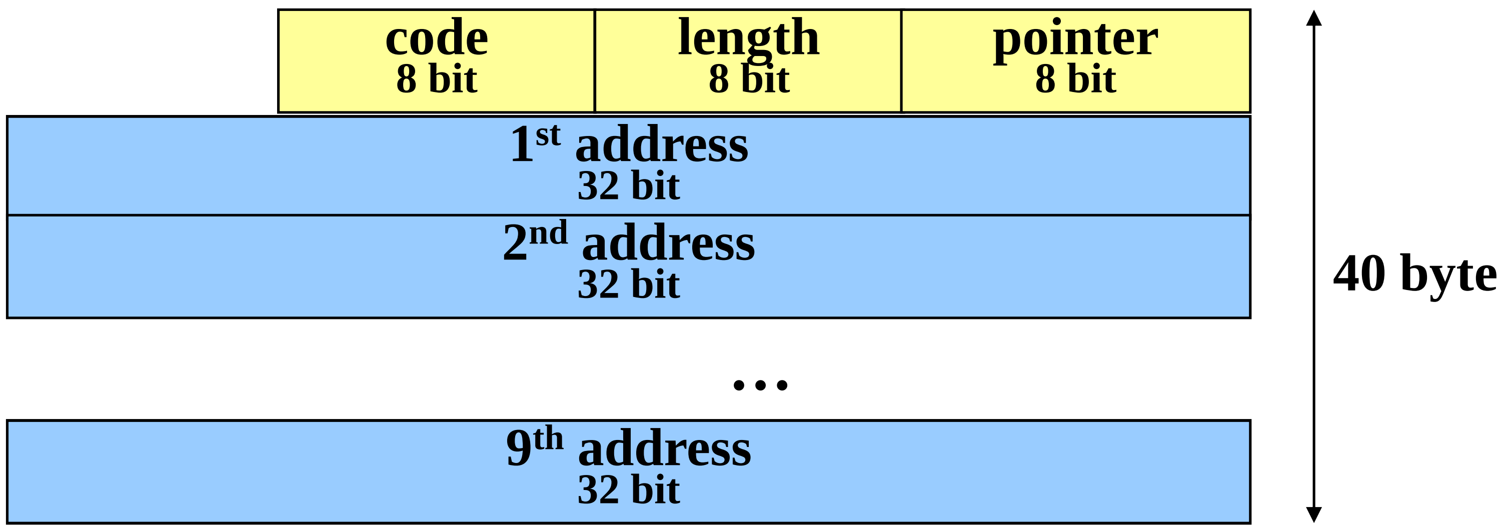
Strict Source Route

- *Source routing* mechanism
- *The address* list reports the IP address of the router to be visited
- The pointer is incremented by 4 at each hop
- If the packet reaches an unwanted router, the packet is dropped (error message)
- (scarcely used!!!)



Loose Source Route

- Like the previous one, but other routers can be visited (packet is not dropped)
- (scarcely used!!!)



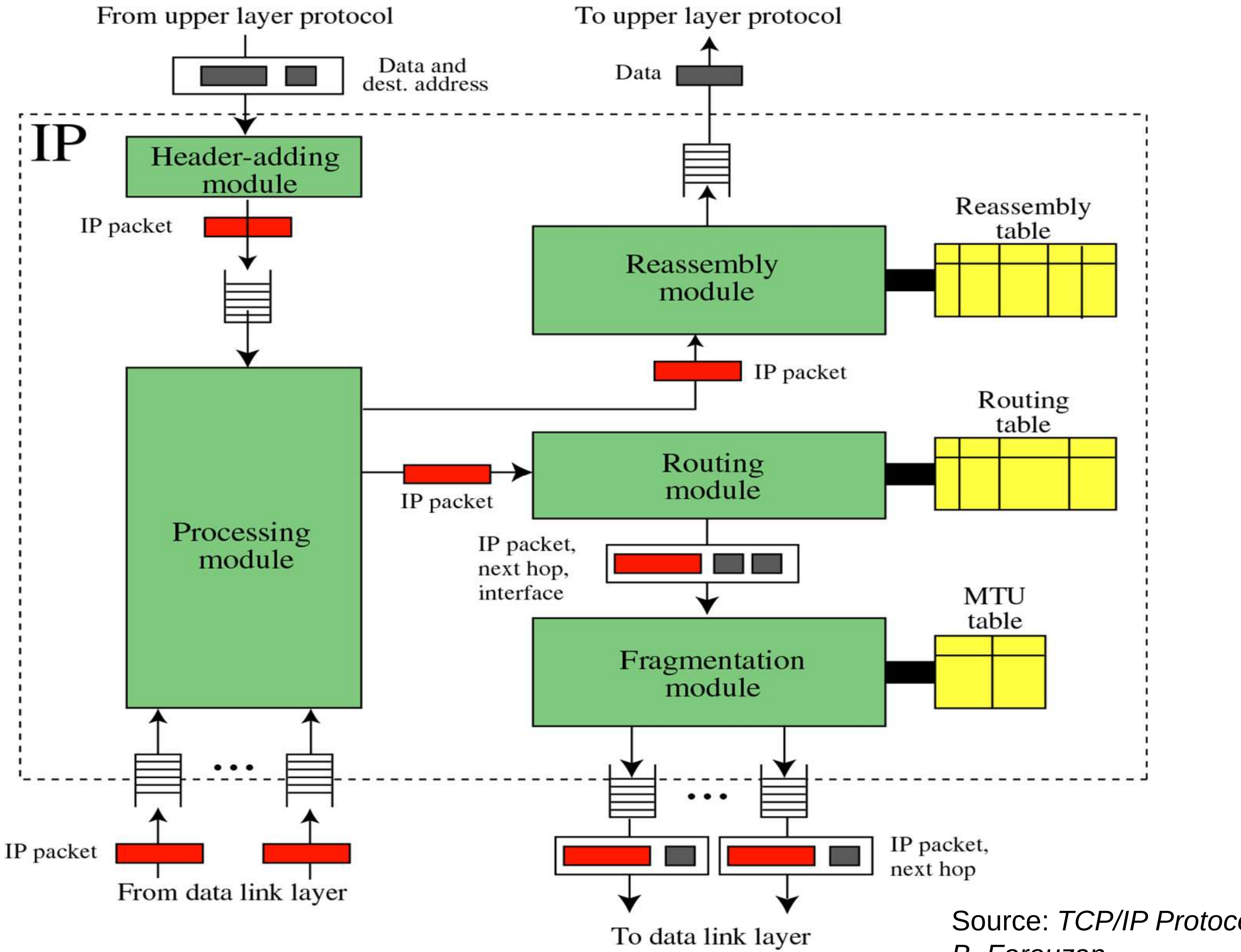
Time Stamp

code 8 bit	length 8 bit	pointer 8 bit	O-Flow 4 bit	Flag 4 bit
1st address 32 bit				
1st time stamp 32 bit				
2nd address 32 bit				
2nd time stamp 32 bit				

...

- Measure of the absolute processing time of a router
- the *Over-Flow* field reports the number of routers which did not add the *timestamp*
- *The Flag* field specifies the operation mode chosen by the sender

IP Protocol X-Rayed



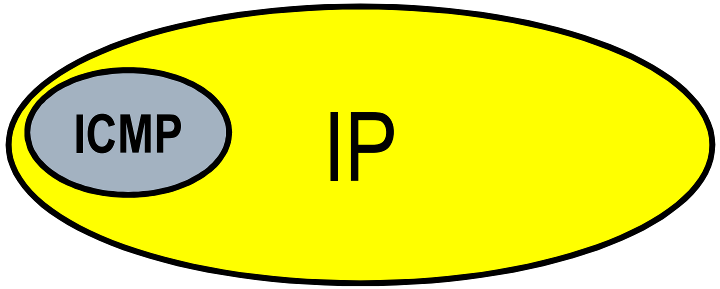
Source: *TCP/IP Protocol Suite, B. Forouzan.*

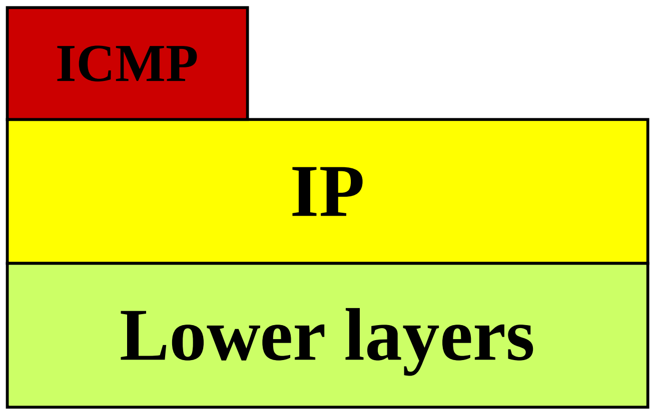
Internet Control Message Protocol (ICMP)

RFC 792

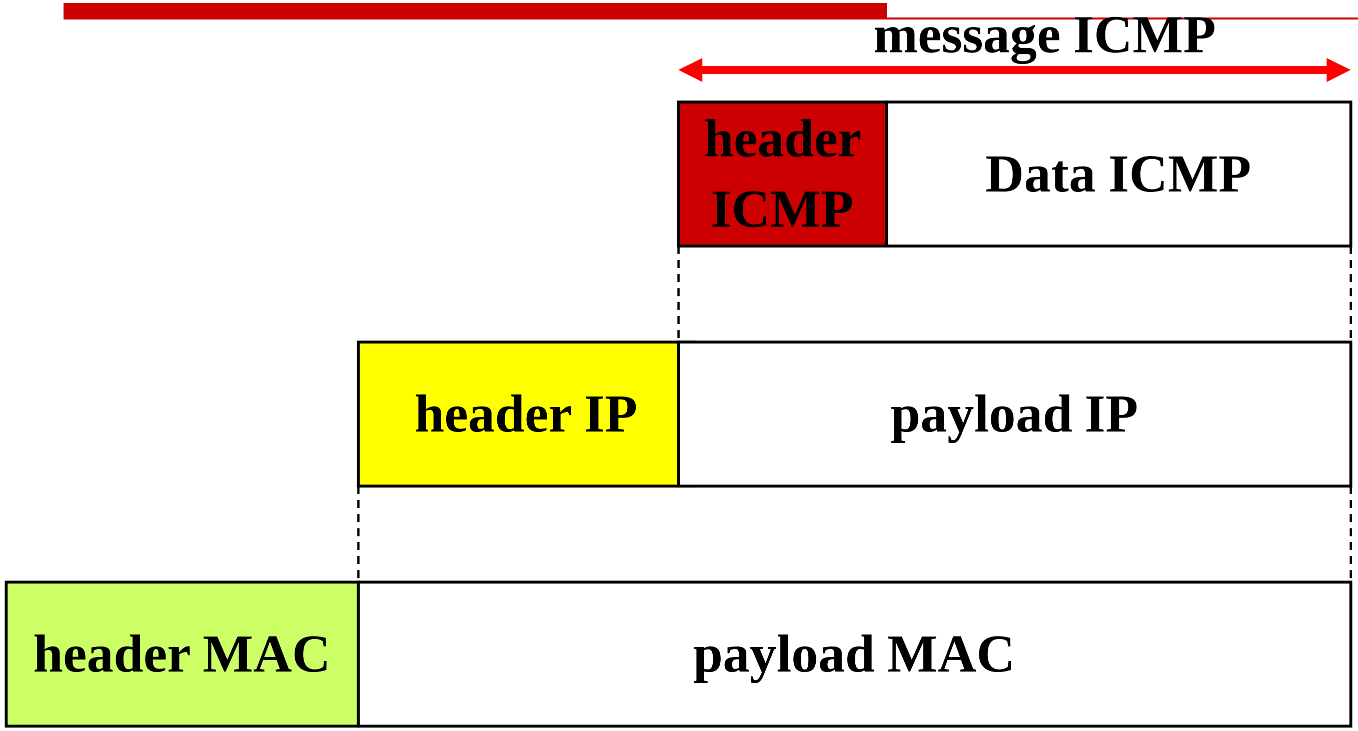
Internet Control Message Protocol (ICMP)

- Signaling protocol between hosts and routers (error signaling, configuration, etc.)
- It’s a layer three protocol (runs side by side with the IP)
- ICMP messages are transported by the IP (ICMP can be seen as an IP user)





Internet Control Message Protocol (ICMP)



- In the IP header the *protocol* field points to the ICMP
- The ICMP message is contained in an IP packet

ICMP Message Format

<table>
<tr><td>type
8 bit</td><td>code
8 bit</td><td colspan="2">checksum
16 bit</td></tr>
<tr><td colspan="4">Remaining header
32 bits</td></tr>
<tr><td colspan="4">Data
Variable length</td></tr>
</table>

Type		Type	
0	Echo reply	11	Parameter problem
3	Destination unreachable	13	Timestamp request
4	Source Quench	14	Timestamp reply
5	Redirect (change a route)	17	Address mask request
8	Echo request	18	Address mask reply
11	Time exceeded		

Types of Messages

- Error Reporting
  - *Destination Unreachable* (type 3)
  - *Source Quench* (type 4)
  - *Time Exceeded* (type 11)
  - *Parameter Problem* (type 12)
  - *Redirection* (type 5)
- Query
  - *Echo Request/Reply* (type 8,0)
  - *Timestamp Request/Reply* (type 13/14)
  - *Address Mask Request/Reply* (type 17/18)
  - *Router Solicitation/Advertisement* (type 10/9)

Error Reporting

- ICMP signals, does not correct
- The error is notified to the source of the IP packet
- Types of Events
  - *Destination Unreachable* (type 3)
  - *Source Quench* (type 4)
  - *Time Exceeded* (type 11)
  - *Parameter Problem* (type 12)
  - *Redirection* (type 5)
- The error signaling messages contain the IP header and the first 8 data byte of the corresponding IP packet

Destination Unreachable

type (3)	code (0-12)	checksum
unused (0)		
header + first 64 bits of the IP packet		

- Whenever a router drops a packet it generates an error message to the packet source
- *code* field identifies the type of error
- Only when the router can get aware of the error
- The most common error is due to an unreachable destination (*code = 7)*

Destination unreachable

type (3)	code (0-12)	checksum
unused (0)		
header + first 64 bits of the IP packet		

Some Codes:

0 network unreachable

1 host unreachable

2 protocol unreachable

3 port unreachable

4 fragmentation needed and DF set

5 source route failed

...

Time exceeded

type (11)	code (0-1)	checksum
unused (0)		
header + first 64 bits of the IP packet		

- Code 0 (sent by routers)
  - *time exceeded* when the TTL goes to 0
  - *time exceeded* sent to the packet source
- Code 1 (sent by the destination)
  - When some fragments are still missing

Parameter problem

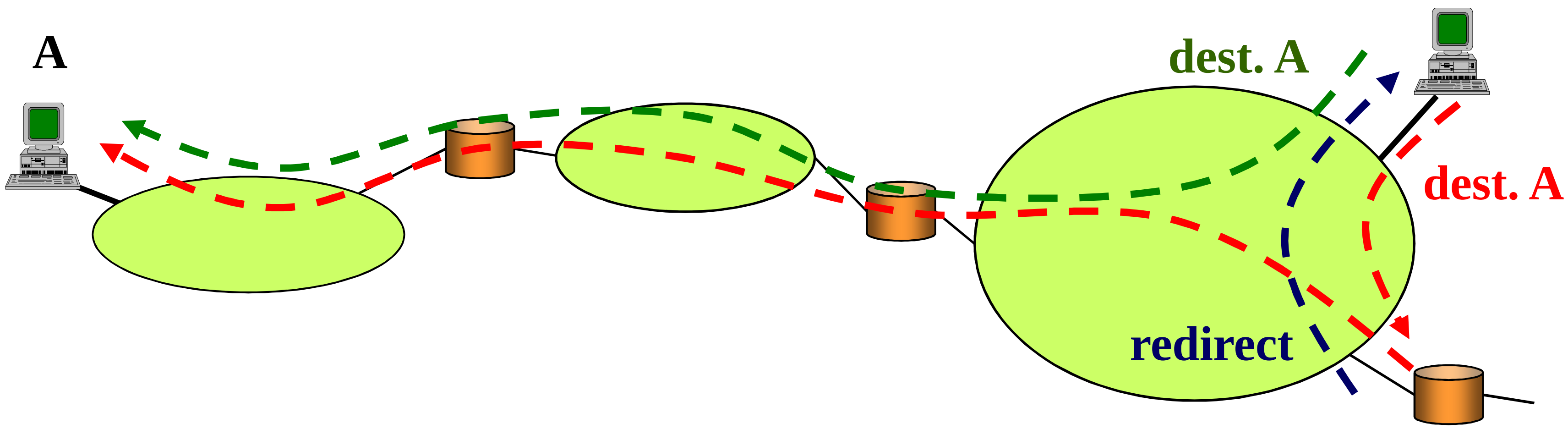
type (12)	code (0-1)	checksum
pointer	unused (0)	
header + first 64 bits of the IP packet		

- Code 0
  - If the IP header has some inconsistencies in any of its parameters; the *pointer* field points to the byte which caused the problem
- Code 1
  - If an option is not implemented or some option parameters are missing

Redirect

type (5)	code (0-3)	checksum
IP address of the router		
header + first 64 bits of the IP packet		

- To change default gateway

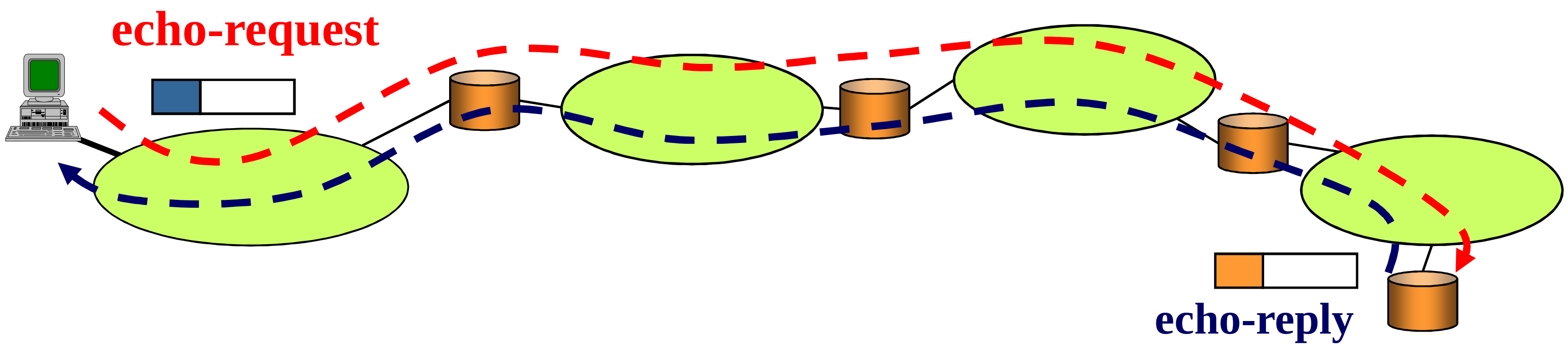


Diagnostic Functionalities

- Request & Reply Paradigm
- Types of messages:
  - *Echo Request/Reply* (type 8,0)
  - *Timestamp Request/Reply* (type 13/14)
  - *Address Mask Request/Reply* (type 17/18)
  - *Router Solicitation/Advertisment* (type 10/9)

Echo Functionalities

- *Echo-request* and *Echo-reply* to test the connectivity towards a given IP address
- A device receiving an *Echo-request* immediately answers with an *Echo reply*

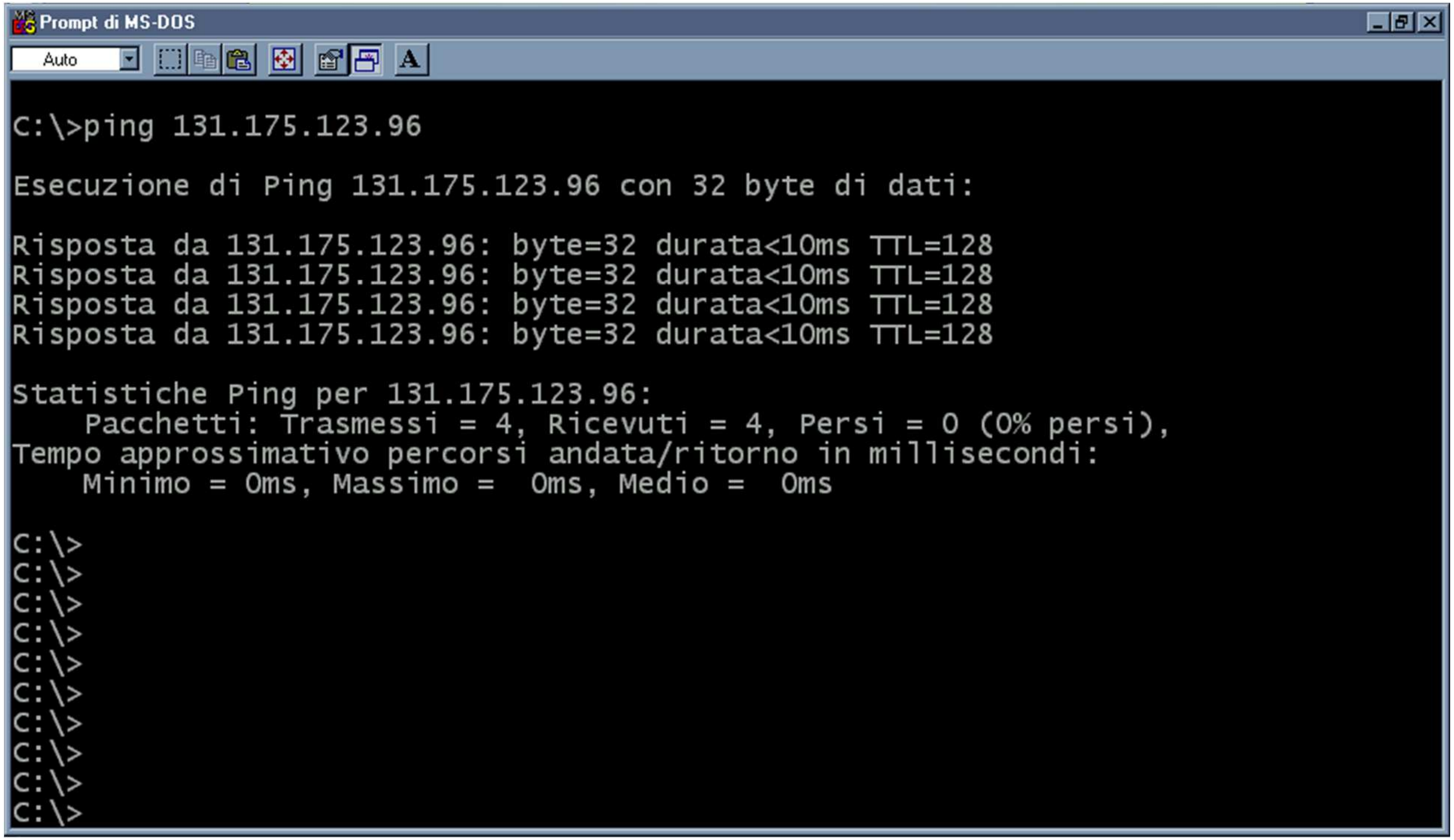


Echo Messages

type (8 request, 0 reply)	code (0)	checksum
identifier		sequence number
optional data		

- *identifier* chosen by the sender
- Reply messages report the same *identifier* of the requests
- Consecutive requests may have the same *identifier* and different *sequence numbers*
- An arbitrary sequence of bits may be added by the sender in the optional field; the same sequence must be reported by the receiver in the reply messages

Echo Usage: PING Application



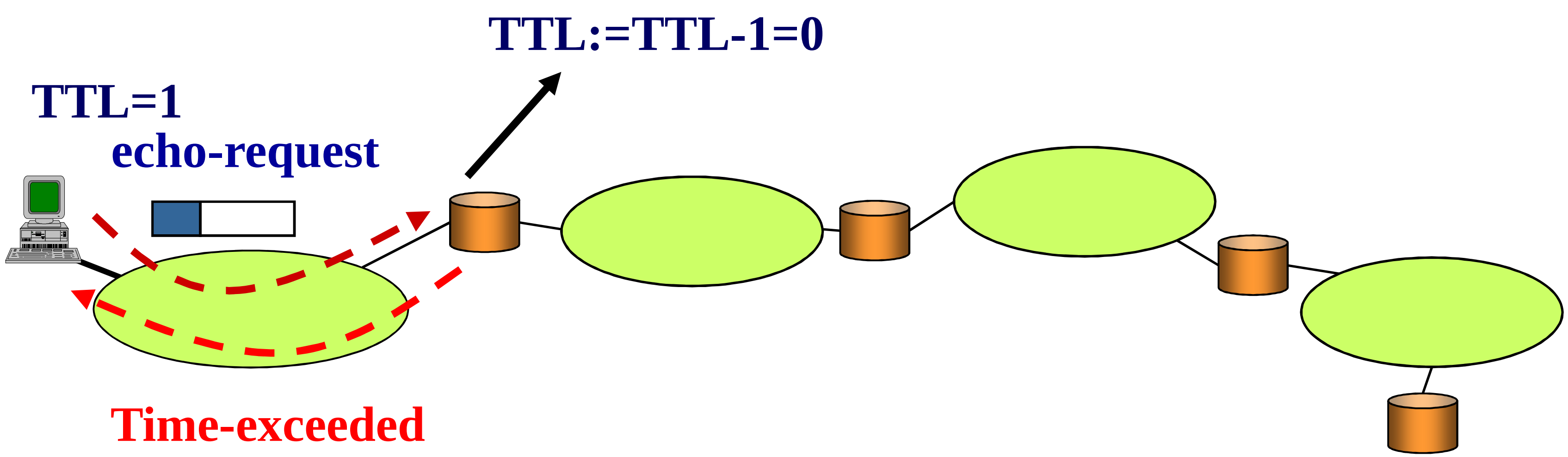
Traceroute Application

Prompt dei comandi

```
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>tracert 132.151.6.21

Rilevazione instradamento verso www.ietf.org [132.151.6.21]
su un massimo di 30 punti di passaggio:

  1   901 ms   794 ms   741 ms  10.136.50.3
  2  1553 ms   982 ms   696 ms  10.136.49.1
  3   722 ms   659 ms   690 ms  10.136.48.161
  4   715 ms   595 ms   659 ms  10.127.1.22
  5  1529 ms   899 ms   628 ms  10.128.219.60
  6  1379 ms   926 ms   692 ms  10.128.219.188
  7  1481 ms   900 ms   719 ms  10.129.211.10
  8   882 ms   900 ms   725 ms  194.185.97.2
  9   859 ms   899 ms   928 ms  pos3-0-0.milano1-cr10.net.inet.it [213.92.71.249]
 10   822 ms   715 ms   665 ms  ge3-1.milano1-gsr0.net.inet.it [194.185.46.77]
 11   758 ms   642 ms   728 ms  mno-b1-pos2-7.telia.net [213.248.77.165]
 12   688 ms   639 ms   897 ms  prs-bb1-pos1-3-0.telia.net [213.248.65.165]
 13   857 ms   765 ms   599 ms  ldn-bb1-pos7-2-0.telia.net [213.248.64.10]
 14   876 ms   983 ms  1025 ms  ldn-bb2-pos0-2-0.telia.net [213.248.65.174]
 15   756 ms   928 ms   842 ms  nyk-bb2-pos6-0-0.telia.net [213.248.65.94]
 16   846 ms   852 ms   901 ms  nyk-bb1-pos0-0-0.telia.net [213.248.80.133]
 17   880 ms   945 ms   847 ms  nyk-i2-pos1-0.telia.net [213.248.82.10]
 18   814 ms   686 ms  1002 ms  POS3-1.IG4.NYC4.ALTER.NET [208.192.177.29]
 19   737 ms   769 ms   789 ms  904.at-1-0-0.XR3.NYC4.ALTER.NET [152.63.19.238]
 20   798 ms   781 ms   997 ms  0.so-2-0-0.XL1.NYC4.ALTER.NET [152.63.17.29]
 21   682 ms   733 ms   797 ms  0.so-2-0-0.TL1.NYC8.ALTER.NET [152.63.0.153]
 22   834 ms   789 ms   787 ms  0.so-5-3-0.TL1.DCA6.ALTER.NET [152.63.0.21]
 23   819 ms   802 ms   815 ms  0.so-6-0-0.XL1.DCA6.ALTER.NET [152.63.38.70]
 24   847 ms   788 ms   770 ms  0.so-0-0-0.XR1.DCA6.ALTER.NET [152.63.35.113]
 25   803 ms   869 ms   688 ms  285.at-5-1-0.XR1.TCO1.ALTER.NET [152.63.33.50]
 26   720 ms   779 ms   820 ms  193.ATM7-0.GW5.TCO1.ALTER.NET [152.63.39.85]
 27   876 ms   779 ms   824 ms  cnrl-gw.customer.alter.net [157.130.44.142]
 28     *        *        *     Richiesta scaduta.
 29     *        *        *     Richiesta scaduta.
 30     *        *        *     Richiesta scaduta.

Rilevazione completata.
```

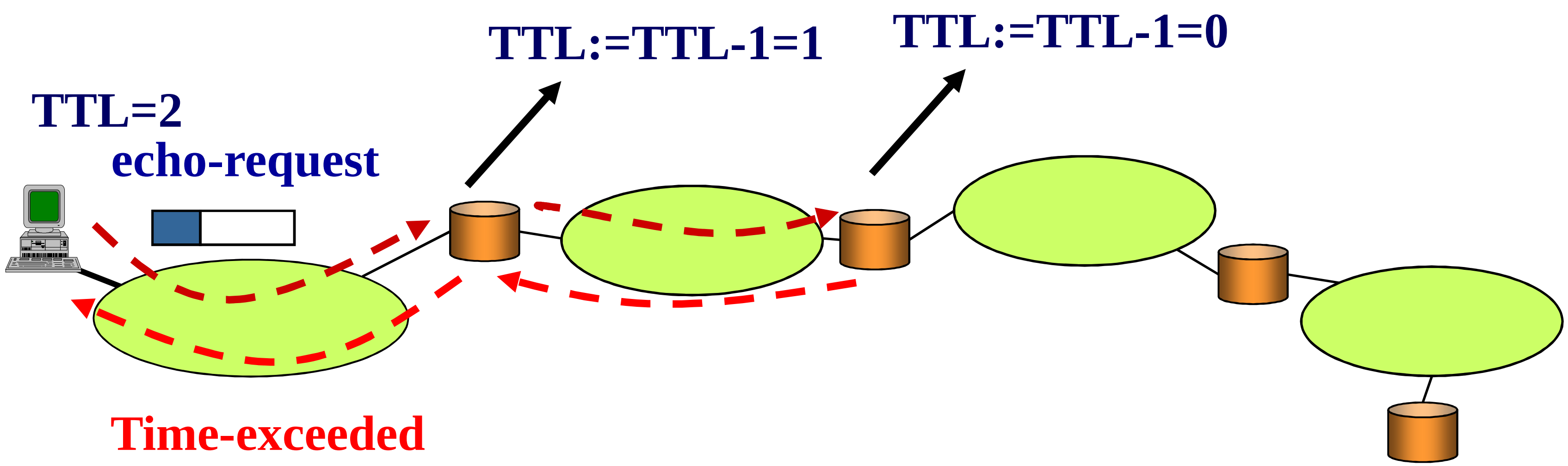
Traceroute: How Does it Work?

- *traceroute* uses (normally) *Echo-requests* towards a specific destination
- The first packet has TTL=1



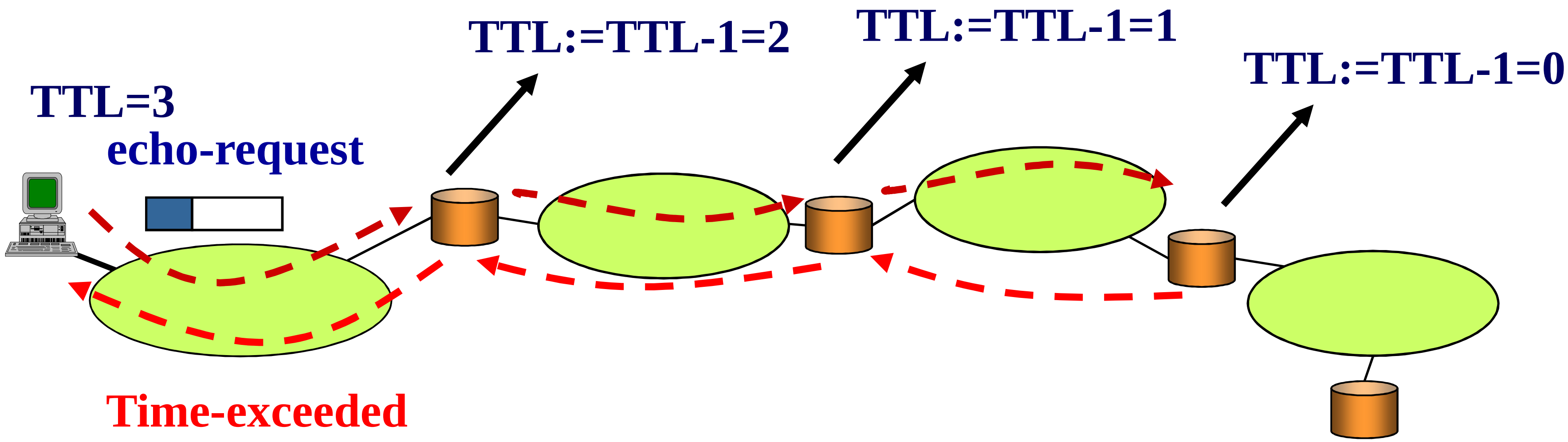
Traceroute: How Does it Work?

- The second packet has TTL=2



Traceroute: How Does it Work?

- The third has TTL=3, and so on so forth …



Timestamp request and *reply*

type (13 request, 14 reply)	code (0)	checksum
identifier		sequence number
originate timestamp		
receive timestamp		
transmit timestamp		

- Used for exchanging information on the source and destination clocks
- *originate timestamp:* filled in by the source
- *receive timestamp*: filled in by the destination upon reception of the packet
- *transmit timestamp:* filled in by the destination before answering

Address mask request **and** ***reply***

type (17 request, 18 reply)	**code** (0)	**checksum**
identifier		**sequence number**
address mask		

- Used to gather info on the netmask (host/router)
- The *address mask* field is filled in by the destination