Data Communication
(DC)

Lecture 7

Overview of the contents

- Network Address Translation (NAT)
- Forwarding of IP Packets
- Internet Protocol (IPv4)
- Internet Control Message Protocol (ICMPv4)

Network Address Translation (NAT)

Network Layer

Network Address Translation (NAT)

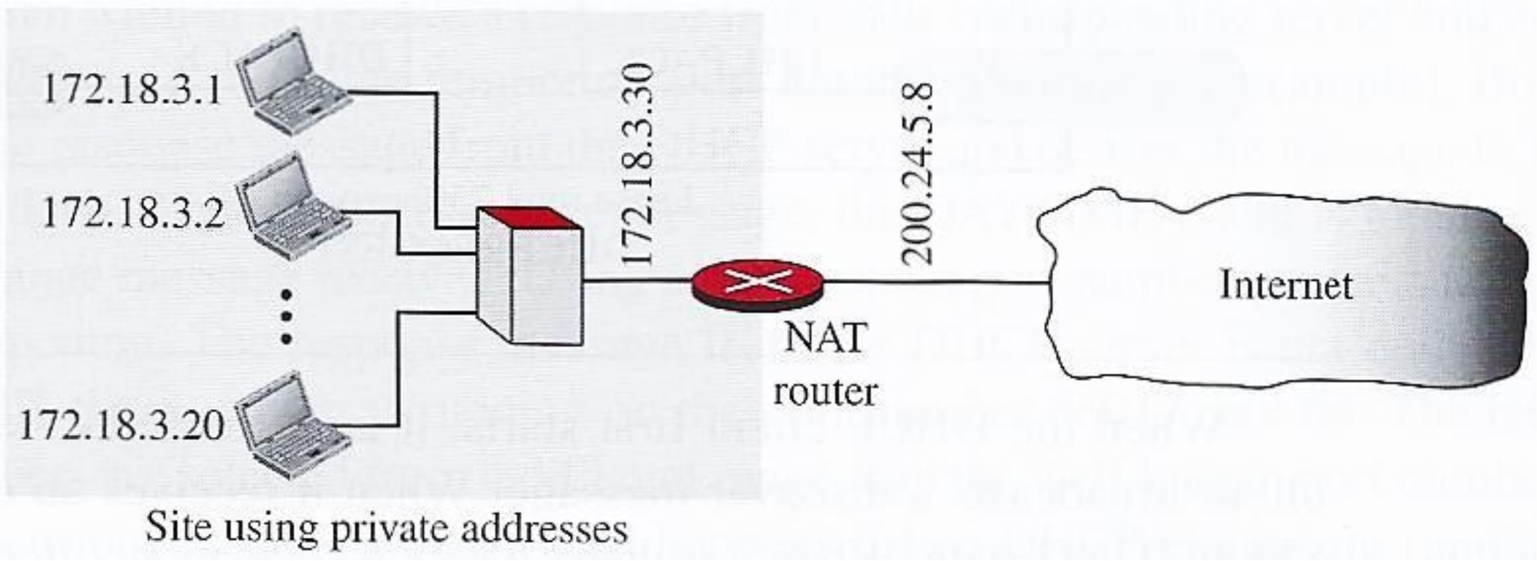
- One problem of the distribution of addresses through ISPs is that the ISP may not be able to grant the increasing demand when a business grows and needs more addresses because the addresses before and after the range may have been already allocated to other networks.
- However, only few addresses are needed for universal communication while many private block addresses can be used for internal communication.
- A technology that can provide the mapping between the private and universal addresses is **Network Address Translation (NAT)**.

Network addresses			Number of addresses	
10.0.0.0	-	10.255.255.255	2^{24}	(16.777.216)
172.16.0.0	-	172.31.255.255	2^{20}	(1.048.576)
192.168.0.0	-	192.168.255.255	2^{16}	(65.536)
169.254.0.0	-	169.254.255.255	2^{16}	(65.536)

Network Layer

Network Address Translation (NAT)

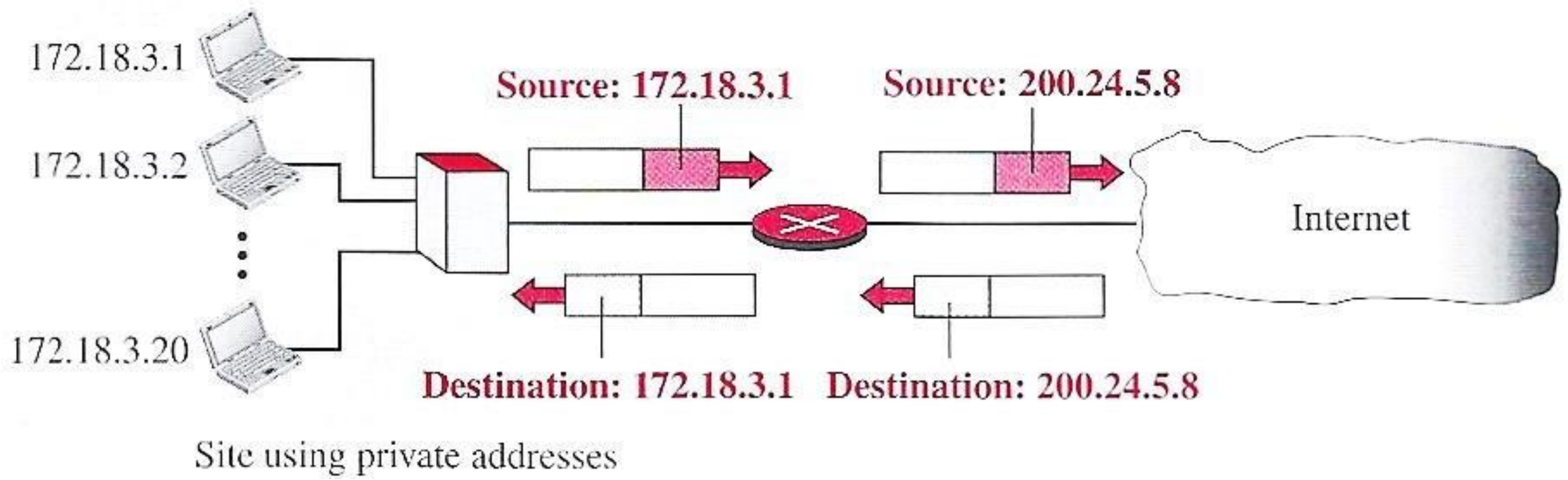
NAT can only be done via a NAT-capable router that runs NAT software.



- In all outgoing packets, the source address is replaced with the global address of the NAT router.
- In all incoming packets, the destination address is changed from the NAT router's global address to appropriate private address.

Network Layer

Network Address Translation (NAT)



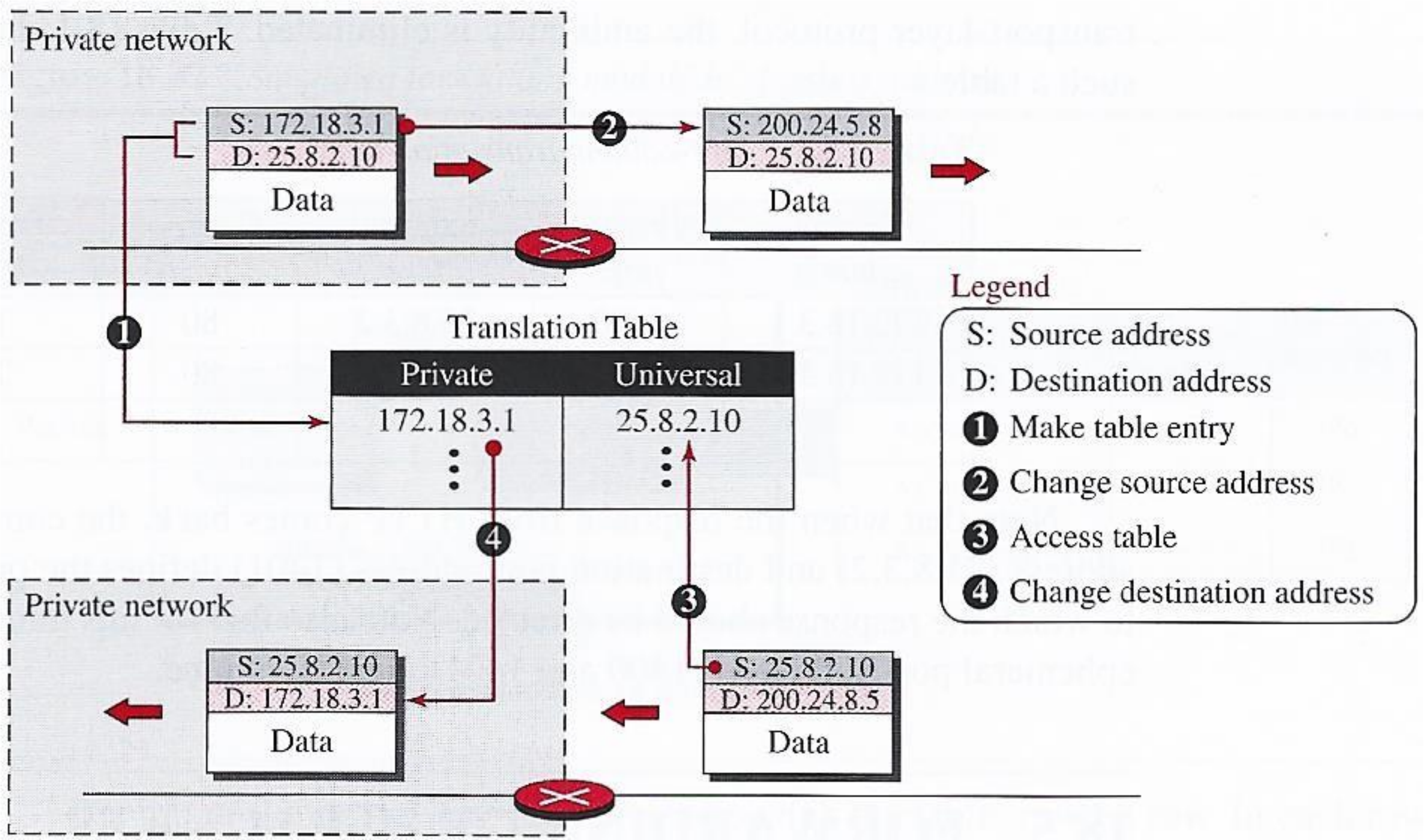
Replacing the source address in the outgoing packets is straightforward.

But how can the NAT router know the destination address of a packet coming from the Internet?

- This problem is solved with a **translation table**, internally in the NAT router.

Network Layer

Network Address Translation (NAT)



The use of only one global address by the NAT router allows only one private-network host to access a given external host.

Network Layer

Network Address Translation (NAT)

A pool of IP addresses: Since a NAT router only has one global address, only one host from the private network can access an external host.

To remove this restriction, the NAT router can use a pool of global addresses e.g., 4 global addresses: 200.24.5.8; 200.24.5.9; 200.24.5.10; 200.24.5.11

Now up to 4 hosts from the private network can communicate with the same external host.

Drawback: No private-network host can access two external server programs on the same host (e.g., http and FTP).

Network Layer

Network Address Translation (NAT)

Both IP address and port number: in order to allow a many-to-many mapping relationship between hosts of a private network and external server programs, more information can be provided in the translation table.

Example: two hosts with the addresses 172.18.3.1 and 172.18.3.2 (from the private network) want to access a HTTP server on an external host with the address 25.8.3.2.

Private address	Private port	External address	External port	Transport protocol
172.81.3.1	1400	25.8.3.2	80	TCP
172.81.3.2	1401	25.8.3.2	80	TCP
…	…	…	…	…

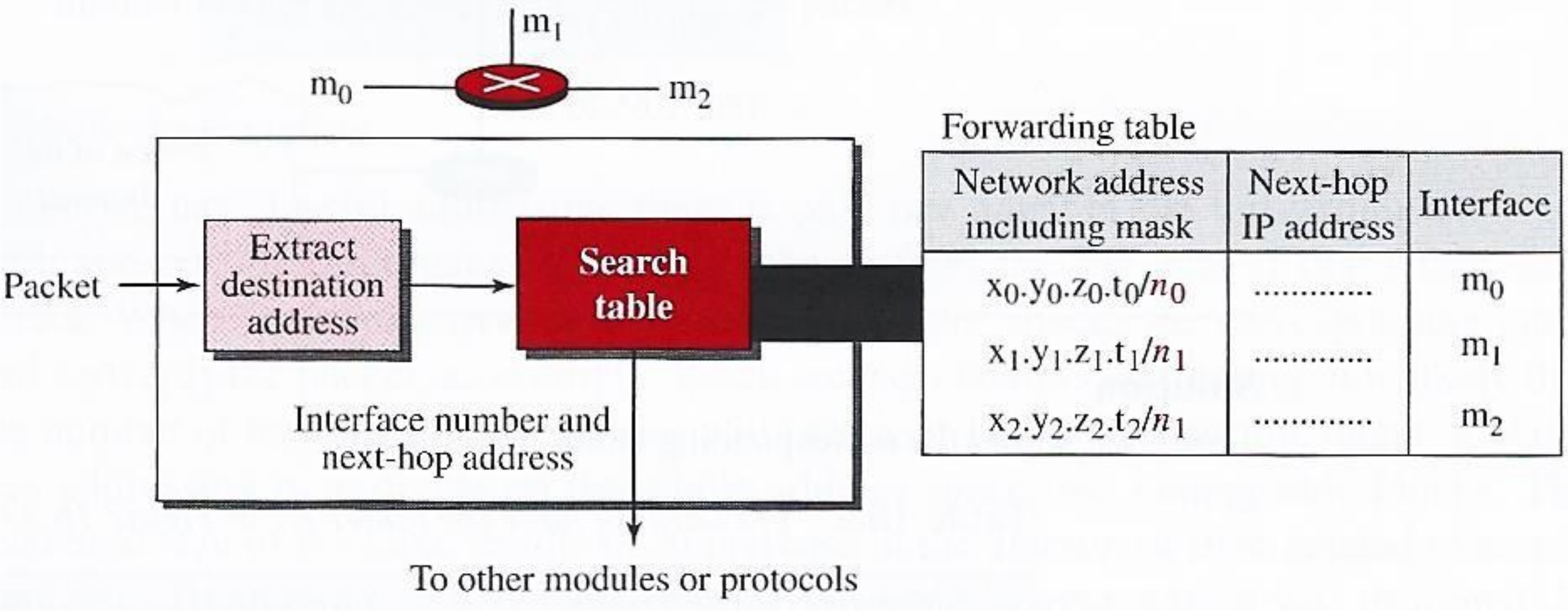
Note that when a response from HTTP comes back, it is the combination of the source address 25.8.3.2 and the destination port number 1400 or 1401 that defines the host on the private network.

Also note that to make this work, **the private port numbers must be unique**.

Forwarding of IP Packets

Network Layer

Forwarding of IP packets (based on destination address)


m1
m0
m2
Packet
Extract destination address
Search table
Interface number and next-hop address
To other modules or protocols

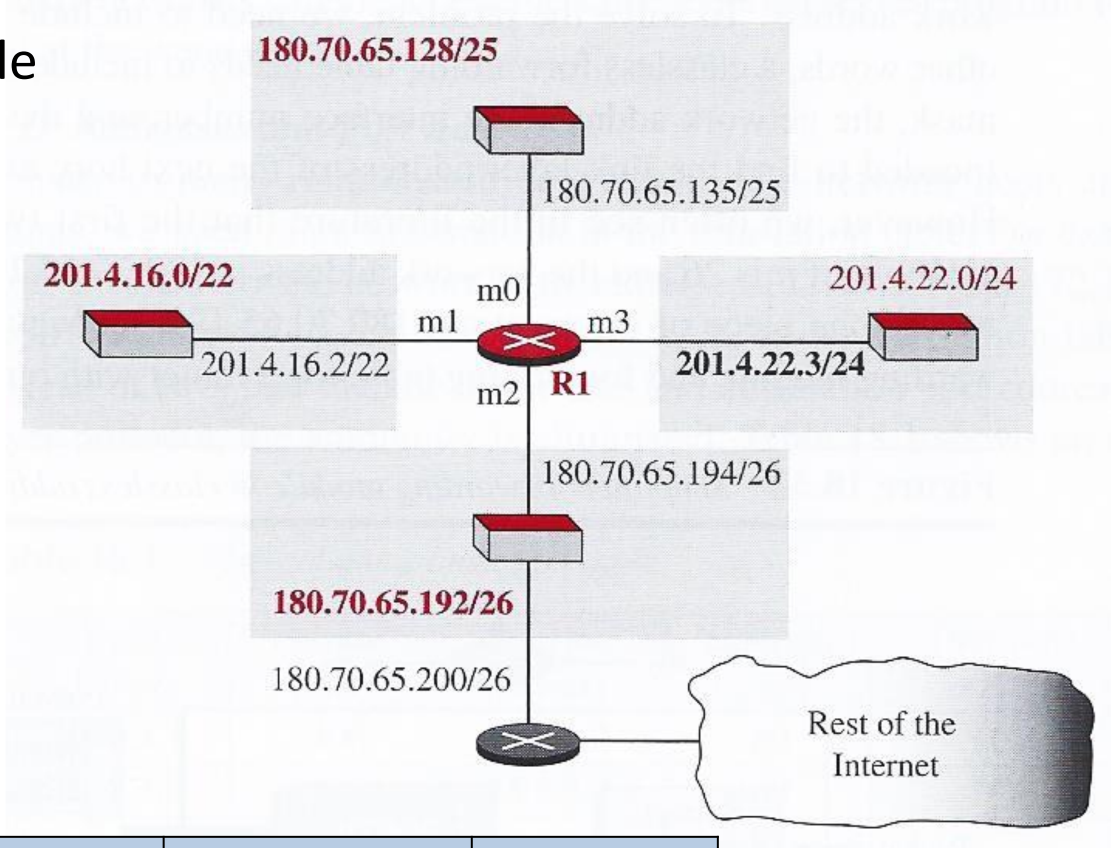
Forwarding table

Network address including mask	Next-hop IP address	Interface
$x_0.y_0.z_0.t_0/n_0$	....	m_0
$x_1.y_1.z_1.t_1/n_1$	....	m_1
$x_2.y_2.z_2.t_2/n_1$	....	m_2

- Destination address and the mask (/n) are used.
- This result is compared with the network addresses (in column 1).
- When a match is found, the Next-hop address (column 2) is used to (via ARP) find the link layer address to be used in the frame to be forwarded.
- Column 3 indicates which output interface of the router the frame is to be sent from.

Network Layer

Forwarding of IP packets: Example



Network address / Mask	Next hop	Router I/O
180.70.65.192/26	-	m2
180.70.65.128/25	-	m0
201.4.22.0/24	-	m3
201.4.16.0/22	-	m1
Default	180.70.65.200	m2

Forwarding table for router R1 in the Figure

Network Layer

Address aggregation

- When we use classless addressing, it is likely that the routing tables will be larger in size.

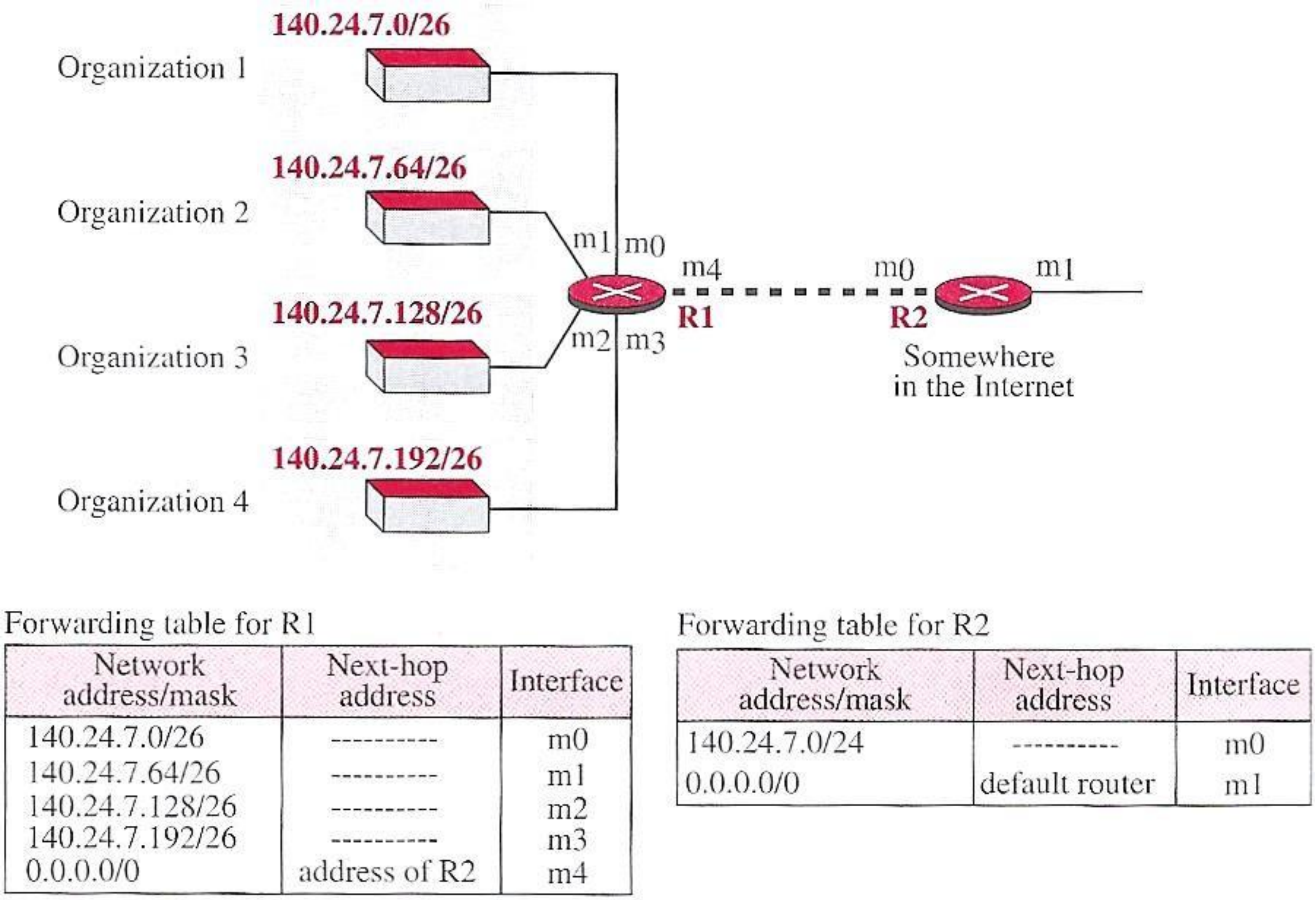
- This is because the whole idea of classless addressing is to divide the entire address space into smaller address blocks that are easier to handle.

- As the tables get larger, so does the search time in them.

- To address this issue, the idea of **address aggregation** was designed.

Network Layer

Address Aggregation - Example



Forwarding table for R1

Network address/mask	Next-hop address	Interface
140.24.7.0/26	----------	m0
140.24.7.64/26	----------	m1
140.24.7.128/26	----------	m2
140.24.7.192/26	----------	m3
0.0.0.0/0	address of R2	m4

Forwarding table for R2

Network address/mask	Next-hop address	Interface
140.24.7.0/24	----------	m0
0.0.0.0/0	default router	m1

Network Layer

Address Aggregation - Example

- Router R1 is connected to 4 organizations (companies), each using 64 addresses.
- Router R2 is somewhere far from router R1.
- Router R1 has a long routing table (here 4), as each packet must be routed correctly to each organization.
- But router R2, in turn, has a small routing table.
- Here, all packets with addresses from 140.24.70.0 to 140.24.70.255 are sent via I/O **m0** regardless of which organization they belong to. This is called **address aggregation**.
- Note, however, that router R2 would have a larger routing table if the 4 address blocks can not be aggregated into a larger block.

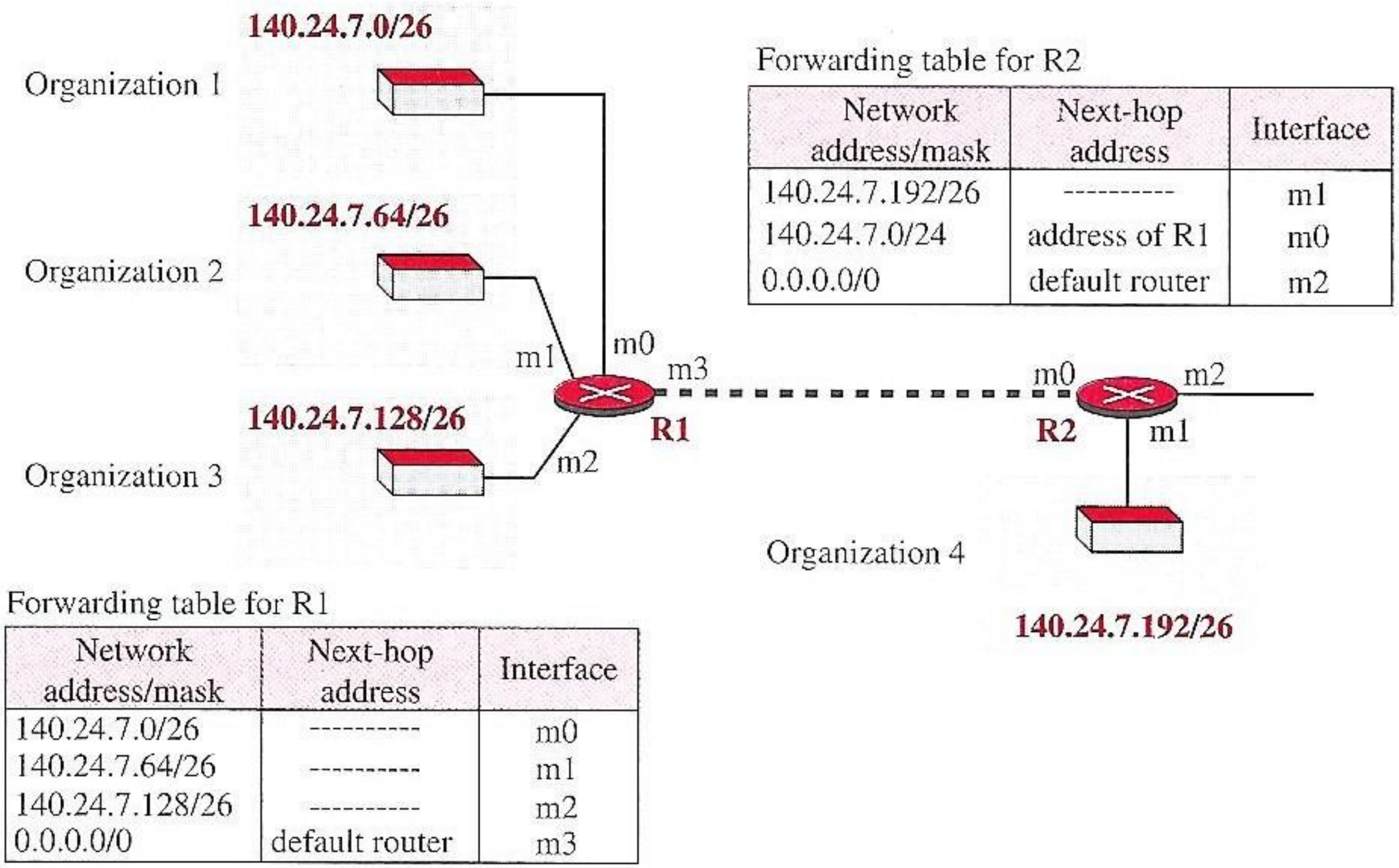
Network Layer

Address Aggregation - Longest mask match

What if e.g., organization 4 geographically is far from the others, can we still apply address aggregation?

The answer is yes! Notice that the routing tables are arranged so that the entries in the forwarding table are sorted from the longest mask to the shortest mask.

This is called: ***Longest mask match***.



Forwarding table for R2

Network address/mask	Next-hop address	Interface
140.24.7.192/26	----------	m1
140.24.7.0/24	address of R1	m0
0.0.0.0/0	default router	m2

Forwarding table for R1

Network address/mask	Next-hop address	Interface
140.24.7.0/26	----------	m0
140.24.7.64/26	----------	m1
140.24.7.128/26	----------	m2
0.0.0.0/0	default router	m3

Network Layer

Address Aggregation - Longest mask match

Assume that a packet arrives at R2 with organization 4 as the destination address **140.24.7.200**.

1. First mask (***/26***) bit-wise AND with the destination address.

   The result gives **140.24.7.192**, which matches the network address in row 1.

   The next-hop address (the destination address in this case) and the router interface number (***m1***) are passed on to ARP for further processing.

The packet reaches organization 4 as it should.

Note here that if mask /24 had been first in the table at R2, then the packet would have mistakenly ended up at router R1!

Network Layer

Hierarchical Routing

To solve the problem of gigantic forwarding tables, we can create a kind of hierarchy in the forwarding tables.

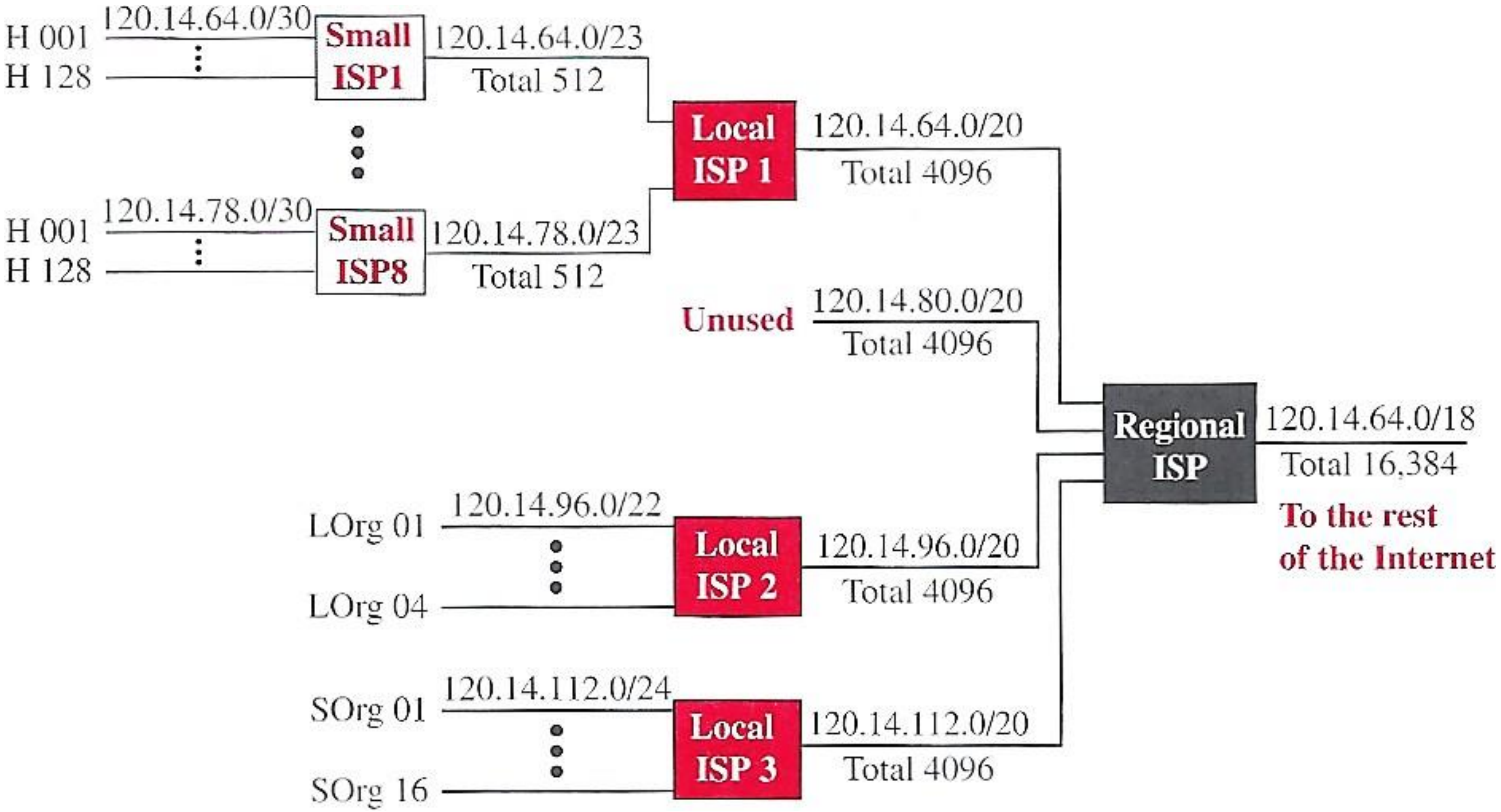
So how does this reduce the size of the routing tables?

- The rest of the Internet does not need to know about this division that the ISP has made inside a network.
- All packets intended for an address in this large block are sent to the ISP.
- This only provides one entry in all routers in the world for all the customers who belong to this ISP.

Of course, routers inside this local ISP must know sub-blocks and are able to forward the packets to specific customers.

Network Layer

Hierarchical Routing – Example



Network Layer

Hierarchical Routing – Example

A regional ISP

gets ***16.384*** addresses with the starting address ***120.14.64.0***
The regional ISP divides these addresses into ***4*** sub-blocks, each with ***4.096*** addresses.
These sub-blocks are now assigned to 3 local ISPs (sub-block 2 is saved for future use)
Note that the mask for each sub-block here is ***/20***. The original block had the mask ***/18***

The first local ISP

has divided its addresses into 8 smaller blocks, each of which is assigned a small ISP.
Each of these small ISPs further divides their address block into 128 households (H001-H128)
Each of these households has 4 addresses. (2^{32-30} = 4 pcs.)
Note that: ***small ISP*** has the mask ***/23*** and every ***household*** has mask ***/30***

The second local ISP

has divided its block into ***4*** smaller blocks.
Each of these blocks is assigned to large customers (LOrg01 – LOrg04) with ***1024*** addresses and a mask of ***/22***.

The third local ISP

has divided its block into ***16*** smaller blocks.
Each of these blocks is assigned to small customers (SOrg01 – SOrg16) with ***256*** addresses and a mask of ***/24***.

Network Layer

Hierarchical Routing – Example

There is a kind of hierarchy in this setup:

All routers on the Internet send packets with addresses from:

120.14.64.0 – 120.14.127.255 to the regional ISP.

The regional ISP sends packets with addresses from:

120.14.64.0 – 120.14.79.255 to the first local ISP.

The first local ISP sends packets with addresses from:

120.14.64.0 – 120.14.64.3 to household H001.

Network Layer

Geographical Routing

To further reduce the size of the routing tables, we extend this hierarchical routing to include geographical routing.

We must divide the entire address space into a few large blocks. We assign a block to America, a block to Europe, a block to Asia, a block to Africa, and so on.

In this way, routers outside e.g., Europe only have one entry to all addresses in Europe.

Network Layer

Forwarding of IP packets (based on labels)

In the 1980s, an effort started to somehow change IP to behave like a connection-oriented protocol in which the routing is replaced by switching.

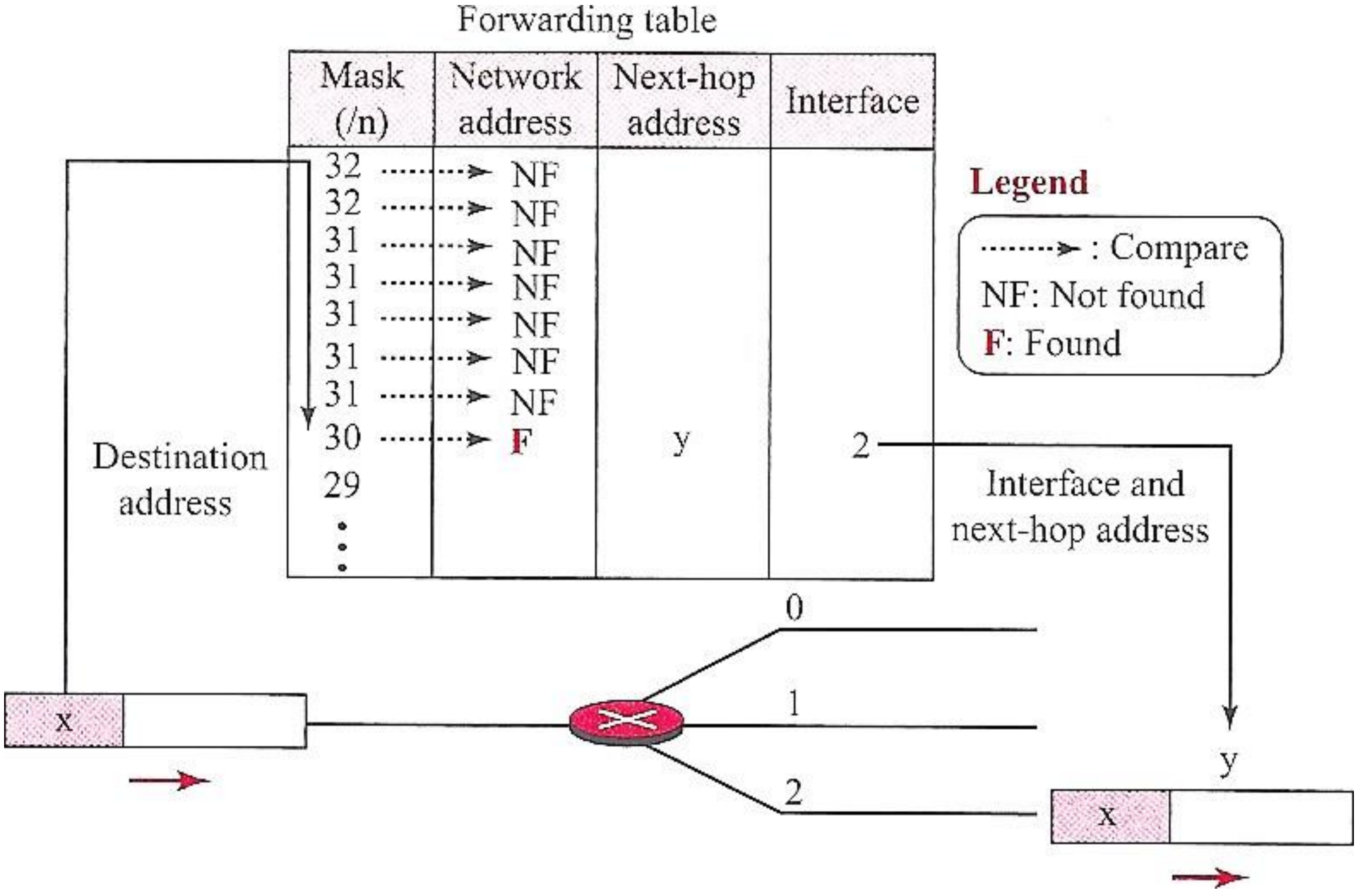
in a connectionless network (datagram approach), a router forwards a packet based on the destination address in the header of the packet. On the other hand, in a connection-oriented network (virtual-circuit approach), a switch forwards a packet based on the label attached to the packet

- ***Connectionless***: datagrams are forwarded based on their destination address by routers.
- ***Connection-oriented***: datagrams are forwarded based on labels attached to them by switches.

Note that: routing is normally based on searching the contents of a table; switching can be done by accessing a table using an index. In other words, routing involves searching; switching involves accessing.

Network Layer

Forwarding of IP packets – Routing Example



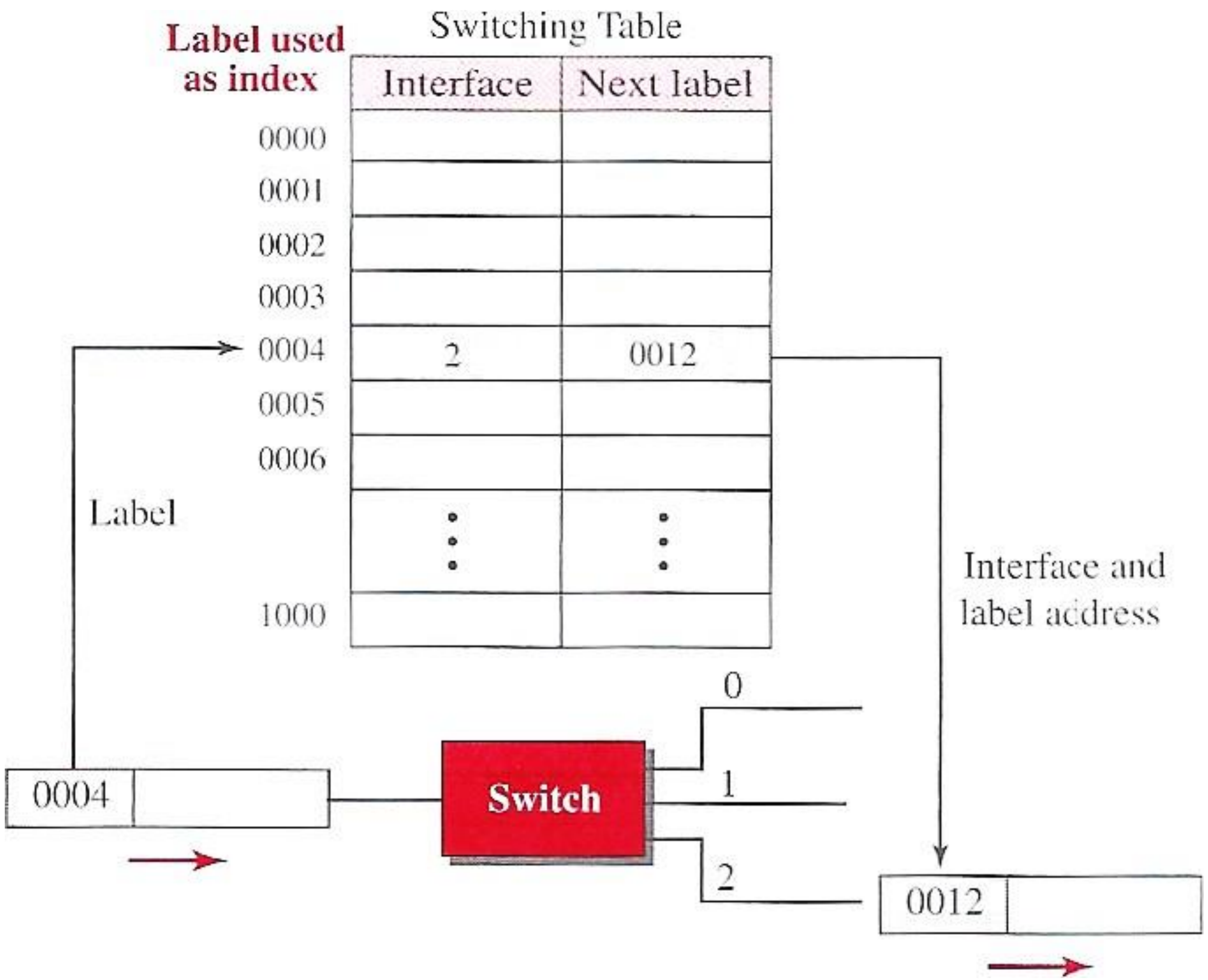
Forwarding table

Mask (/n)	Network address	Next-hop address	Interface
32	NF		
32	NF		
31	NF		
31	NF		
31	NF		
31	NF		
31	NF		
30	F	y	2
29			
⋮			

When routing, a search must be performed! (it takes time)

Network Layer

Forwarding of IP packets – Switching Example



Switching Table

Label used as index	Interface	Next label
0000		
0001		
0002		
0003		
0004	2	0012
0005		
0006		
	⋮	⋮
1000		

Here the label (0004) is used as an index to the table, i.e., no search! finding the information in the table is immediate.

Network Layer

Multi-**P**rotocol **L**abel **S**witching (**MPLS**)

During the 1980s, several manufacturers developed routers that implement switch technology. Later, **IETF** (**I**nternet **E**ngineering **T**ask **F**orce) approved a standard, called **M**ulti-**P**rotocol **L**abel **S**witching (**MPLS)**.

These routers can then replace conventional routers.

- These routers can forward packets based on the destination address,
- But also behave like a switch when packets need to be forwarded based on labels.

A new header

To be able to simulate connection-oriented switching using protocols such as ***IP***, you need to add an extra field to the packet that contains the label.

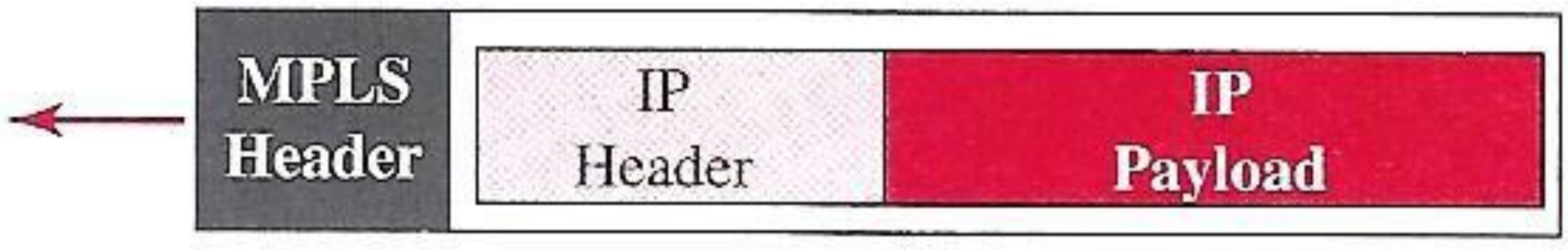
IPv4 does not allow this extension. **(but IPv6 does!)**

Network Layer

Multi-**P**rotocol **L**abel **S**witching (**MPLS**)

Solution: Encapsulate the IPv4 datagram in an MPLS packet.
Here ***MPLS*** will simulate a layer between the Data Link Layer and the Network Layer

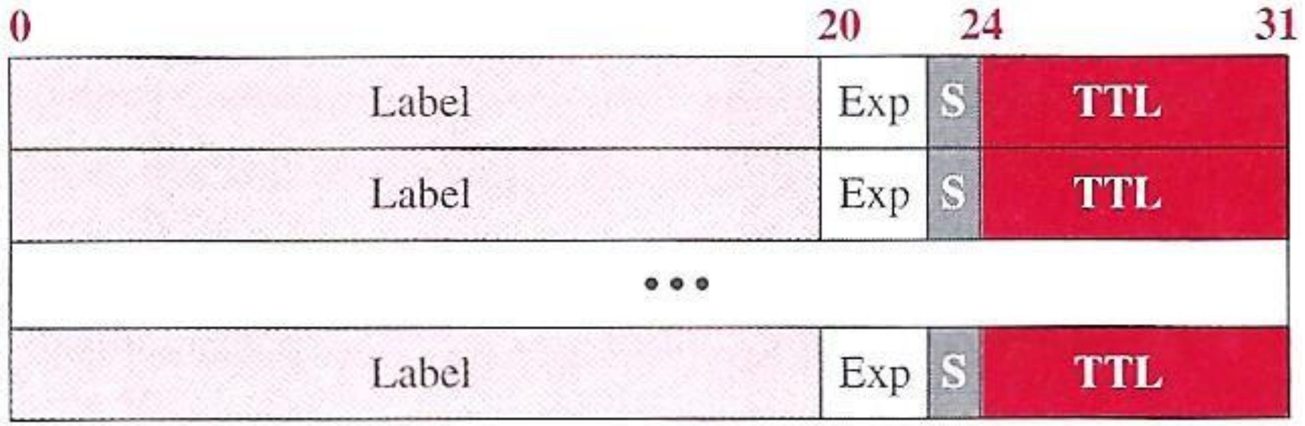
The whole IP packet is encapsulated as the payload in an MPLS packet and an MPLS header is added.



Network Layer

<u>M</u>ulti-<u>P</u>rotocol <u>L</u>abel <u>S</u>witching (**MPLS**)

In fact, the MPLS header is a stack of sub-headers used for multilevel hierarchical switching.



Label field: This 20-bit long field defines the label used to index the forwarding table in the router
Exp field: This 3-bit field is reserved for <u>experimental purposes</u>.
S field: This field of only 1-bit indicates the end (S = 1) of the stack of sub-headers.
TTL field: This 8-bit long field is similar to the **TTL** (<u>**T**</u>ime <u>**T**</u>o <u>**L**</u>ive) field in an IP datagram. Each visited router decrements the value in this field by one. When it becomes 0, the packet is discarded to avoid looping.

Network Layer

Multi-**P**rotocol **L**abel **S**witching (**MPLS**)

A Stack of labels in **MPLS,** allows hierarchical switching.
This is similar to conventional hierarchical routing (as we have seen before)

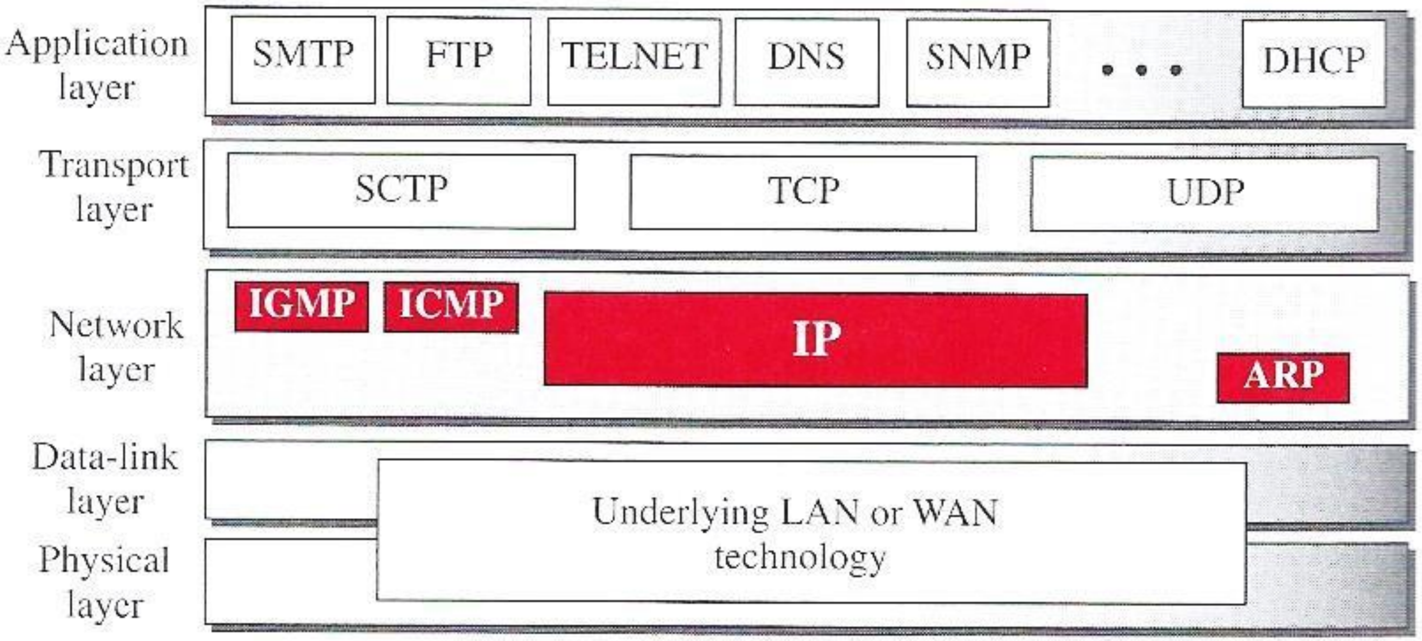
E.g., a packet with two labels can use the top label to forward the packet through switches outside an organization; the bottom label can be used to forward the packet inside the organization to reach the destination subnet.

Internet Protocol (IPv4)

Network Layer

Internet Protocol **IPv4**

The network layer in version 4 can be thought of as one main protocol and three auxiliary ones.



IPv4 is responsible for packetizing, forwarding, and delivery of a packet at the network layer.
Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery.
Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting.
Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses.

Network Layer

Internet Protocol **IPv4**

IPv4 is an unreliable and connectionless datagram protocol - *a* ***best-effort delivery*** service.

The term best-effort means that IPv4 does not offer any flow and error control, and that packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network (however, error detection on the header).

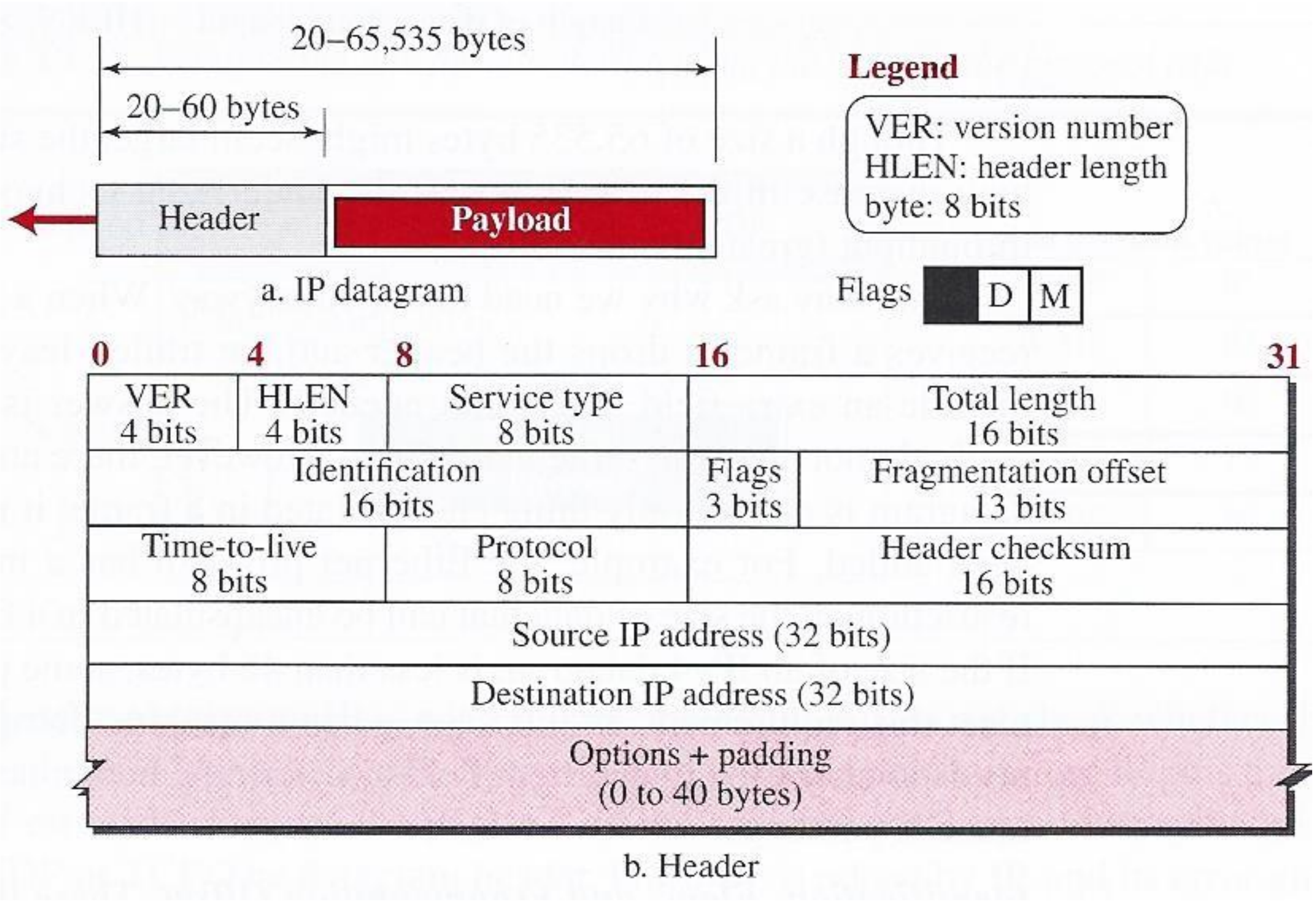
IPv4 recognizes the unreliability of the lower layers and does its best to get its transmission to the receiver, <u>but without any kind of guarantee</u>.

If reliability is important, then IPv4 must be paired with a reliable transport layer protocol e.g., **TCP**.

- Datagrams are used and each datagram is handled independently.
- This means that they do not necessarily follow the same path and therefore may arrive out-of-order at the receiver.
- Packets may also be damaged or lost. But IPv4 relies on the higher-level protocols to take care of these problems.

Network Layer

Internet Protocol **IPv4** – Header format



a. IP datagram

b. Header

A datagram is a packet of variable length. It consists of two parts: header and data.

Network Layer

Internet Protocol **IPv4** – Header format

The header is 20 to 60 bytes in length and contains information essential to routing and delivery.

- **Version (Ver)**: is a 4-bit field that defines the version of the IPv4 protocol, which obviously has the value of 4. This field tells the IPv4 software that all fields in datagrams must be interpreted as specified in the IPv4 protocol.

- **Header length (HLEN)**: is a 4-bit field that indicates the total length of the datagram header in 4-byte words.
  This field is necessary as the header can have a variable size.
  If there are no options, then the header takes up 20 bytes, and the value of HLEN will thus be 5 (5 x 4 bytes = 20 bytes).

Network Layer

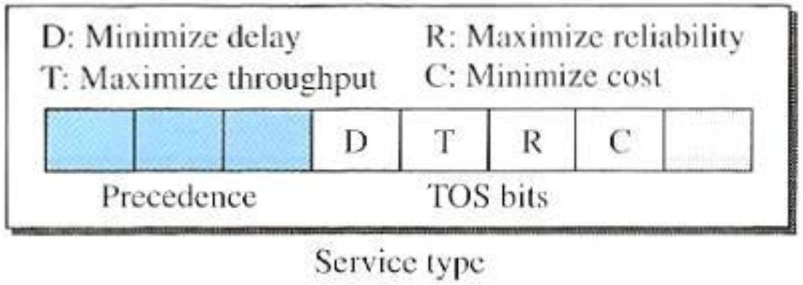
Internet Protocol **IPv4** – Header format

- **Service**: This field has had both name and interpretation changed. The field is 8 bits and was previously called **Service type**, now it is called **differentiated service**.

  The field is used to distinguish and prioritize data transmission in case of congestion.

Network Layer

Internet Protocol **IPv4** – Header format (only for information)



- **Service type (former meaning)**
  In this implementation, the 3 MSBs (bits) are called Precedence bits, the next 4 bits are called type of service (TOS) and the last bit is not used.

  a) **Precedence**: This 3 bit subfield defines the priority of the datagram in case of congestion (traffic).

     If a router p.g.a. congestion has to discard datagrams, then it will be the ones with the lowest priority.
     Datagrams used for e.g., network management, are more important than other datagrams.

Network Layer

Internet Protocol **IPv4** – Header format (only for information)

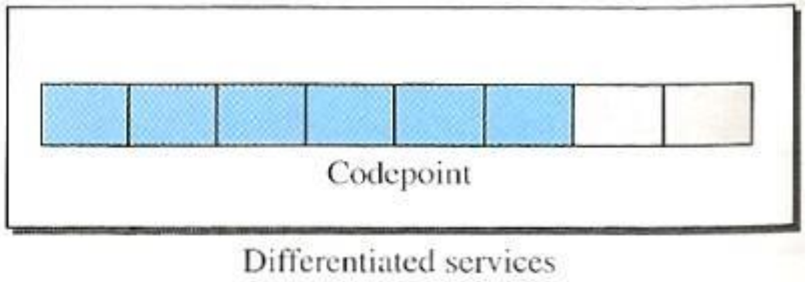
b) **TOS bits**: This 4-bit subfield has a special meaning, note that only one bit at a time can be 1.

TOS bits	Description
0000	Normal (default)
0001	Minimize cost
0010	Maximize reliability
0100	Maximize throughput
1000	Minimize delay

- Interactive activities, activities that require immediate attention and activities that require quick response must have a **minimum delay**.
- Activities that send a lot of data must have **maximum throughput**.
- Management activities must have **maximum reliability**.
- Background activities must have a **minimum cost**.

Network Layer

Internet Protocol **IPv4** – Header format (only for information)



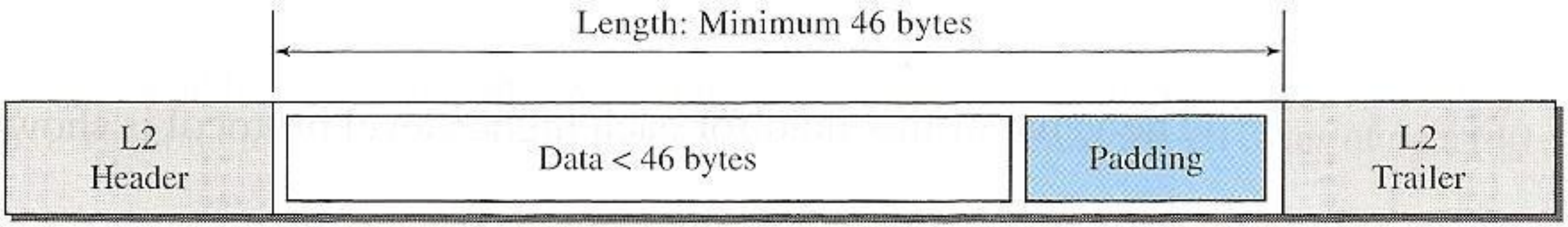
- **differentiated service** **(current meaning)**
  In this implementation, the 6 MSB (bits) form a subfield called codepoint, the last two bits are not used.
  a) When the 3 LSBs (bits) are all 0, then the 3 MSBs (bits) are interpreted in the same way as in the Percedence subfield in the Service Type implementation.
  b) When the 3 LSBs (bits) are not all 0, the 6 bits define 64 different service types, based on their priority. Category 1 contains 32 service types, Categories 2 and 3 contain 16 service types. The priority of these types of services is determined by the **IETF** (**I**nternet **E**ngineering **T**ask **F**orce)

Category	Codepoint	Authority
1	XXXXX0	Internet (IETF)
2	XXXX11	Local
3	XXXX01	Temporary or experimental

Network Layer

Internet Protocol **IPv4** – Header format

- **Total Length**: This field is 16 bits and indicates the total length of the datagram (both header and data).
  Since this field in IPv4 is 16 bits, the length of the datagram is limited to 65,535 (2^{16}-1) bytes, from which 20 to 60 bytes must be subtracted because of the header.
  The rest is data from the upper layer. If the length of a datagram becomes less than the minimum length of the protocol used on the Data Link layer e.g., Ethernet, then fills up with padding ("Data < 46 bytes" corresponds to the Datagram).



Network Layer

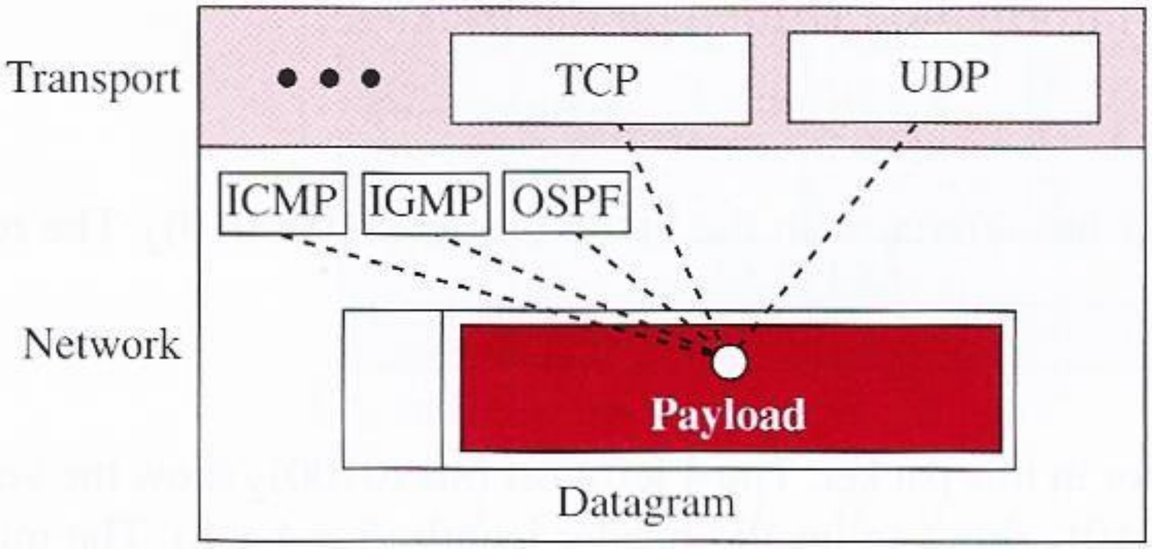
Internet Protocol **IPv4** – Header format

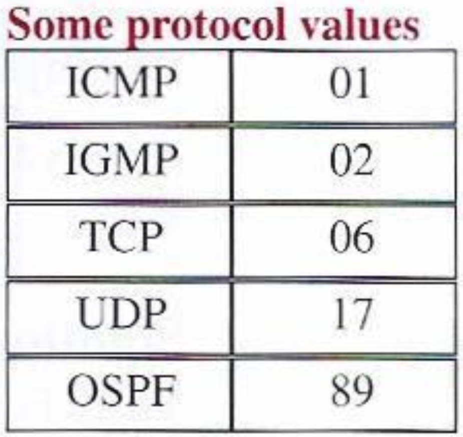
- **Identification**: This field is used in connection with fragmentation (we will see later).
- **Flags**: This field is used in connection with fragmentation (we will see later).
- **Fragmentation offset**: This field is used in connection with fragmentation (we will see later).
- **Time to Live**: A datagram has a limited lifetime when transmitted around the Internet to avoid circulating on the Internet forever.
  This value is approximately two times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

Network Layer

Internet Protocol **IPv4** – Header format

- **Protocol**: This 8-bit field indicates which protocol the payload (packet) carried by the IPv4 datagram comes from.
  An IPv4 datagram can encapsulate data from many different protocols, e.g., **TCP**, **UDP**, **ICMP** and **IGMP**.
  The field indicates who is the final receiver of the data that the IPv4 datagram provides, and thus which protocol the data belongs to.





Some protocol values

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

Network Layer

Internet Protocol **IPv4** – Header format

The value in the Protocol field for different protocols

Value	Protocol	Name
1	ICMP	**I**nternet **C**ontrol **M**essage **P**rotocol
2	IGMP	**I**nternet **G**roup **M**anagement **P**rotocol
6	TCP	**T**ransmission **C**ontrol **P**rotocol
17	UDP	**U**ser **D**atagram **P**rotocol
89	OSPF	**O**pen **S**hortest **P**ath **F**irst (routing protocol)

Network Layer

Internet Protocol **IPv4** – Header format

- **Checksum**: We will look at the Checksum concept and its calculation later.

- **Source Address**: The sender address is a 32-bit field that indicates the sender. This field must remain unchanged throughout the transmission through the Internet from sender to receiver.

- **Destination address**: the receiver address is a 32-bit field that indicates the receiver. This field must remain unchanged throughout the transmission through the Internet from sender to receiver.

Network Layer

Internet Protocol **IPv4** – Checksum

IPv4 checksum is performed as follows:

- The checksum is initialized to ***0***.
- The entire IPv4 header is divided into 16-bit sections/words.
- All these sections are added together.
- The resulting sum is complemented.
- This value is inserted into the header.

The checksum in an IPv4 datagram includes only the header, not data!

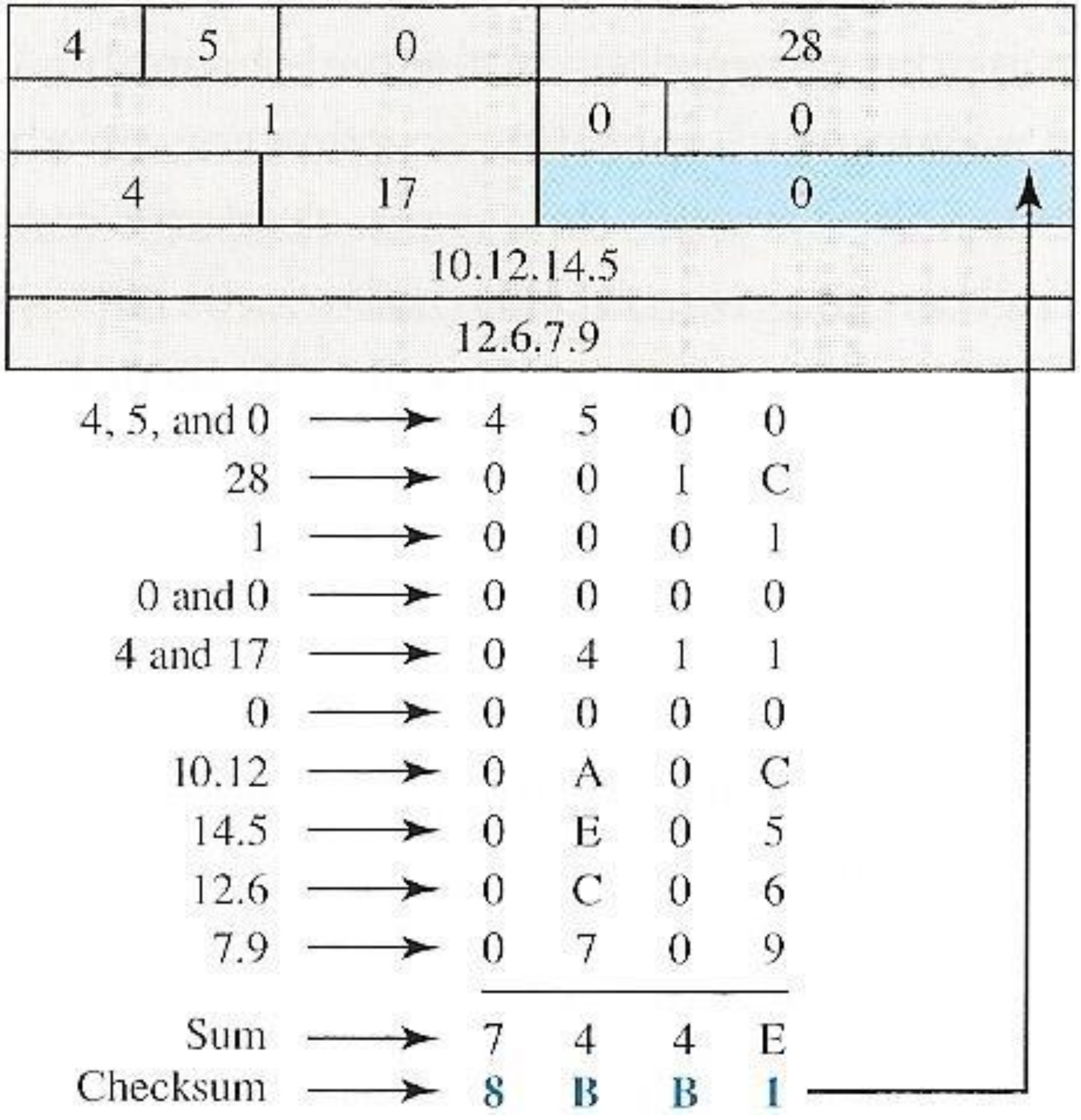
There are two reasons for this:

- All upper-level protocols have their own checksum field that covers the data they encapsulate into an IPv4 datagram.
- Since the header in an IPv4 datagram changes every time it passes a router, and if the data (payload), which does not change, was also included in the checksum, then the entire datagram had to be recalculated. Now you can just calculate the checksum for what is actually changed (the header).

Network Layer

Internet Protocol **IPv4** – Checksum Example

Wrapped Sum = Sum mod FFFF
Checksum = FFFF - Wrapped Sum



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

Network Layer

Fragmentation

A datagram can travel through many different networks.

Every router it encounters:

- decapsulates the IP datagram from the frame it receives
- processes it
- encapsulates it back in another frame.

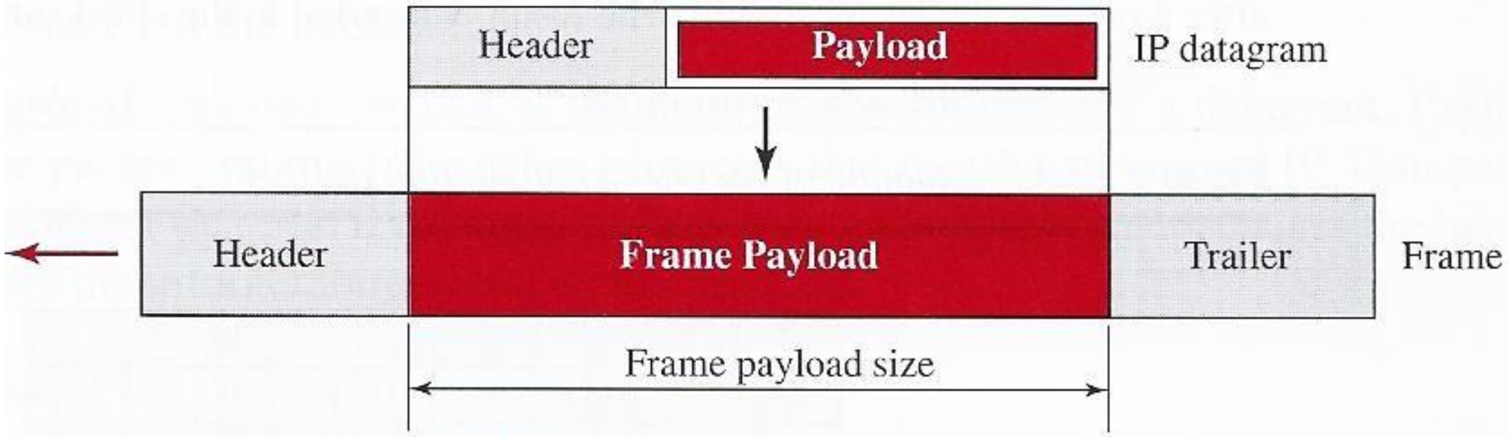
The format and size of the received frame depends on the protocol used by the physical network where the frame came from.

The format and size of the sent frame depends on the protocol used by the physical network where the frame is going to travel.

Example: If a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Network Layer

Fragmentation - Maximum Transfer Unit (MTU)



MTU: Maximum size of frame payload

MTU for some selected networks	
Protocol	MTU
Hyperchannel	65.535
Token Ring (16Mbps)	17.914
Token Ring (4Mbps)	4.464
FDDI (optical fiber)	4.325
Ethernet	1.500
X.25	576
PPP	296

Because of the MTU restrictions, the datagram must be divided to make it possible to pass different networks. This is called **Fragmentation**.

Network Layer

Fragmentation

A datagram can be fragmented by the source or any router in the path.

- It is fragmented into sizes that meet the requirement of the used data link layer in the network in terms of MTU constraint.

Once a datagram is fragmented, each fragment is independent and will have its own header

- Where most fields are the same.
- But some fields have been changed.

A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram may be fragmented several times before it reaches the final destination.

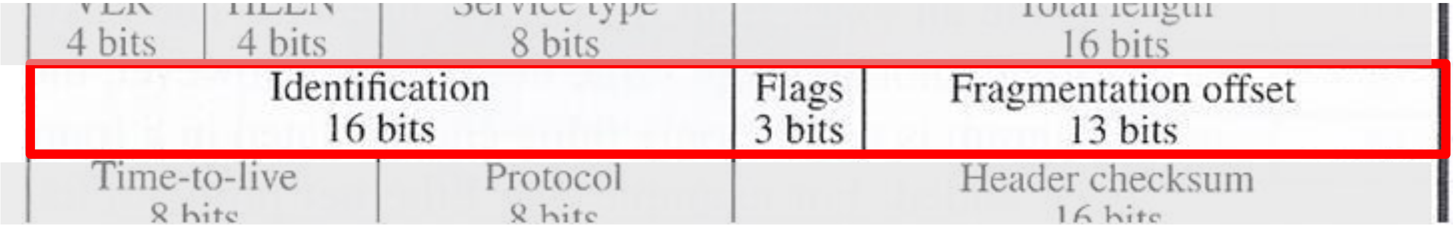
Reassembly of the fragmented datagram is done only at the receiver (not the routers).

Network Layer

Fragmentation

The IPv4 datagram has 3 fields related to fragmentation and reassembly:

- Identification field.
- Flags field.
- Fragmentation offset field.



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	

The host or router that fragments a datagram must change the values of three fields: **flags**, **fragmentation offset**, and **total length**. The rest of the fields must be copied. Of course, the value of the checksum must be recalculated regardless of fragmentation.

Network Layer

Fragmentation: The identification field

This 16-bit identification field contains an identification number to identify the source host where the datagram is generated.

- Every datagram sent out from the host has a unique number.
- if a datagram is fragmented, then all fragments get the same identification number copied from the original datagram.

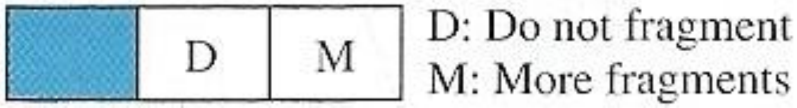
This number helps the receiver to reassemble a datagram. i.e., all fragments with the same number must be assembled into a datagram.

Network Layer

Fragmentation: Flags field

This field contains 3 bits.

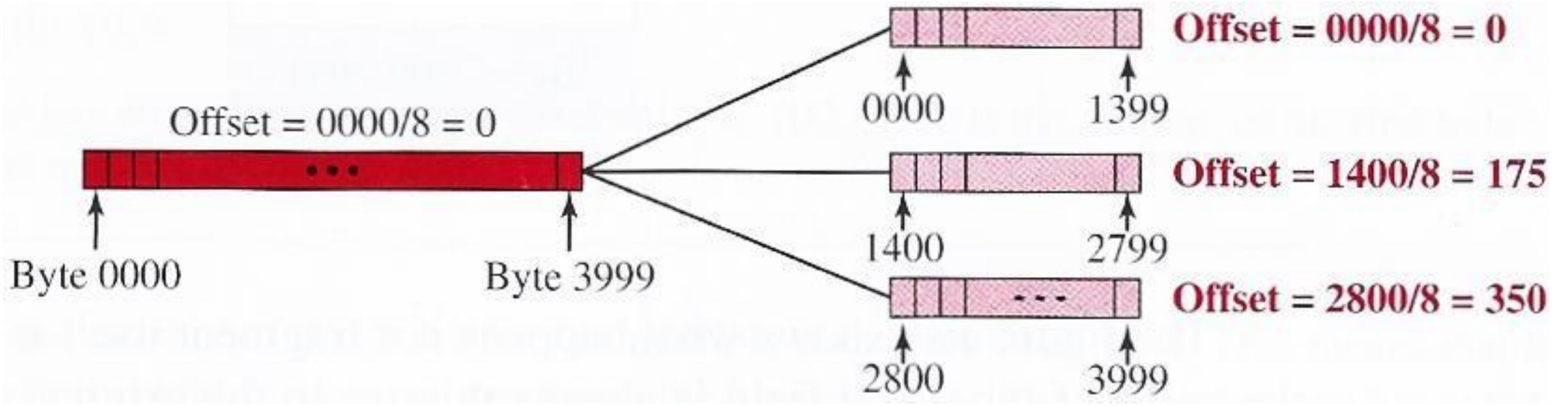
- The first bit is reserved (not used).
- The second bit is called ***Do not fragment bit***.
  - If this value is 1, then the datagram must not be fragmented. If the datagram cannot be delivered via some of the available physical networks, it is discarded and an **ICMP** (**I**nternet **C**ontrol **M**essage **P**rotocol) error message is sent to the sender.
  - If the value is 0, then the datagram can be fragmented if necessary.
- The third bit is called ***morefragment bit***.
  - If the value is 1, then it means that the current datagram is not the last one.
  - If the value is 0, then it means that the current datagram is the last one (or the only one).



Network Layer

Fragmentation: The fragmentation offset field

This field contain 13 bits and indicates the position of the current fragment relative to the whole datagram. The position is measured in units of 8 bytes.



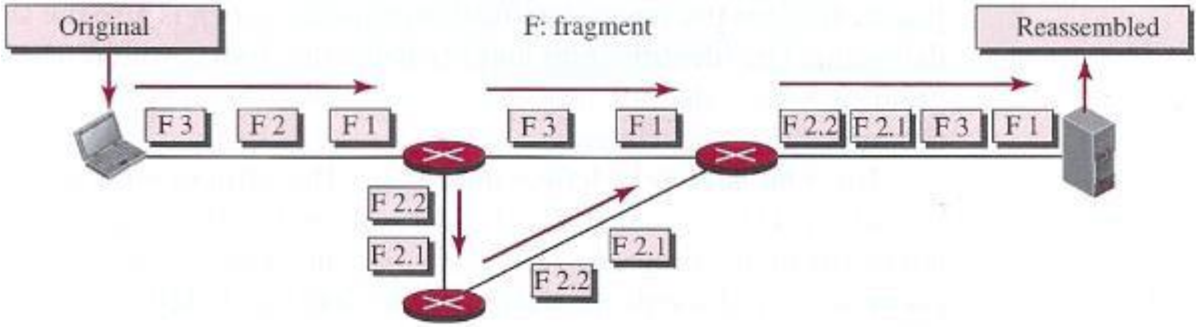
The value of the offset is measured in units of 8 bytes. This is done because the length of the offset field is only 13 bits long and cannot represent a sequence of bytes greater than 8191, which does not match the datagram's maximum length of 65,535.

Therefore, the offset is specified in units of 8 bytes (***8 x 8191 = 65.535 bytes***).

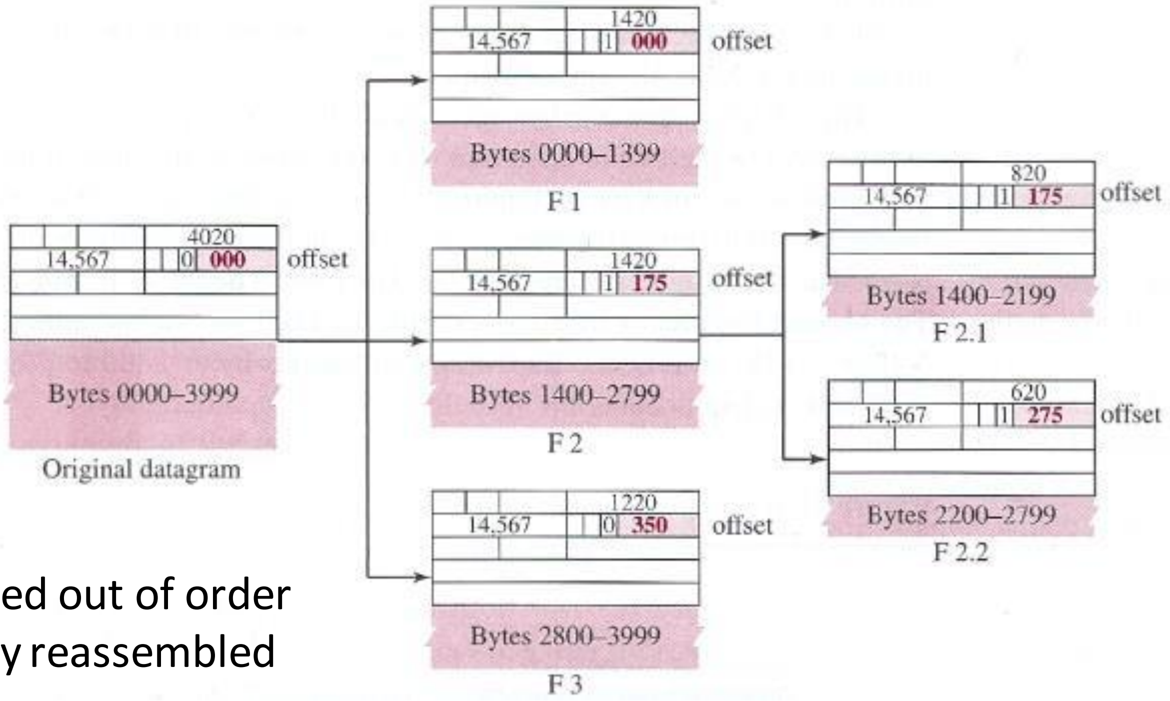
This forces hosts or routers that fragment datagrams to choose the size of each fragment so that the first byte number is divisible by 8.

Network Layer

Fragmentation: Example



- The value of the identification field is the same in all fragments (14,567)
- The more bit is set to 1 for all fragments except the last one.
- The value of the offset field for each fragment is shown. if a fragment itself is fragmented, the value of the offset field is always relative to the original datagram (see F2.1 and F2.2).



Note that although the fragments arrived out of order at the destination, they can be correctly reassembled by using the following strategy.

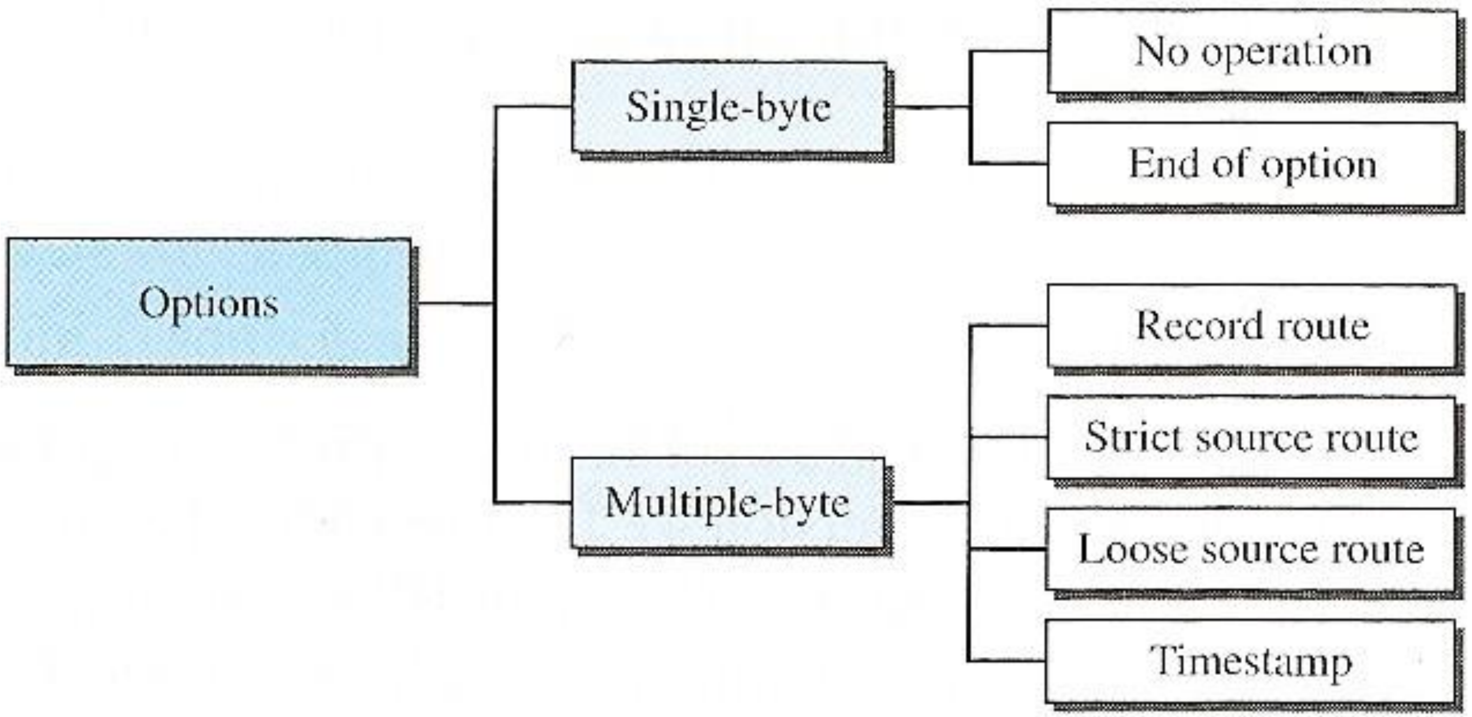
1. The first fragment has an offset field value of zero.
2. Divide the length of the first fragment by 8 to get the offset value of the second fragment.

3. Repeat step 2 until a fragment with the more bit value of 0.

Network Layer

Internet Protocol **IPv4** – Options section

The fixed part of the header is 20 bytes. In addition, we have a variable part of 40 bytes, which we call **option section**.



Options are divided into two broad categories: single-byte options and multiple-byte options.

Network Layer

Internet Protocol **IPv4** – Options section

- **No Operation**: this is a 1-byte option used as a filler between options.
- **End of Option**: this is a 1-byte option used for padding at the end of the option field. It can only be used as the last option.
- **Record Route**: A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

Network Layer

Internet Protocol **IPv4** – Options section

- **Strict Source Route**: This option is used by the sender to predetermine the route that a datagram should take through the Internet.

  This selection of the route by the senders can have several purposes:

  - The sender can select a route with specific services, such as minimum delay or maximum throughput.

  - The sender can choose a route that is safer and more reliable. E.g., you can choose a route so that you do not use the competitors' networks.

  If the sender specifies a specific route, then all routers mentioned in this option must be visited.

  A router must not be visited if it is not mentioned in this option.

  If a datagram visits a router that is not in the list, it is discarded.

  If a datagram has not visited all the routers in the list when one arrives at the receiver, it will also be discarded.

Network Layer

Internet Protocol **IPv4** – Options section

- **Loose Source Route**: This option is similar to the Strict Source Route but is less strict. All routers in the list must be visited, but other routers can also be visited.

- **Timestamp**: A timestamp option is used to record the time of datagram processed by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time. Knowing the time when a datagram is processed can help users and managers track the behavior of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another.

Network Layer

Security in IPv4 datagrams

Although IP security is not part of this course, we will still briefly look at some security issues in the IP protocol and the corresponding solutions that exist.

There are three issues that particularly applicable to IP protocol:

- Packet sniffing
- Package modification
- IP spoofing

Network Layer

Security in IPv4 datagrams - Packet sniffing

Packet sniffing is a passive attack where the attacker does not change the contents of the packets.

An intruder can intercept an IP packet and make a copy of it.

This type of attack is very difficult to detect as the sender and receiver will probably never discover that packets have been copied.

Although packet sniffing cannot be stopped, encryption of the packet can make the attacker’s effort useless. The attacker may still sniff the packet, but the content is not detectable.

Network Layer

Security in IPv4 datagrams - Packet modification

This type of attack is more active.

Here, the attacker intercept the packets, changes their contents, and forwards the modified packets to the receiver.

The receiver is deceived into believing that the packages come from the original sender.

This type of attack can be detected using a ***data integrity mechanism***.

Before a receiver opens and uses the contents of the message, this mechanism is used to ensure that the packet has not been modified during transmission.

Network Layer

Security in IPv4 datagrams - IP spoofing

An attacker could disguise himself as another person and create an IP packet that carries the source address of another computer.

The attacker can e.g., send such IP packet to a bank and pretend to be one of the bank's customers.

This type of attack can be avoided by using a ***source authentication mechanism***.

Network Layer

Security in IPv4 datagrams - IPSec

Today, IP packets can be protected from the above attacks by using the **IPSec** (IP Security) protocol.

This (IPSec) protocol, used in conjunction with the IP protocol, creates a ***connection-oriented service between two hosts***, in which IP packets can be exchanged without having to worry about the attacks as described.

IPSec adds the following services:

- **Algorithms and keys.** Two hosts that want to establish a secure channel between them can agree to use available algorithms and keys for security purposes.
- **Packet encryption.** The packets exchanged between two parties can be encrypted for privacy. This can be done with encryption algorithms and a shared key, which have been agreed in the first step. This makes packet sniffing useless.
- **Data integrity.** Data integrity guarantees that packets have not been modified during the transmission. If a package does not pass a data integrity test, it will be discarded! This protects against packet modification attacks.
- **Source authentication**. IPSec can authenticate the origin of the packet to be sure that the packet is not created by an imposter. This can prevent IP spoofing attacks.

Internet Control Message Protocol (ICMPv4)

Network Layer

ICMPv4

As we have seen before, IP protocol delivers an unreliable and connectionless datagram.

The IP protocol has no error-reporting or any error-correcting mechanism.

- What happens if a router has to discard a datagram because it cannot find a router to the final destination?
- What happens if the time-to-live field reaches the value 0?
- What happens if the final destination host must discard the received fragments of a datagram because it has not received all fragments within a predetermined time limit?

The IP protocol also lacks a mechanism for management queries:

- A host sometimes needs to determine if another host or a router is alive or not.
- Sometimes a network manager needs information from another host or router.

Internet Control Message Protocol (ICMP) is designed to compensate for these issues. One can consider ICMP as a companion to the IP protocol.

Network Layer

ICMPv4 – Message types

ICMP messages are divided into two broad categories:

- Error-reporting messages.
- Query messages.

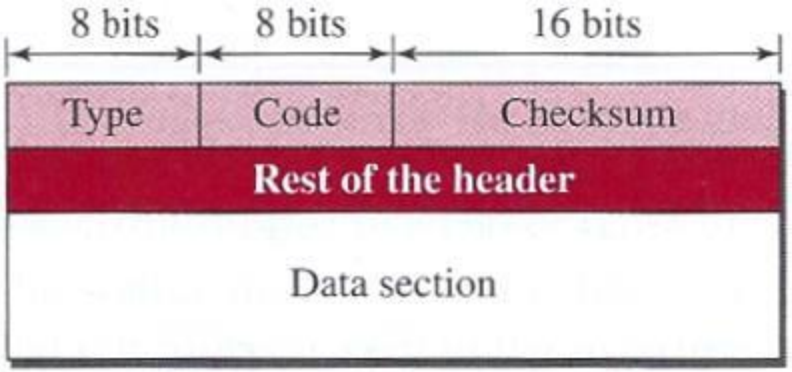
Error-reporting messages: report problems that a router or a host (destination) may encounter when it processes an IP packet.

Query messages: help a host or a network manager get specific information from a router or another host. Some examples:

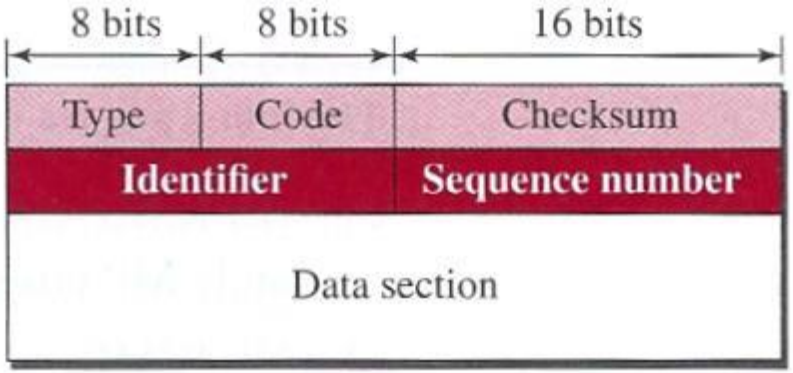
- Nodes can detect and discover their neighbors.
- Hosts can get information about routers found on their network.
- Routers can help a node redirect its messages.

Network Layer

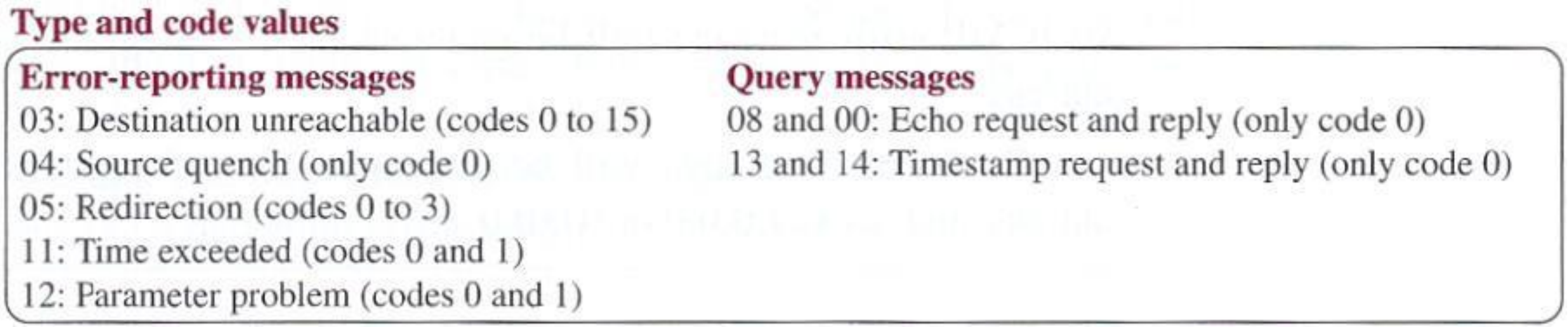
ICMPv4 – Message types



Error-reporting messages



Query messages



Type and code values

Error-reporting messages	**Query messages**
03: Destination unreachable (codes 0 to 15)	08 and 00: Echo request and reply (only code 0)
04: Source quench (only code 0)	13 and 14: Timestamp request and reply (only code 0)
05: Redirection (codes 0 to 3)	
11: Time exceeded (codes 0 and 1)	
12: Parameter problem (codes 0 and 1)	

Type: The 1-byte field indicates the ICMP type.

Code: The 1-byte code field specifies the reason for the particular message type.

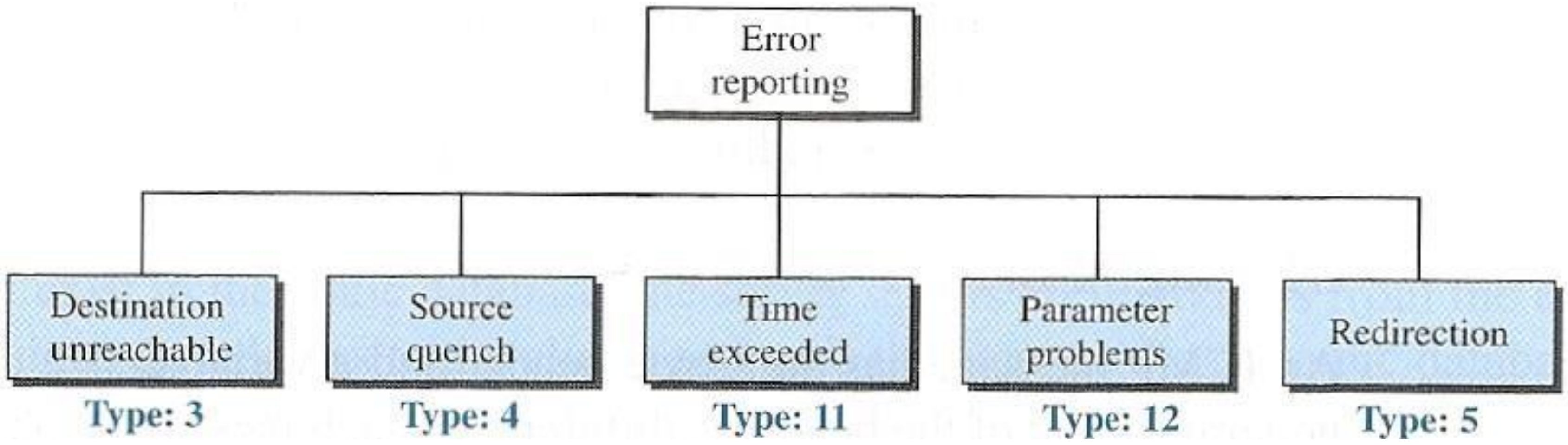
Checksum: This last 2-byte field indicates the checksum (which we will discuss later)

The rest of the header: is specific to each message type.

Network Layer

ICMPv4 – Error reporting

- ICMP is partially designed to compensate for IP deficiencies.
- But ICMP does not correct errors, it simply report them.
- Error correction is left to protocols on the upper layer.
- Error messages are always sent to the original source because the only information available in the datagram about the route is the source and destination IP addresses.



Network Layer

ICMPv4 – Error reporting

Some important guidelines regarding ICMP error messages:

- ✓ **No ICMP error message will be generated based on a datagram which contains an ICMP error message.**
- ✓ **No ICMP error message will be generated if a fragmented datagram is not the first fragment.**
- ✓ **No ICMP error message will be generated from datagrams that have multicast addresses.**
- ✓ **No ICMP error message will be generated from datagrams that have special addresses such as, 127.0.0.0 or 0.0.0.0**

Network Layer

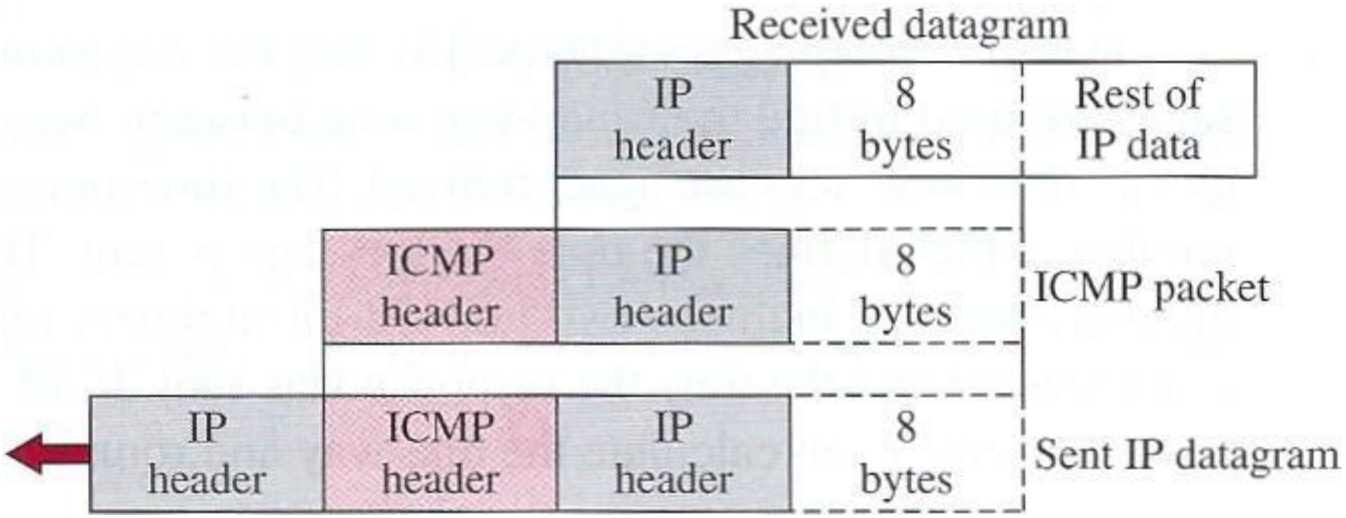
ICMPv4 – Error reporting

One should be aware that all ICMP error messages contain a data section that contains: ***The IP header and the first 8 bytes of the IP data section.***

The IP header is included because it contains the sender address to which the error message is sent.

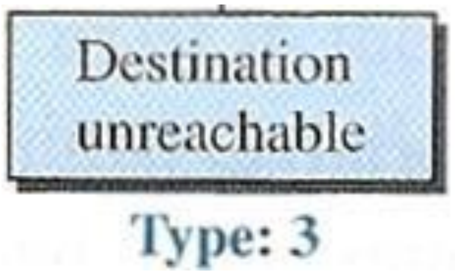
The 8 bytes of data are included because the first 8 bytes provide information about the port numbers (**UDP** and **TCP**) and sequence number (**TCP**). This information is needed so the source can inform the protocols (**TCP** or **UDP**) about the error.

ICMP forms an error packet, which is then encapsulated in an IP datagram.



Network Layer

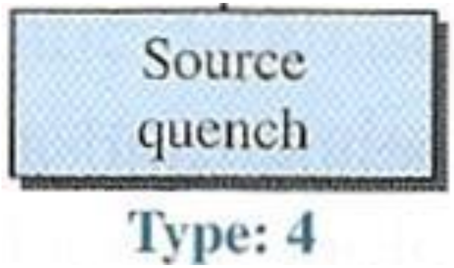
ICMPv4 – Error reporting



Type3: Destination Unreachable: this message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination. For example, code 0 tells the source that a host is unreachable. This may happen, for example, when we use the HTTP protocol to access a web page, but the server is down.

Network Layer

ICMPv4 – Error reporting

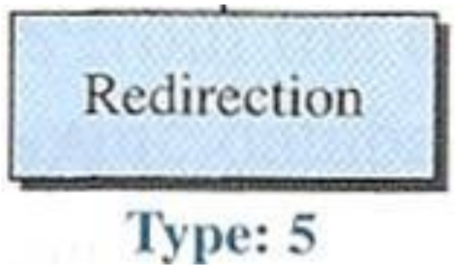


Type4: Source Quench: this message informs the sender that the network has encountered congestion and the datagram has been dropped. The source needs to slow down sending more datagrams.

In other words, ICMP adds a kind of congestion control mechanism to the IP protocol by using this type of message.

Network Layer

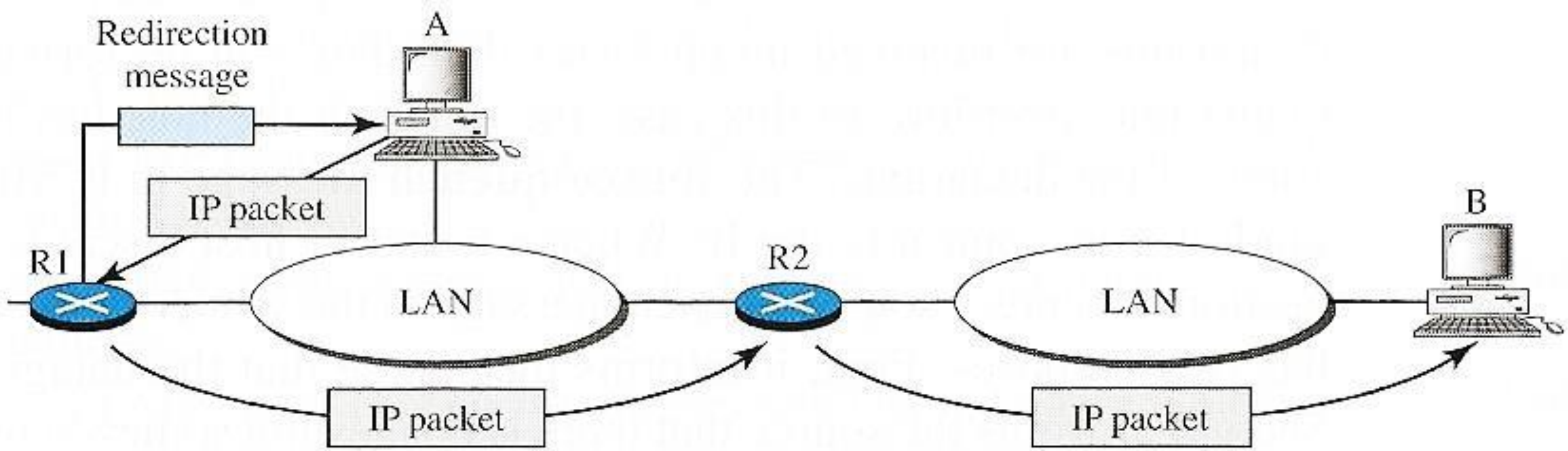
ICMPv4 – Error reporting



Type5: Redirection: this message is used when the source uses a wrong router to send out its message. The router redirects the message to the appropriate router but informs the source that it needs to change its default router in the future. The IP address of the default router is sent in the message.

Network Layer

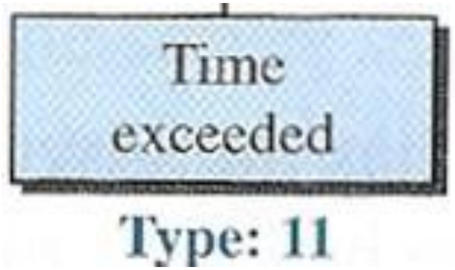
ICMPv4 – Error reporting: Redirection example



- **Host A** wants to send a datagram to **host B**.
- **Router R2** would be the most obvious choice, but host A sends the datagram to **router R1**.
- **Router R1** consults its routing table, and sees that the package should have been for router **R2**.
- **Router R1** sends the packet to **router R2**.
- **Route R1** sends a **TYPE5 ICMP message** (redirection) to **host A**.
- **Host A** updates its routing table with the new routing information.

Network Layer

ICMPv4 – Error reporting



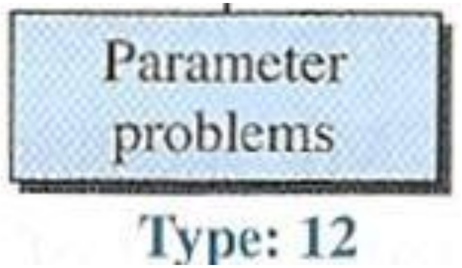
Type11: Time Exceeded: this message can be generated for two reasons:

1. As we have seen before, the IP datagram has a time-to-live (TTL) field that is counted down by 1 each time it passes a router. If a router finds that this value has become 0 after such a countdown, then a message is sent back to the sender.

2. The time-exceeded message can also be sent when not all fragments of a datagram arrive within a predefined period of time.

Network Layer

ICMPv4 – Error reporting



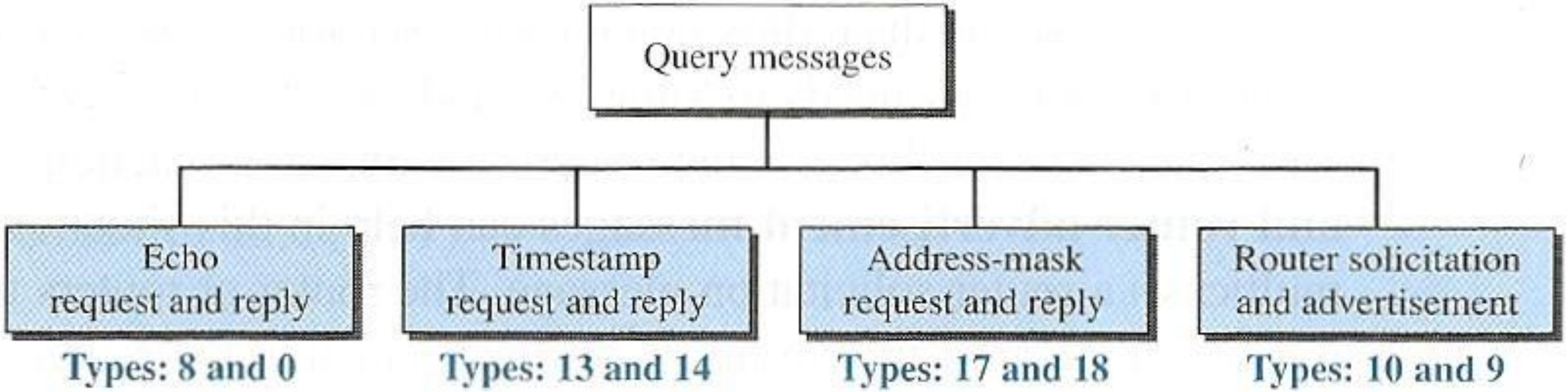
Type12: Parameter problems: Every ambiguity in a datagram's header can create serious problems as it travels through the Internet.

If a router on the way or the receiver detects missing values in any fields in the header, or some options can not be interpreted, then the datagram is discarded, and a message is sent back to the sender.

Network Layer

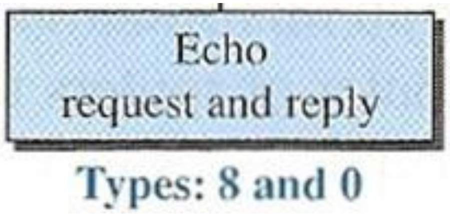
ICMPv4 – Query messages

In addition to error reporting, ICMP can diagnose some network issues. This is achieved via inquiry messages. There are 4 different sets of queries.



Network Layer

ICMPv4 – Query messages



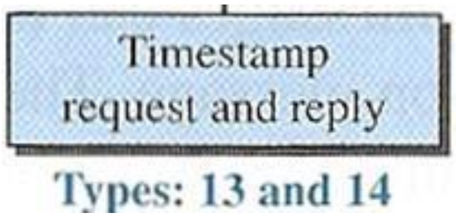
Echo request (type 8) and echo reply (type 0): this pair of messages are used by a host or a router to test the liveliness of another host or router. A host or router sends an echo request message to another host or router; if the latter is alive, it responds with an echo reply message.

Echo request and reply can help determine if there is any communication at the IP level. It also shows that intermediate routers that forward the IP datagram, are working.

We shortly see the applications of this pair in a debugging tool, ping

Network Layer

ICMPv4 – Query messages

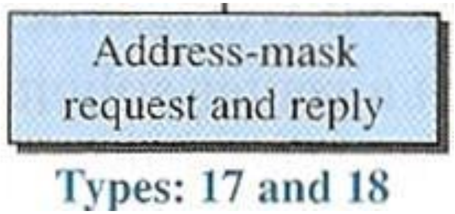


Timestamp request (type 13) and reply (type 14): Two nodes (hosts or routers) can use this request/reply to determine the round-trip time for an IP datagram traveling between them. It can also be used to synchronize the local clocks in the two nodes.

Network Layer

ICMPv4 – Query messages

the message is categorized as obsolete by the **IETF** (**I**nternet **E**ngineering **T**ask **F**orce)!



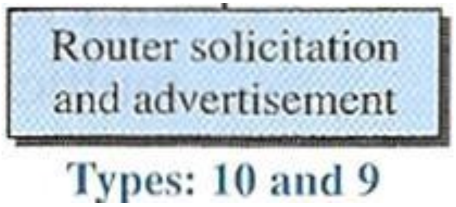
Address mask: Type 17 and 18: A host may know its IP address, but not the corresponding mask.
E.g., if the host has the address 159.31.17.24, but does not know the corresponding mask (e.g., /24).

- To find out his mask, the host sends a request (type 17) to a router on his LAN.
- If the host knows the router's IP address, then the message is sent as unicast, otherwise it is sent as a broadcast.
- The router sends a reply (type 18), with information about the host's mask.
- The mask can be used together with the IP address to determine the address of the subnet. (as we have seen)

Network Layer

ICMPv4 – Query messages

the message is categorized as obsolete by the **IETF** (Internet Engineering Task Force)!



Router solicitation and advertisement

Types: 10 and 9

Router solicitation and advertisement: Type 10 and 9: As we have seen before, in connection with "redirection", a host that wants to send an IP packet to another host on another LAN, know the address of a suitable router..
The host must also know if the router is "alive" and working properly. Here, this type of request /reply can help.

- The host can be type 10 broadcast.
  (broadcast can, as we remember, only take place on the LAN).
- The routers that receive this request broadcast their routings information, that is, ***type 9***.

Alternatively, a router may also periodically broadcast (***advertise type 9***) its information, even if no host has requested (***type 10***) to do so.

Network Layer

Debugging tools

There are many tools that can be used for troubleshooting on the Internet. We can follow the route for packets, we can examine many different host and router types and so on.

But here we will look in particular at two tools:

- ***Ping***
- ***Traceroute***

Network Layer

Debugging tools: Ping

The ping program can be used to determine if a host is alive and is able to respond.
We use the Ping program here to see how it uses ICMP packets.

- The sender sends an ICMP echo request (type = 8 and code = 0)

- The receiver responds with an ICMP echo reply if it is alive

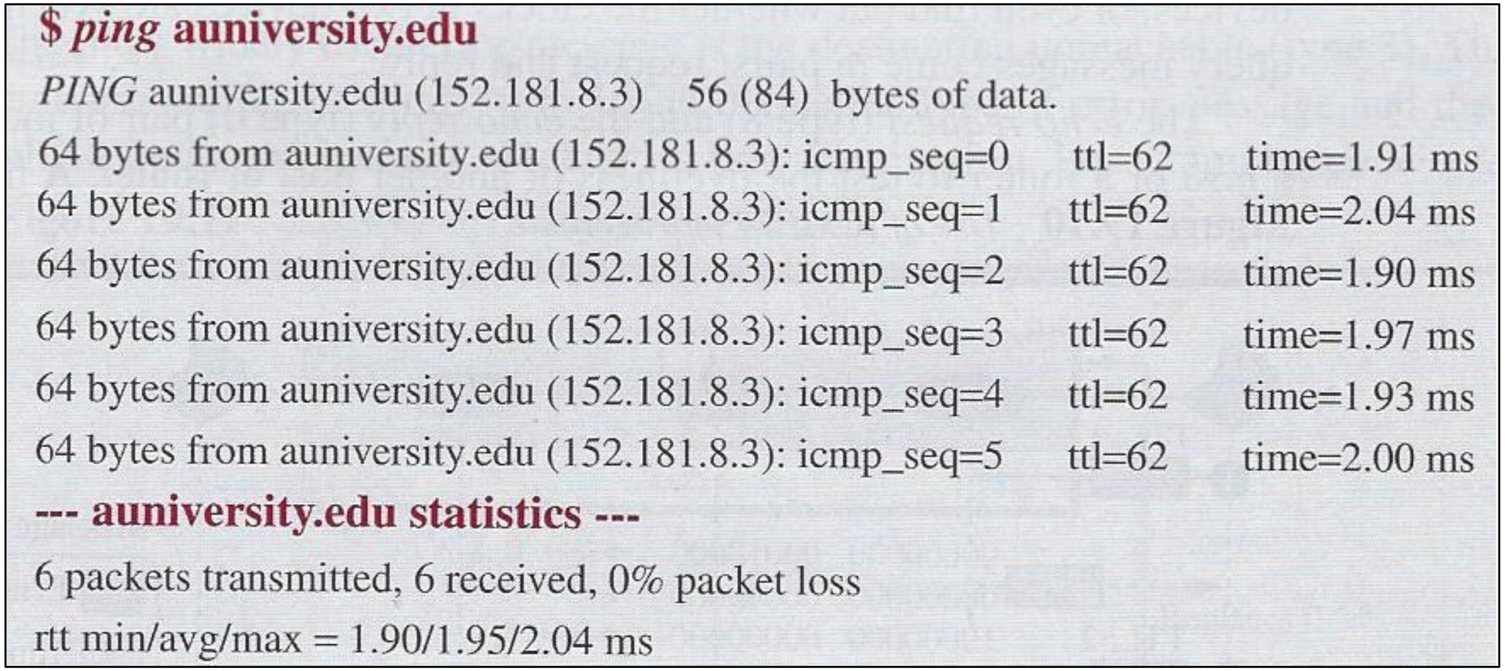
- The ping program now sets the identifier field in the echo request and echo reply messages. (so that you know which messages belong to the sequence)

- The ping program starts the sequence number from 0 and this number is increased by 1 each time a new message is sent.

- The ping program can calculate Round-Trip-Time (RTT). It insert the sending time in the data section of the message. When the message arrives, it subtract the departure time from the arrival time to get RTT.

Network Layer

Debugging tools: Ping



```
$ ping auniversity.edu
PING auniversity.edu (152.181.8.3)    56 (84)  bytes of data.
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=0     ttl=62     time=1.91 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=1     ttl=62     time=2.04 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=2     ttl=62     time=1.90 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=3     ttl=62     time=1.97 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=4     ttl=62     time=1.93 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=5     ttl=62     time=2.00 ms
--- auniversity.edu statistics ---
6 packets transmitted, 6 received, 0% packet loss
rtt min/avg/max = 1.90/1.95/2.04 ms
```

The Time-To-Live (TTL) in the IP header in this example is 62, which means that the datagram can visit up to 62 hops before it is discarded.

The ping program tells you that there are 56 bytes of data out of 84 bytes (total). This corresponds to the **IP datagram**

IP header (20 byte) + ICMP header (8byte) + data (56 byte) = total (84 byte).

Note: each row says 64 bytes! This corresponds to the **ICMP package**

ICMP header (8byte) + data (56 byte) = 64 byte

Network Layer

Debugging tools: Traceroute

This program can be used to trace the path of a packet from a source to the destination. The program is called:

- **Traceroute** in UNIX
- **Tracert** in Windows

The program conveniently uses two ICMP error-reporting messages to figure out the path:

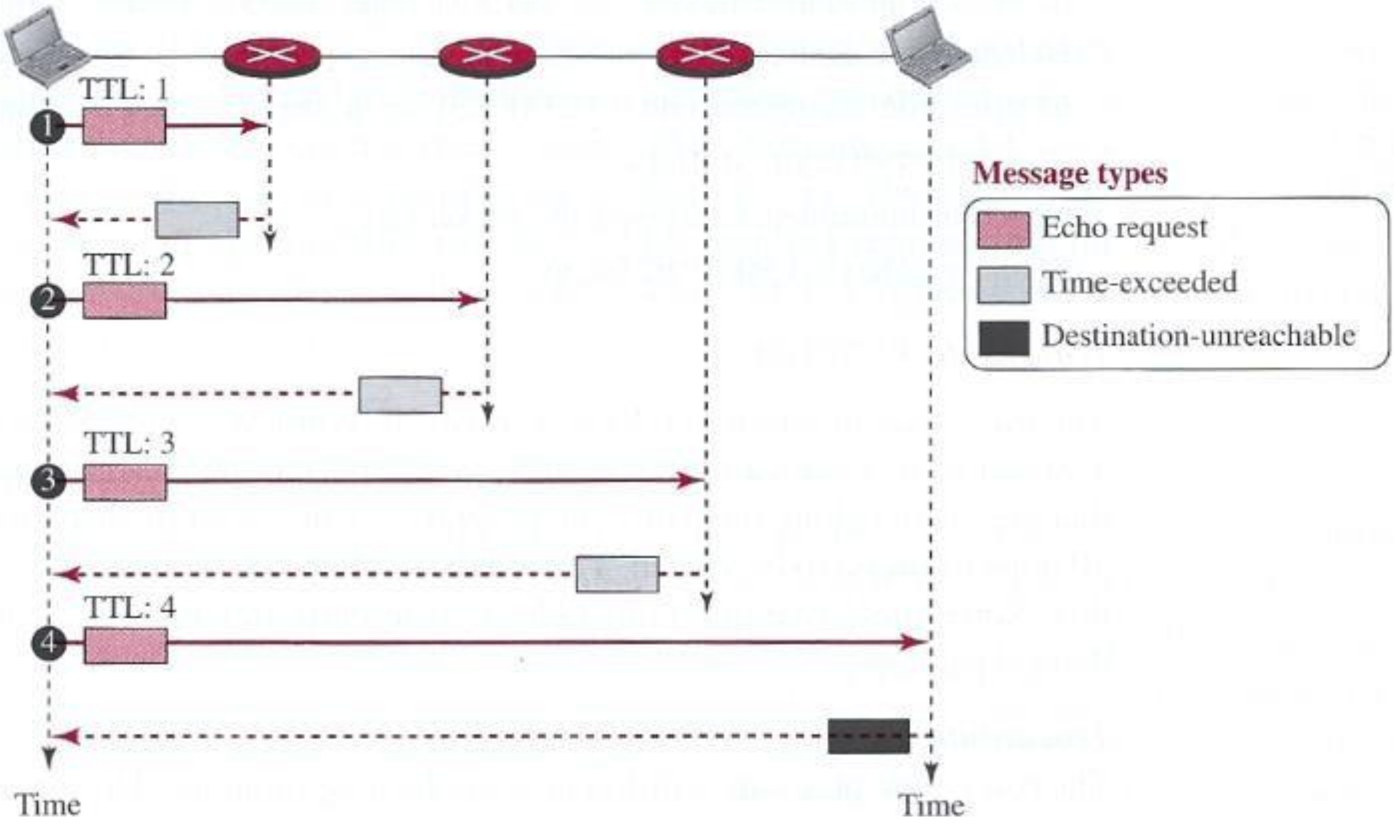
- **Type11: Time Exceeded**
- **Type3: Destination Unreachable**

The program is an application layer program and uses UDP as transport layer protocol.

Network Layer

Debugging tools: Traceroute

The program is an application layer program and uses UDP as transport layer protocol.



Network Layer

Debugging tools: Traceroute

1. The traceroute program uses the following procedure to find the IP address of the first router and the round-trip time between the sender and the first router.
   a. The traceroute program on the sender uses UDP to send a packet to the receiver host. The message is encapsulated in an IP datagram with the value in the time-to-live field set to 1. The program records the departure time of the packet.

   b. The first router receives the packet and counts down the TTL value so that the value becomes 0. The packet is discarded for this reason and an ICMP Time Exceeded message is sent back to the sender.

   c. The traceroute program on the sender receives this ICMP error message, from which the IP address of the first router can be found from the source IP address of the error message.
   The program also record the arrival time. The difference between this time and the departure time from step ***a*** is the round-trip time. The program repeats steps ***a-c*** three times to obtain a better average round-trip time.
   The first time takes longer as ARP has to look up in its table, the second and third time uses the physical address stored in the cache.

Network Layer

Debugging tools: Traceroute

2. The traceroute program repeats steps ***a*** to ***c*** (in step 1) to find the IP address and round-trip time of another router. The value in the TTL field is increased by 1 every time (e.g., ***2*** for the second router, and **3** for the third router).

3. The traceroute program repeats step 2 until the IP address and round-trip time of the receiver host are found.

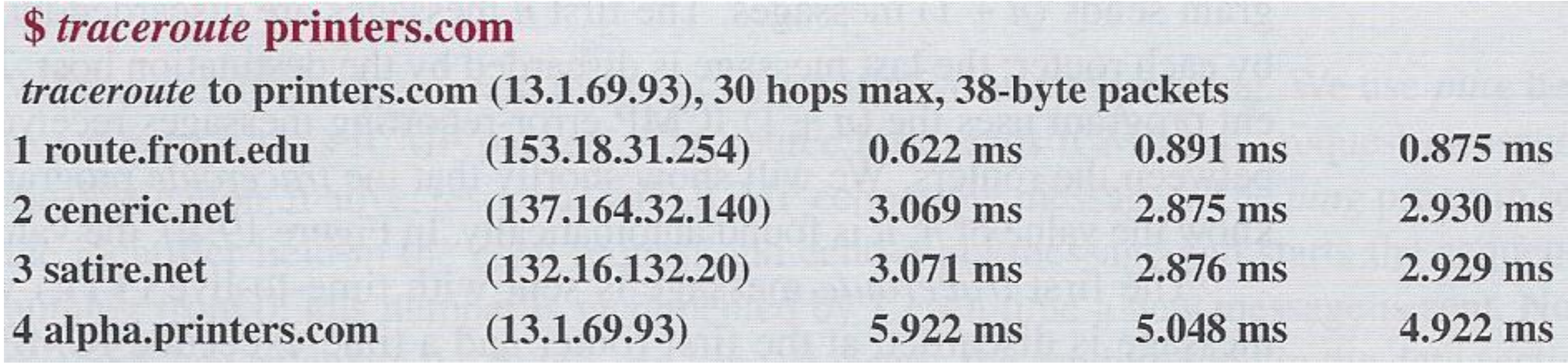
For the sender host to know that the packet has finally reached the receiver host, the traceroute program use UDP packets:

- The program has set a port value in the UDP header that does not exist.
- When the receiver host receives the packet, it cannot find the specified port number in any application layer programs.
- The receiver host therefore sends an ICMP message of the type "The recipient cannot be reached" back to the sender host (Type3: Destination Unreachable, code: 3)

Note that nothing like this happens when the UDP packet arrives at the routers as routers do not check the UDP header.

Network Layer

Debugging tools: Traceroute



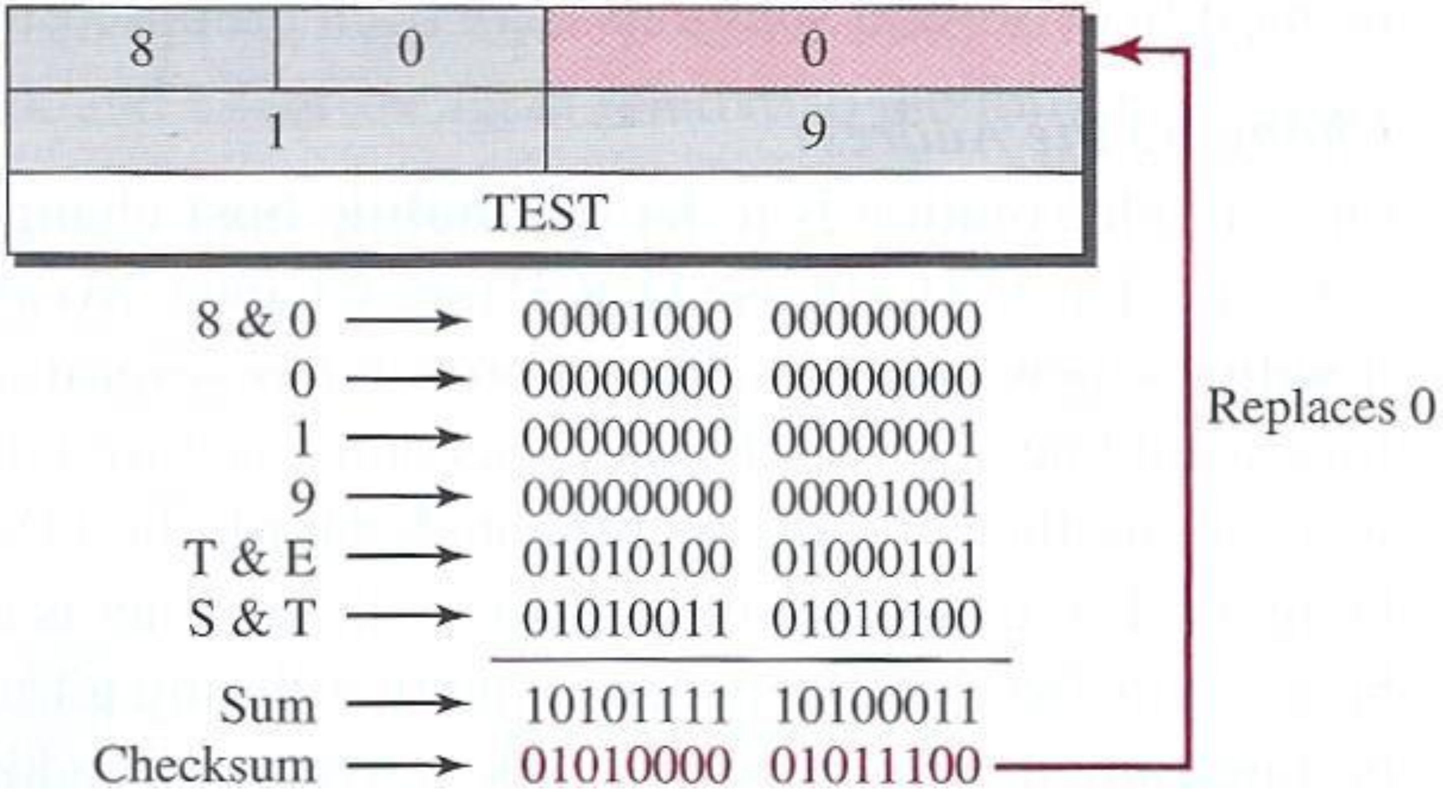
```
$ traceroute printers.com
traceroute to printers.com (13.1.69.93), 30 hops max, 38-byte packets
1 route.front.edu        (153.18.31.254)    0.622 ms      0.891 ms      0.875 ms
2 ceneric.net            (137.164.32.140)   3.069 ms      2.875 ms      2.930 ms
3 satire.net             (132.16.132.20)    3.071 ms      2.876 ms      2.929 ms
4 alpha.printers.com     (13.1.69.93)       5.922 ms      5.048 ms      4.922 ms
```

Network Layer

ICMP Checksum

The Checksum is calculated for the entire message, i.e., both header and data section.

- The checksum field is initially set to 0
- The message is divided into 16-bit sections, which are added together
- The sum is complemented (reversed bit by bit)
- The result is inserted in the Checksum field.


8
0
0
1
9
TEST
8 & 0 → 00001000 00000000
0 → 00000000 00000000
1 → 00000000 00000001
9 → 00000000 00001001
T & E → 01010100 01000101
S & T → 01010011 01010100
Sum → 10101111 10100011
Checksum → 01010000 01011100
Replaces 0