SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

UNIT - I

18CSC302J - COMPUTER NETWORKS

Course Outcome

The purpose of learning this unit is to:

- ***CLR-1 : Describe the importance of various Internet protocols like ARP, RARP, ICMP, Multicasting and multi routing, SCTP.***

At the end of this unit, learners will be able to:

- ***CLO-1 : Identify the basics of different types of network and transport layer protocols.***

INTERNET PROTOCOL VERSION 4

OBJECTIVES:

- To explain the general idea behind the IP protocol and the position of IP in TCP/IP protocol suite.
- To show the general format of an IPv4 datagram.
- To discuss fragmentation and reassembly of datagrams.
- To discuss several options that can be in an IPv4 datagram and their applications.
- To show how a checksum is calculated for the header of an IPv4 datagram at the sending site and how the checksum is checked at the receiver site.

Introduction of IP

The Internet Protocol (IP) is the transmission mechanism used by the TCP/IP protocols at the network layer.

IP is an unreliable and connectionless datagram protocol—a best-effort delivery service.

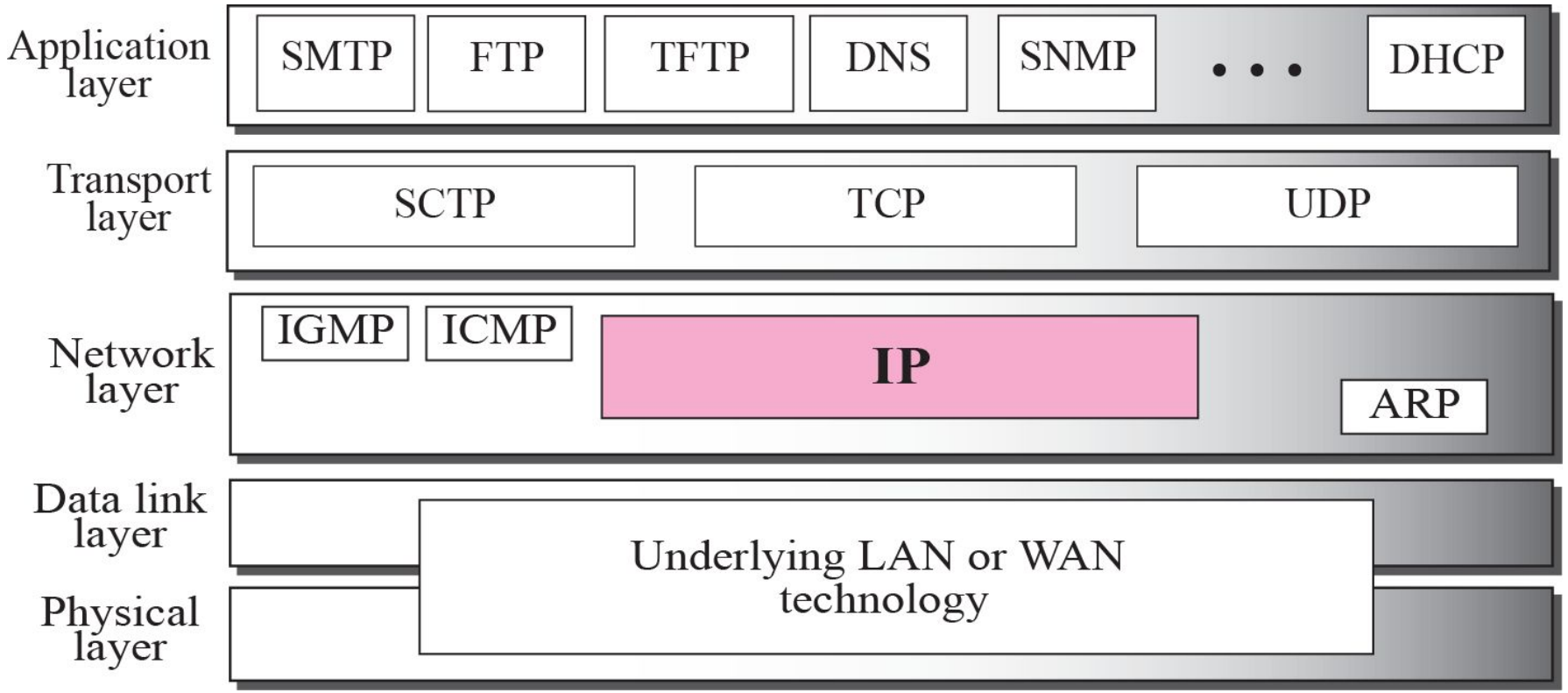
The term best-effort means that IP packets can be corrupted, lost, arrive out of order, or delayed and may create congestion for the network.

IP is also a connectionless protocol for a packet switching network that uses the datagram approach.

This means that each datagram is handled independently, and each datagram can follow a different route to the destination.

This implies that datagrams sent by the same source to the same destination could arrive out of order, some could be lost or corrupted during transmission.

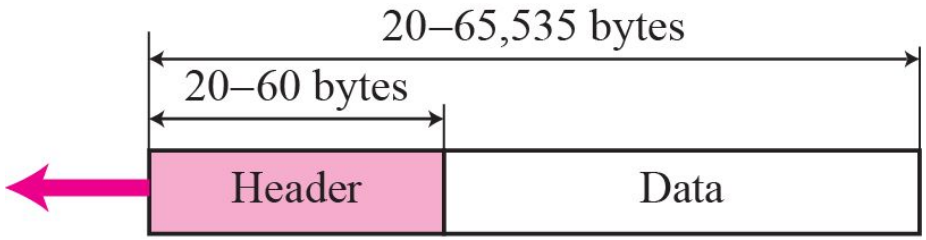
Figure 1.1 **Position of IP in TCP/IP protocol suite**



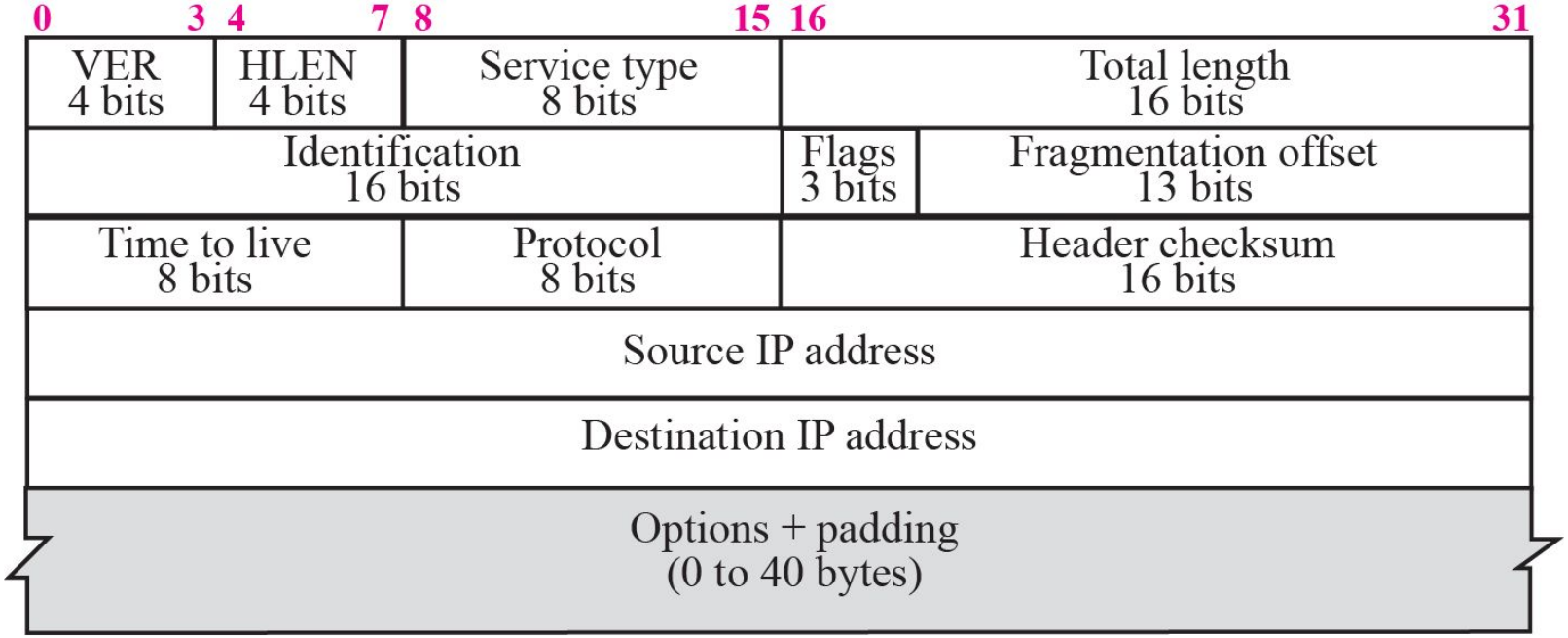
IP Datagrams

Packets in the network (internet) layer are called datagrams.
A datagram is a variable-length packet consisting of two parts: header and data.
The header is 20 to 60 bytes in length and contains information essential to routing and delivery.
It is customary in TCP/IP to show the header in 4-byte sections.
A brief description of each field are on next slide.

Figure 1.2 IP datagram



a. IP datagram



b. Header format

Header Format

Version (VER): This 4-bit field defines the version of the IP protocol. Currently the version is 4.

Header length (HLEN): This 4-bit field defines the total length of the datagram header in 4-byte words. This field is needed because the length of the header is variable (between 20 and 60 bytes).

Service type: In the original design of IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled. This field now defines a set of differentiated services. The new interpretation is shown in Figure 1.3.

The codepoint subfield can be used in two different ways.

a. When the 3 right-most bits are 0s, the 3 left-most bits are interpreted the same as the precedence bits in the service type interpretation.
b. When the 3 right-most bits are not all 0s, the 6 bits define 56 (64 − 8) services based on the priority assignment by the Internet or local authorities according to Table 1.1.

Figure 1.3 Service type

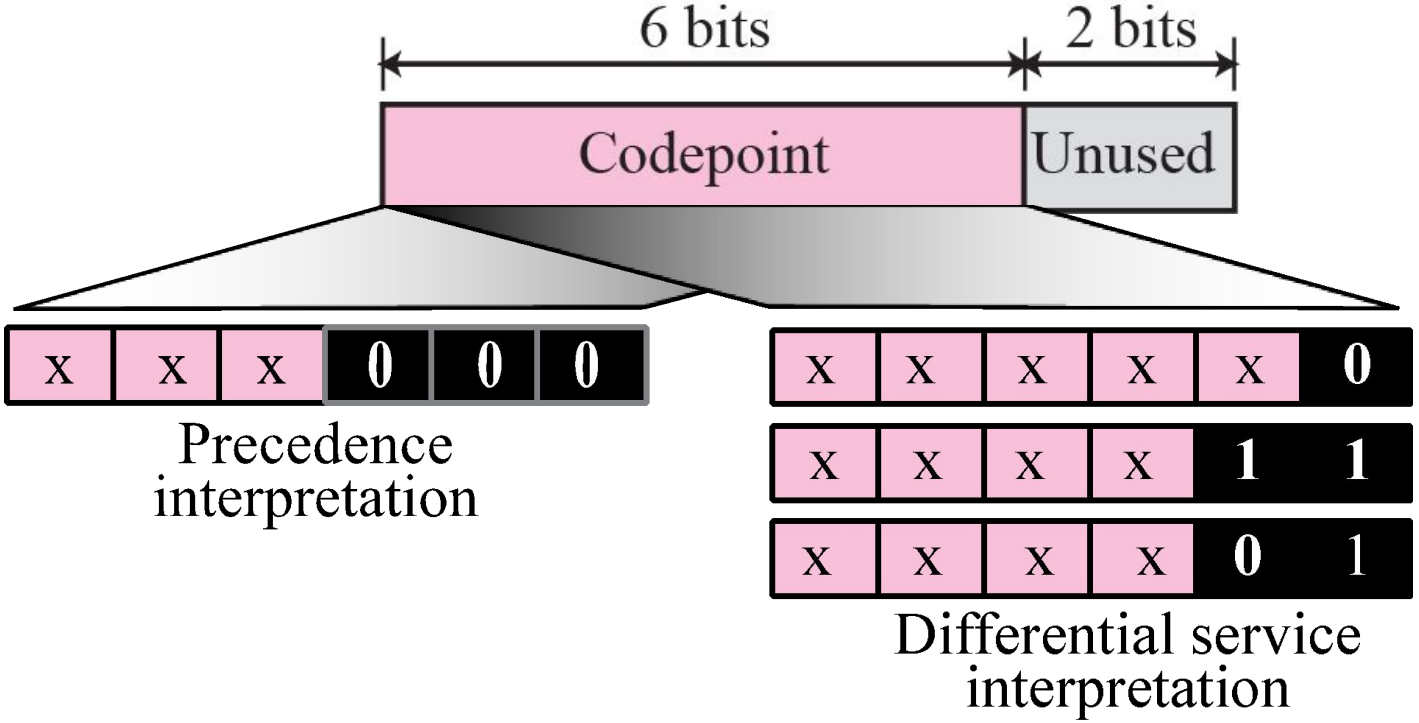


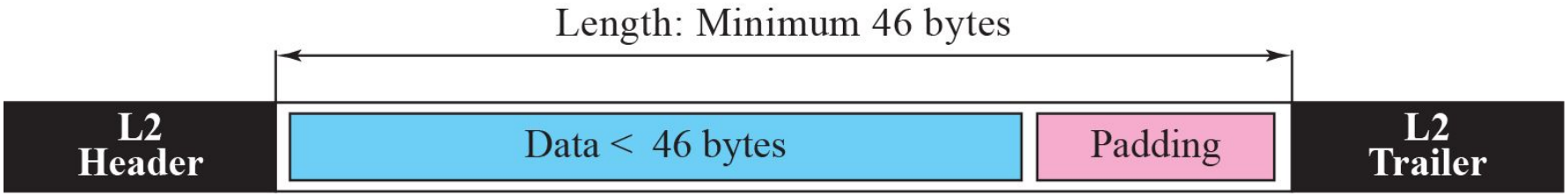
Table 1.1 Values of Codepoints

Category	*Codepoint*	*Assigning Authority*
1	XXXXX0	Internet
2	XXXX11	Local
3	XXXX01	Temporary or experimental

Header Format

Total length: This is a 16-bit field that defines the total length (header plus data) of the IP datagram in bytes.
Length of data = total length − header length

Figure 1.4 Encapsulation of a small datagram in an Ethernet frame



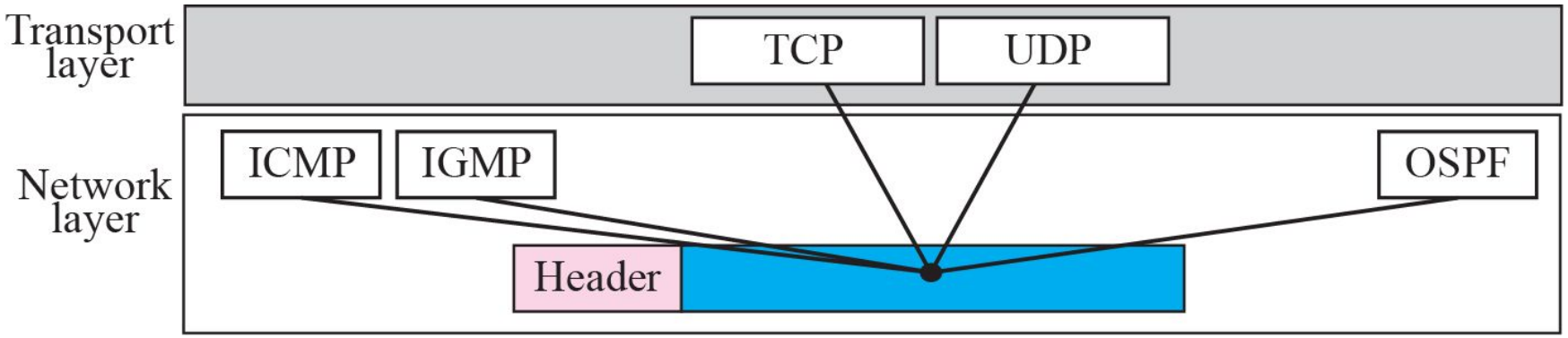
Identification: This field is used in fragmentation.
Flags: This field is used in fragmentation.
Fragmentation offset: This field is used in fragmentation.
Time to live: A datagram has a limited lifetime in its travel through an internet. This field was originally designed to hold a timestamp, which was decremented by each visited router. The datagram was discarded when the value became zero.

Header Format Cont.

Protocol: This 8-bit field defines the higher-level protocol that uses the services of the IP layer. An IP datagram can encapsulate data from several higher level protocols such as TCP, UDP, ICMP, and IGMP.

The IP protocol multiplexes and demultiplexes data from different higher-level protocols, the value of this field helps in the demultiplexing process when the datagram arrives at its final destination (see Figure 1.5).

Figure 1.5 Multiplexing



Header Format Cont.

Some of the value of this field for different higher-level protocols is shown in Table 1.2.

Table 1.2 Protocols

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

Checksum: The checksum concept and its calculation.

Source address: This 32-bit field defines the IP address of the source.

Destination address: This 32-bit field defines the IP address of the destination.

Source and Destination address field must remain unchanged during the time the IP datagram travels from the source host to the destination host.

Example 1.1

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

```
01000010
```

The receiver discards the packet. Why?

Solution

There is an error in this packet. The 4 left-most bits (0100) show the version, which is correct. The next 4 bits (0010) show the wrong header length ($2 \times 4 = 8$). The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.

Example 1.2

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

Solution

The HLEN value is 8, which means the total number of bytes in the header is 8×4 or 32 bytes. The first 20 bytes are the base header, the next 12 bytes are the options.

Example 1.3

In an IP packet, the value of HLEN is 5_{16} and the value of the total length field is 0028_{16}. How many bytes of data are being carried by this packet?

Solution

The HLEN value is 5, which means the total number of bytes in the header is 5 × 4 or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40 − 20).

Example 1.4

An IP packet has arrived with the first few hexadecimal digits as shown below:

```
45000028000100000102 . . .
```

How many hops can this packet travel before being dropped? The data belong to what upper layer protocol?

Solution

To find the time-to-live field, we skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper layer protocol is IGMP (see Table 1.2)

IP Fragmentation

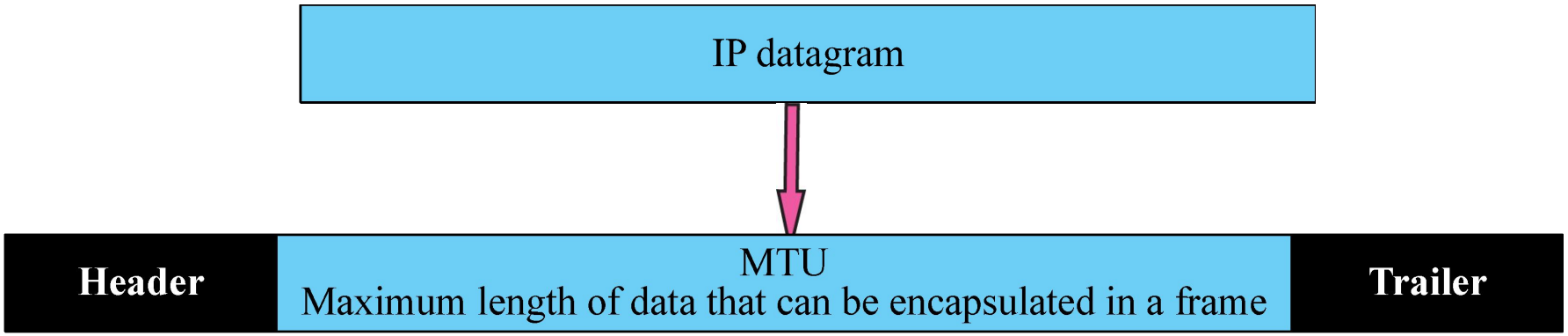
A datagram can travel through different networks.
Each router decapsulates the IP datagram from the frame it receives, processes it, and then encapsulates it in another frame.
The format and size of the received frame depend on the protocol used by the physical network through which the frame has just traveled.
The format and size of the sent frame depend on the protocol used by the physical network through which the frame is going to travel.

Maximum Transfer Unit (MTU)

Each data link layer protocol has its own frame format in most protocols. One of the fields defined in the format is the maximum size of the data field.
When a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network.

IP Fragmentation Cont.

Figure 7.6 ***MTU***



When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some changed.

A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU.

In other words, a datagram can be fragmented several times before it reaches the final destination.

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

IP Fragmentation Cont.

Fields Related to Fragmentation

The fields that are related to fragmentation and reassembly of an IP datagram are the identification, flags, and fragmentation offset fields.

Identification: This 16-bit field identifies a datagram originating from the source host. The combination of the identification and source IP address must uniquely define a datagram as it leaves the source host.

Flags: This is a three-bit field.

The first bit is reserved (not used).

The second bit is called the do not fragment bit.

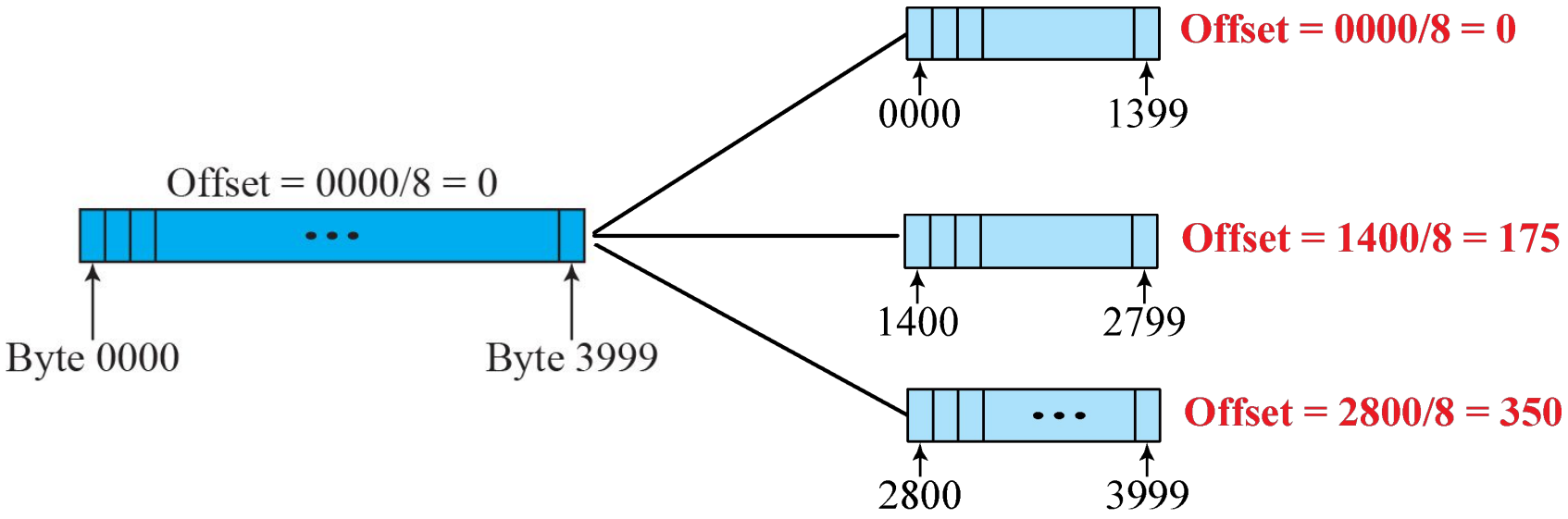
The third bit is called the more fragment bit.

Figure 1.7 ***Flags field***

IP Fragmentation Cont.

Fragmentation offset: This 13-bit field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes. Figure 1.8 shows a datagram with a data size of 4000 bytes fragmented into three fragments.

Figure 1.8 ***Fragmentation example***



IP Fragmentation Cont.

Notice the value of the identification field is the same in all fragments. Notice the value of the flags field with the more bit set for all fragments except the last.

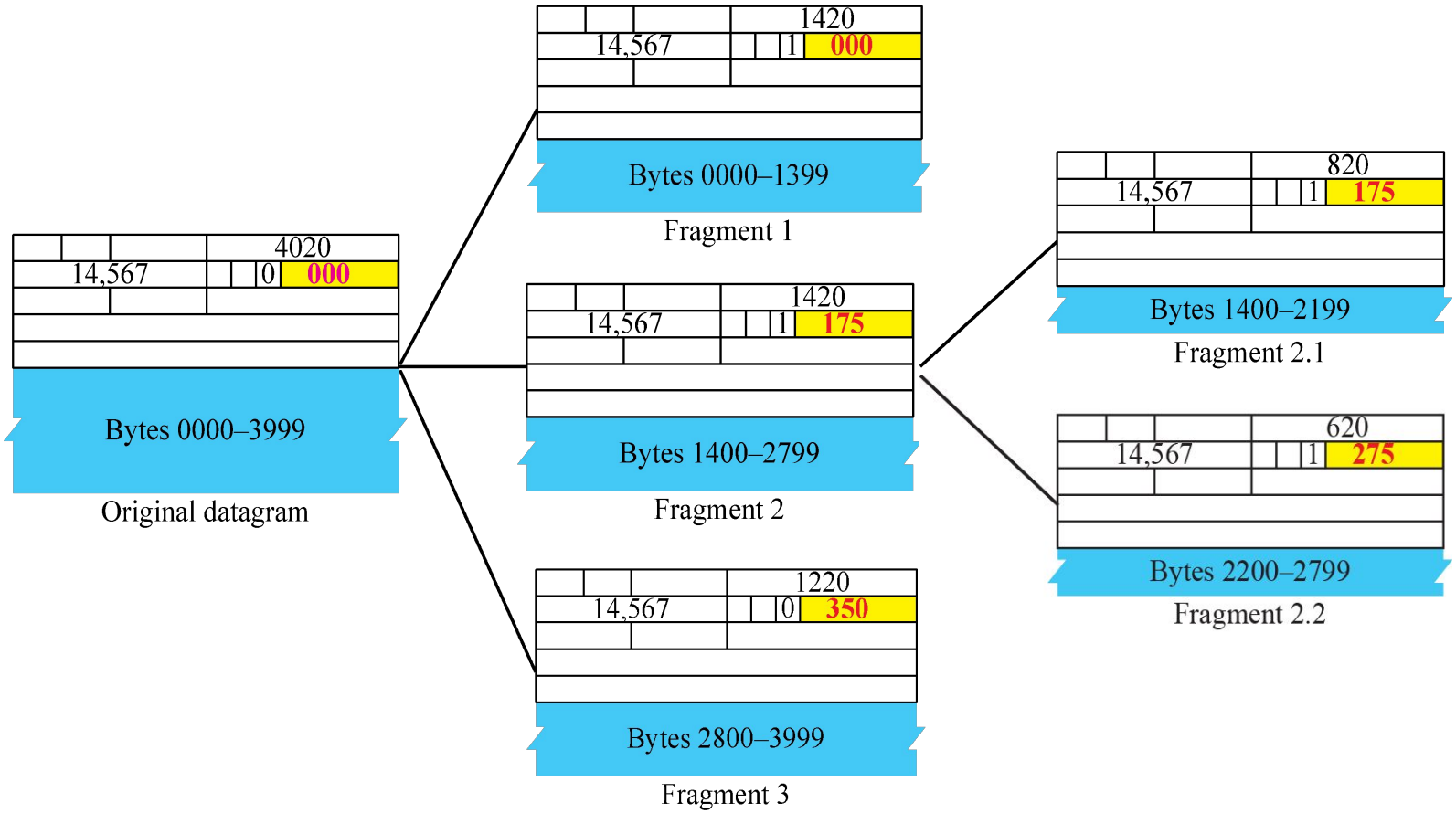


Figure 1.9 ***Detailed fragmentation example***

Example 1.5

A packet has arrived with an M bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A nonfragmented packet is considered the last fragment.

Example 1.6

A packet has arrived with an M bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

If the M bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don't know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). See also the next example.

Example 1.7

A packet has arrived with an M bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?

Solution

Because the M bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 1.8

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

Solution

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

Example 1.9

A packet has arrived in which the offset value is 100, the value of HLEN is 5 and the value of the total length field is 100. What is the number of the first byte and the last byte?

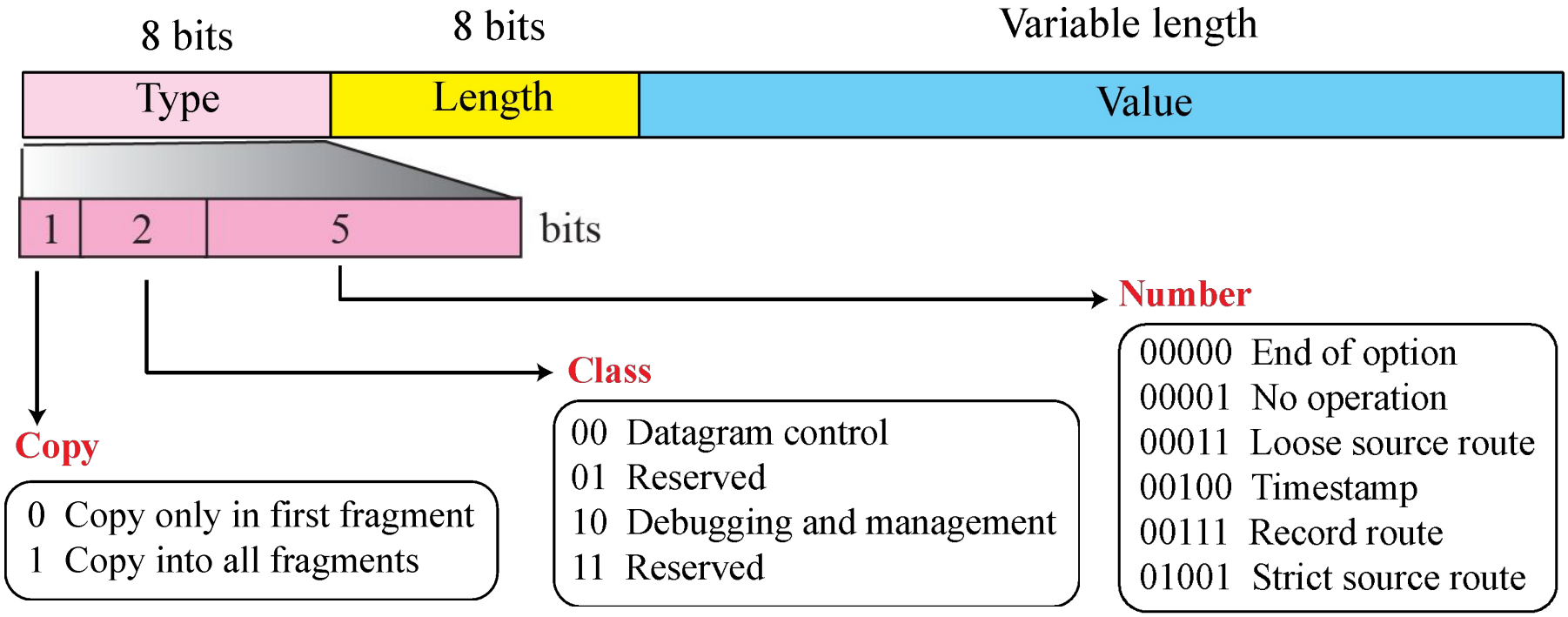
Solution

The first byte number is 100 × 8 = 800. The total length is 100 bytes and the header length is 20 bytes (5 × 4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

IP Fragmentation 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.
This means that all implementations must be able to handle options if they are present in the header.

Figure 1.10 Option format



It is composed of a 1-byte type field, a 1-byte length field, and a variable-sized value field.

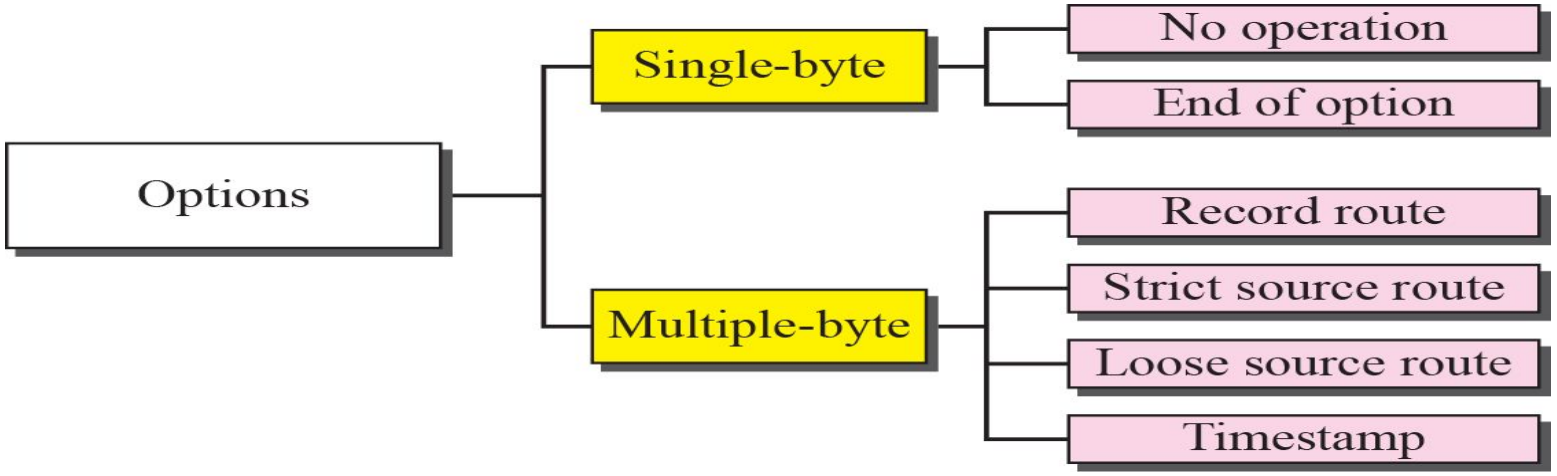
The three fields are often referred to as type-length-value or TLV.

IP Fragmentation Options Cont.

Type: The type field is 8 bits long and contains three subfields: copy, class, and number.
Copy: This 1-bit subfield controls the presence of the option in fragmentation.
value is 0, the option must be copied only to the first fragment.
value is 1, the option must be copied to all fragments.
Class: This 2-bit subfield defines the general purpose of the option.
Value is 00, option is used for datagram control.
Value is10, option is used for debugging and management.
The other two possible values (01 and 11) have not yet been defined.
Number: This 5-bit subfield defines the type of option.
Although 5 bits can define up to 32 different types, currently only 6 types are in use.
Length: 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 data that specific options require.
Like the length field, this field is also not present in all option types.

Option Types

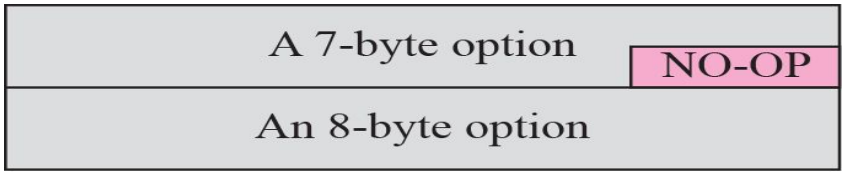
Figure 1.11 Categories of options



No-Operation: It is a 1-byte option used as a filler between options. For example, it can be used to align the next option on a 16-bit or 32-bit boundary

a. No operation option

b. Used to align beginning of an option



c. Used to align the next option

Figure 1.12 ***No operation option***

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 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.

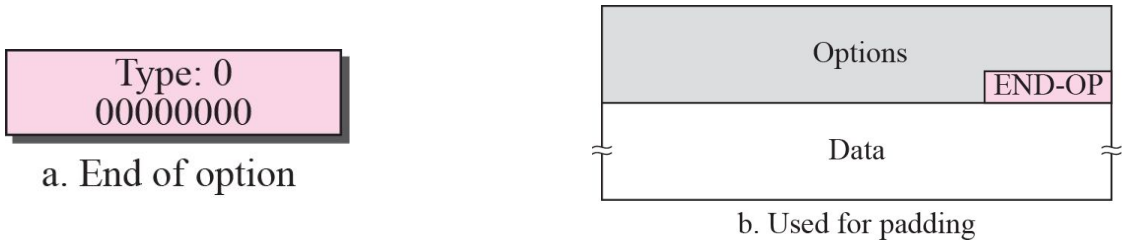


Figure 1.13 ***Endo-of-option option***

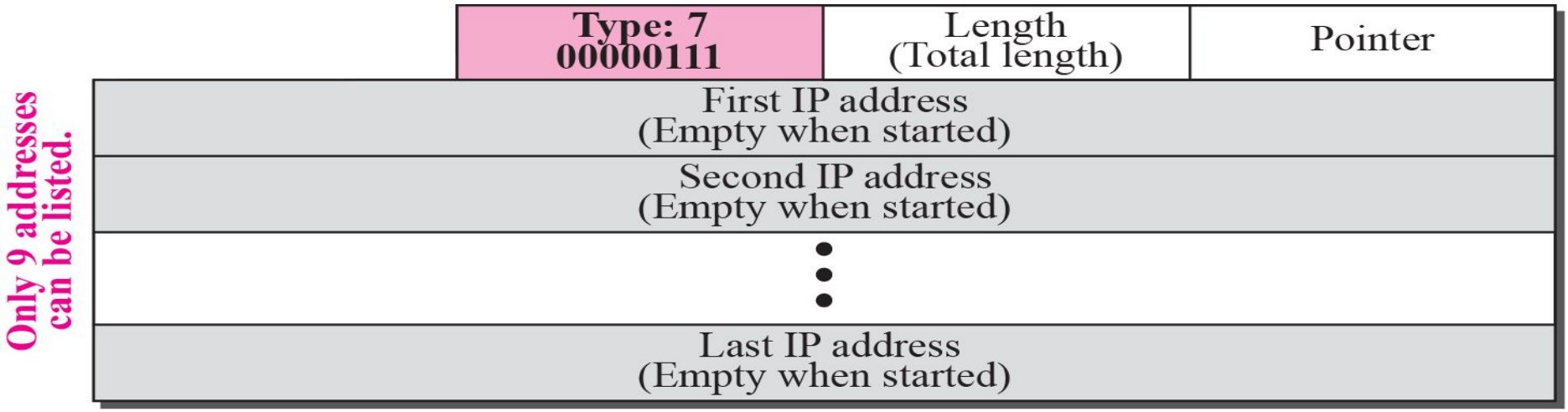
Record-Route Option:

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, which must include 20 bytes for the base header.

This implies that only 40 bytes are left over for the option part.

The source creates placeholder fields in the option to be filled by the visited routers.

Figure 1.14 ***Record-route option***



When the datagram is traveling, each router that processes the datagram compares the value of the pointer with the value of the length.

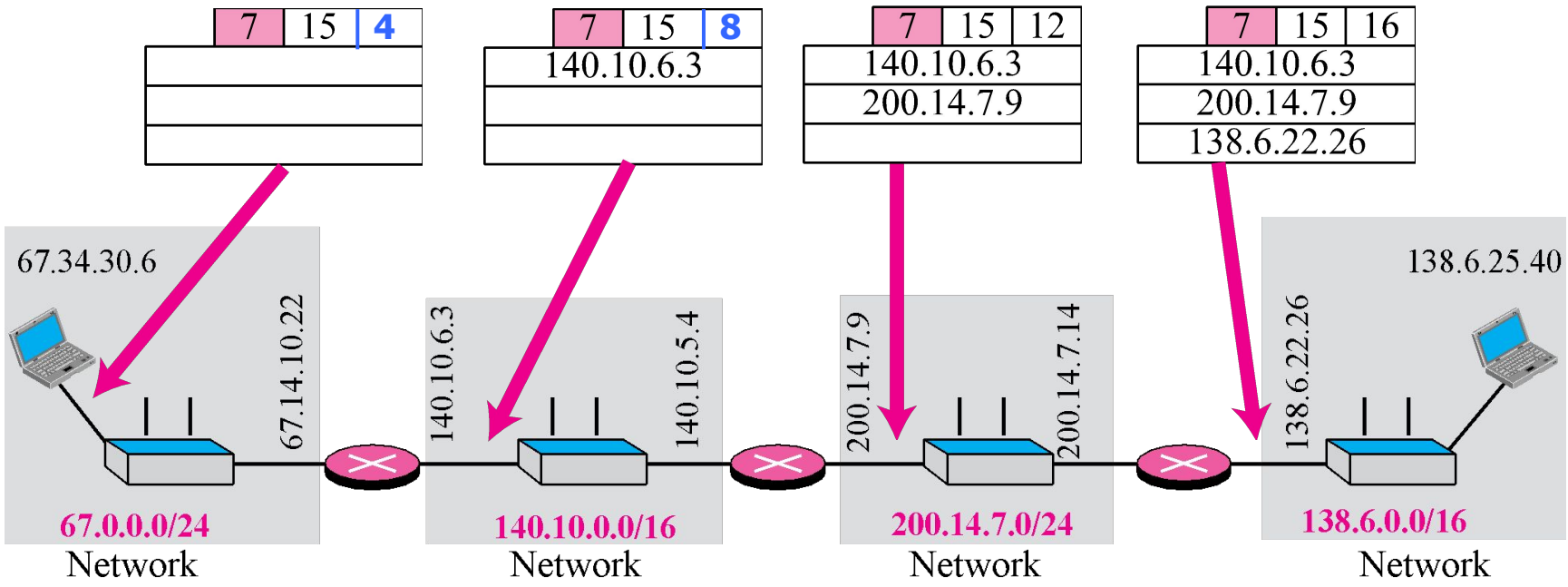
If the value of the pointer is greater than the value of the length, the option is full and no changes are made.

The pointer field is an offset integer field containing the byte number of the first empty entry. In other words, it points to the first available entry.

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

The router then increments the value of the pointer by 4.

Figure 1.15 *Record-route concept*



Strict-source-route option:

It 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.

Figure 1.16 *Strict-source-route option*

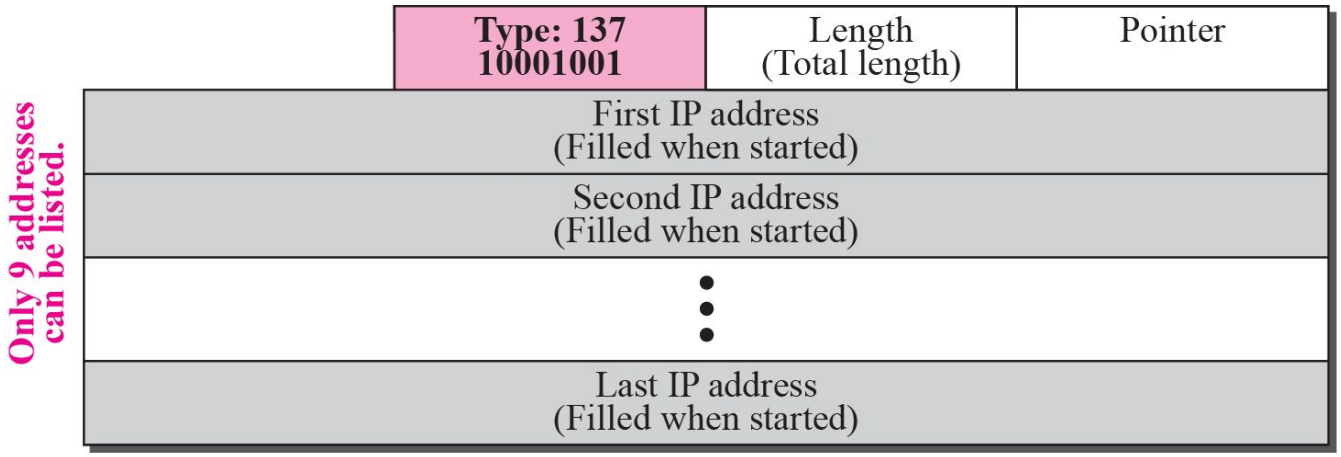


Figure 1.17 *Strict-source-route option*

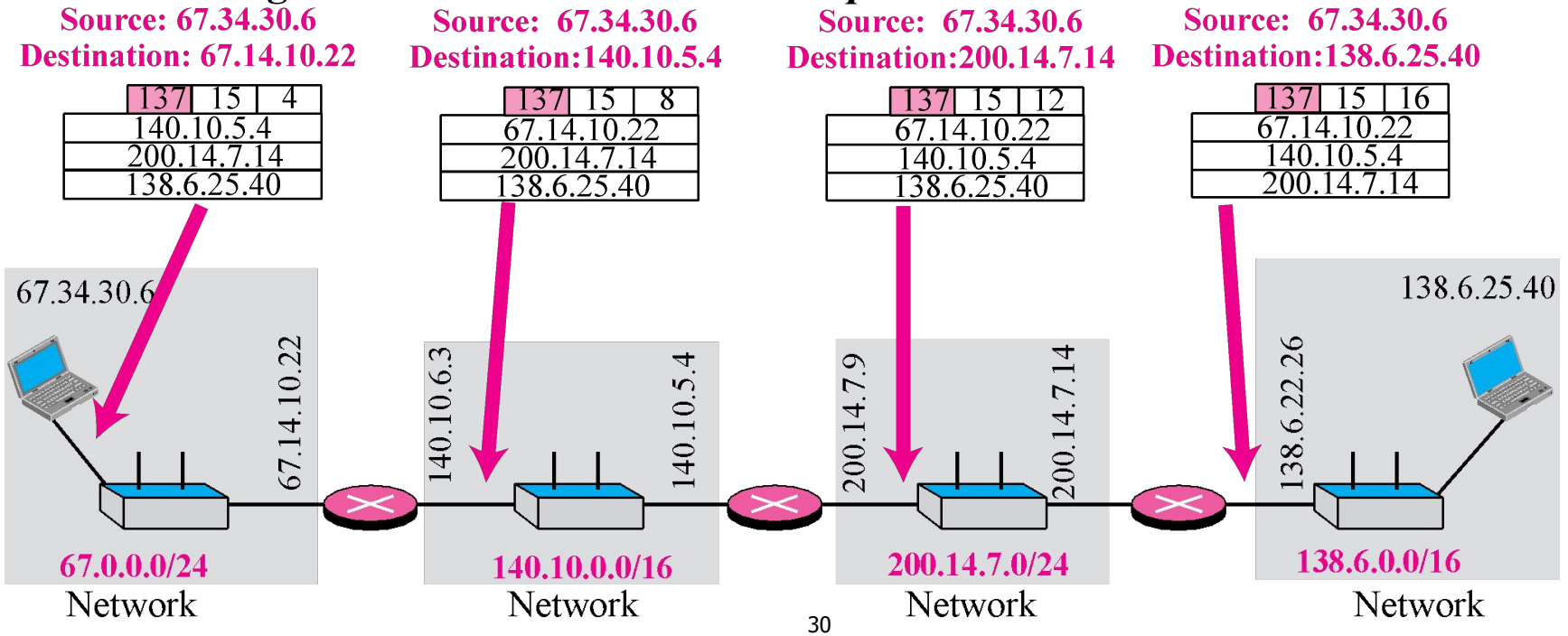
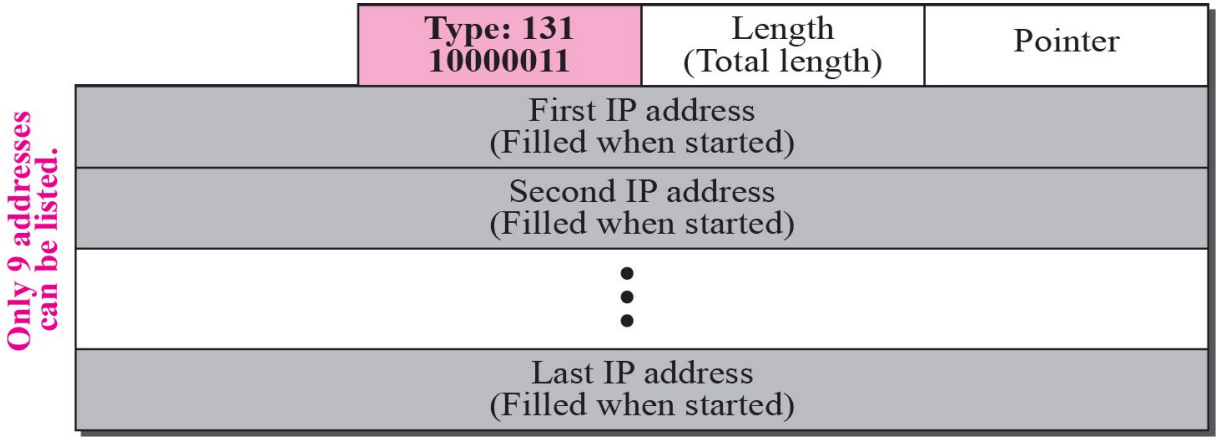


Figure 1.18 *Loose-source-route option*



A loose-source-route option is similar to the strict source route, but it is more relaxed.

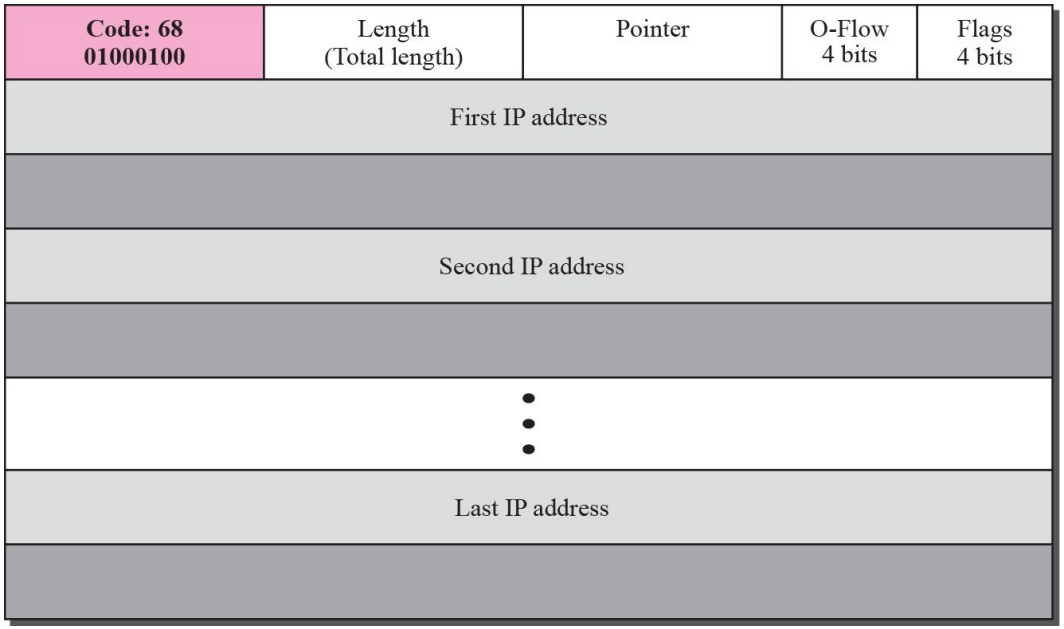
Each router in the list must be visited, but the datagram can visit other routers as well.

Timestamp: used to record the time of datagram processing by a router.

The time is expressed in milliseconds from midnight, Universal Time.

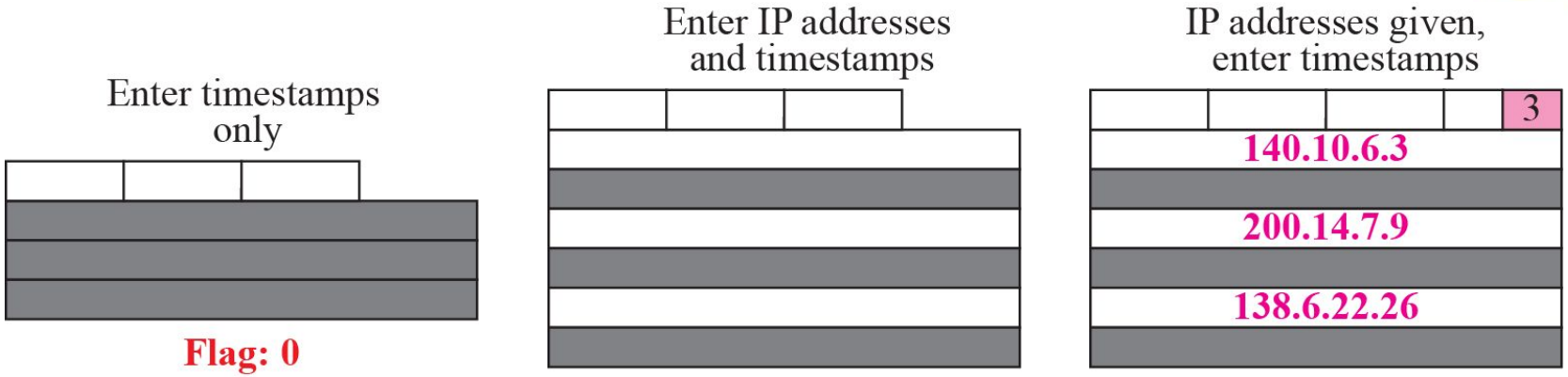
However, no privileged users of the Internet are not usually aware of the physical topology of the Internet.

Figure 1.19 Time-stamp *option*



The flags field specifies the visited router responsibilities. If the flag value is 0, each router adds only the timestamp in the provided field. If the flag value is 1, each router must add its outgoing IP address and the timestamp. If the value is 3, the IP addresses are given, and each router must check the given IP address with its own incoming IP address.

Figure 1.20 *Use of flags in timestamp*



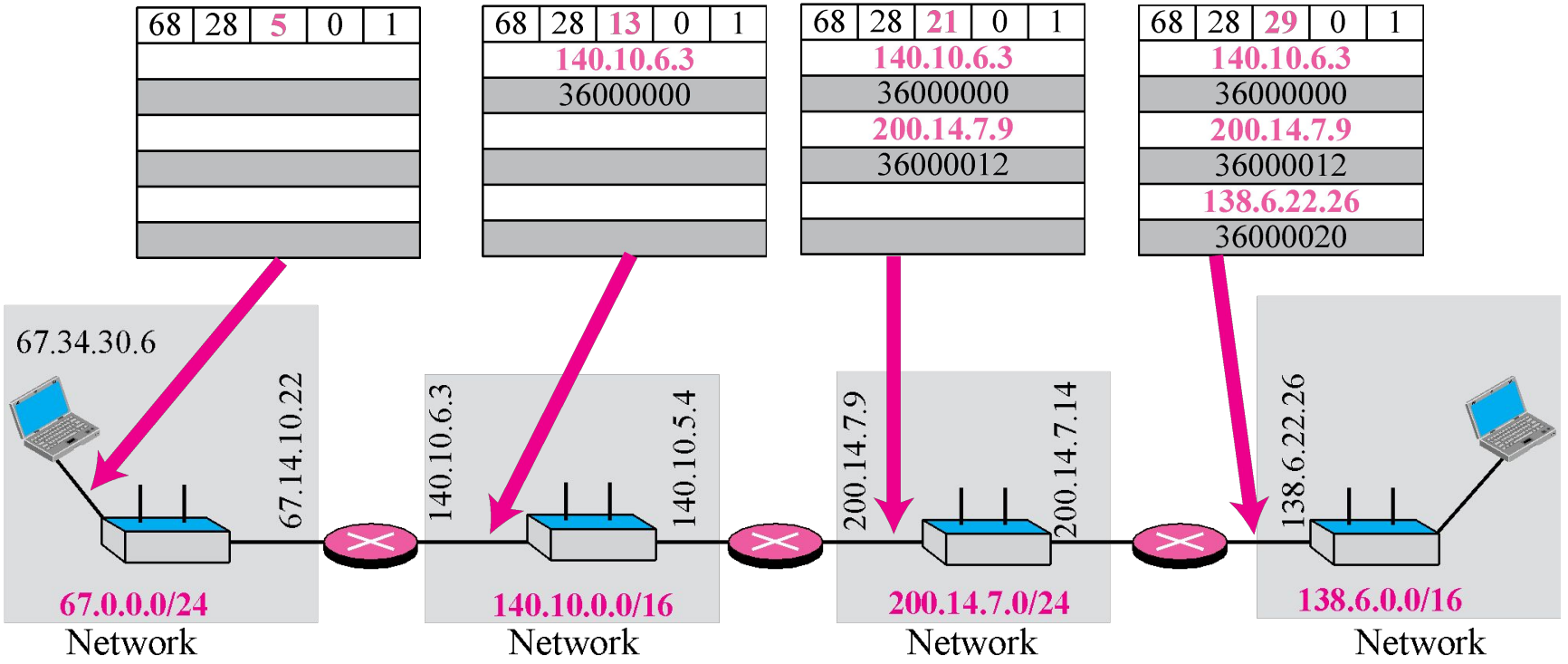


Figure 1.21 *Timestamp concept*

Example 1.10

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 1.11

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 1.12

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 1.13

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 1.14

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 1.15

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 1.16

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
```

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.

Checksum Calculation at the Sender

At the sender, the packet header is divided into n-bit sections (n is usually 16).

These sections are added together using one's complement arithmetic, resulting in a sum that is also n bits long.

The sum is then complemented (all 0s changed to 1s and all 1s to 0s) to produce the checksum.

Checksum Calculation at the Receiver

The receiver divides the received packet into k sections and adds all sections.

It then complements the result.

If the final result is 0, the packet is accepted; otherwise, it is rejected.

Figure 1.22 shows graphically what happens at the sender and the receiver.

Figure 1.22 *Checksum concept*

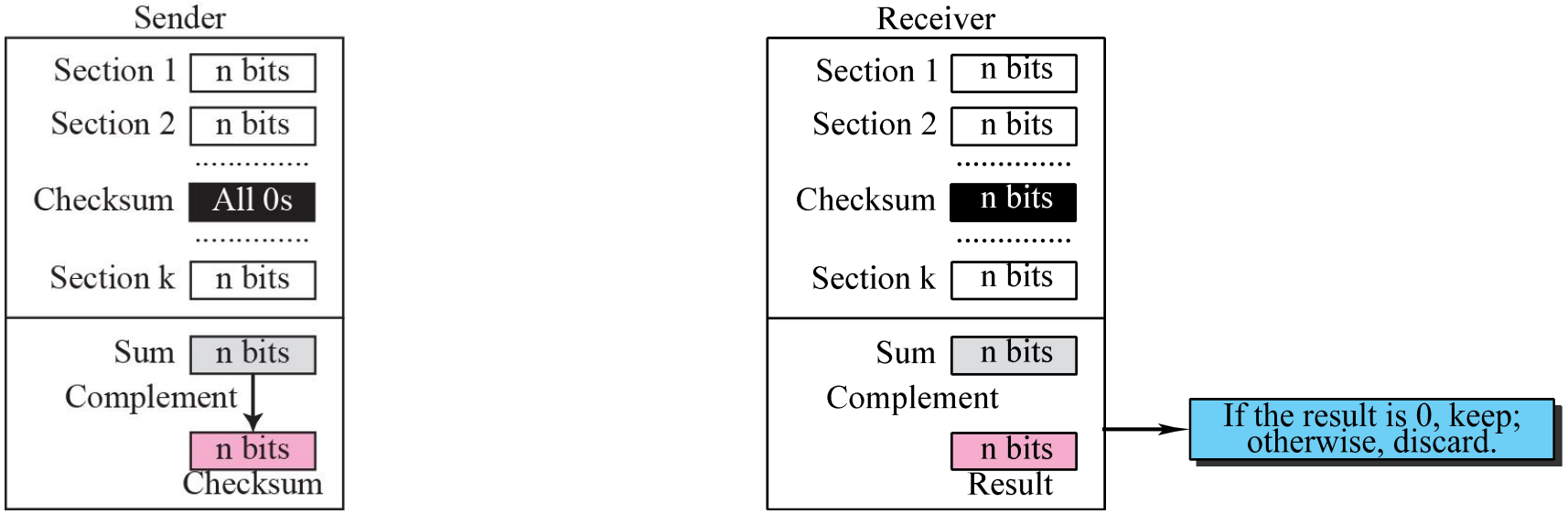
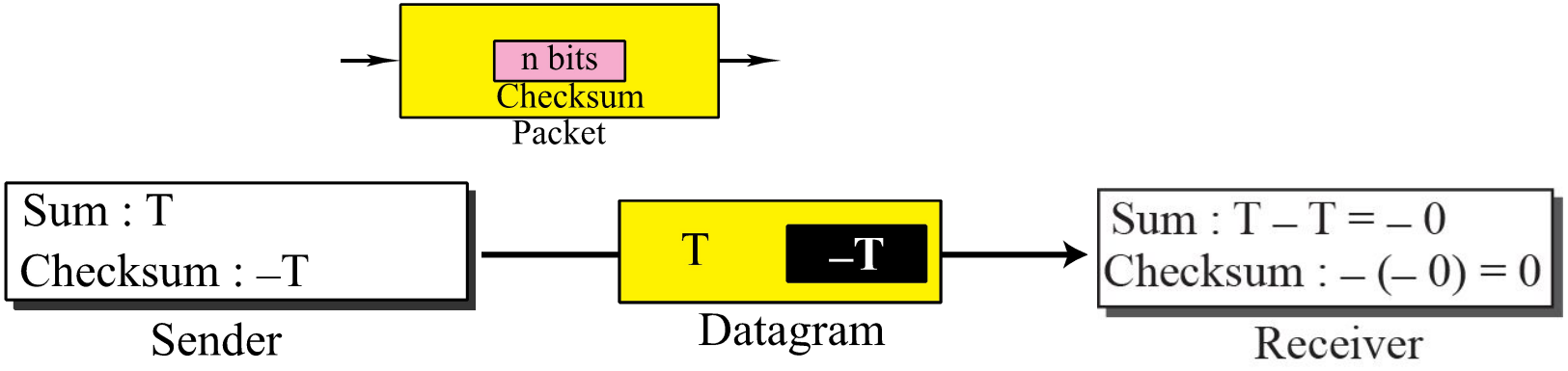


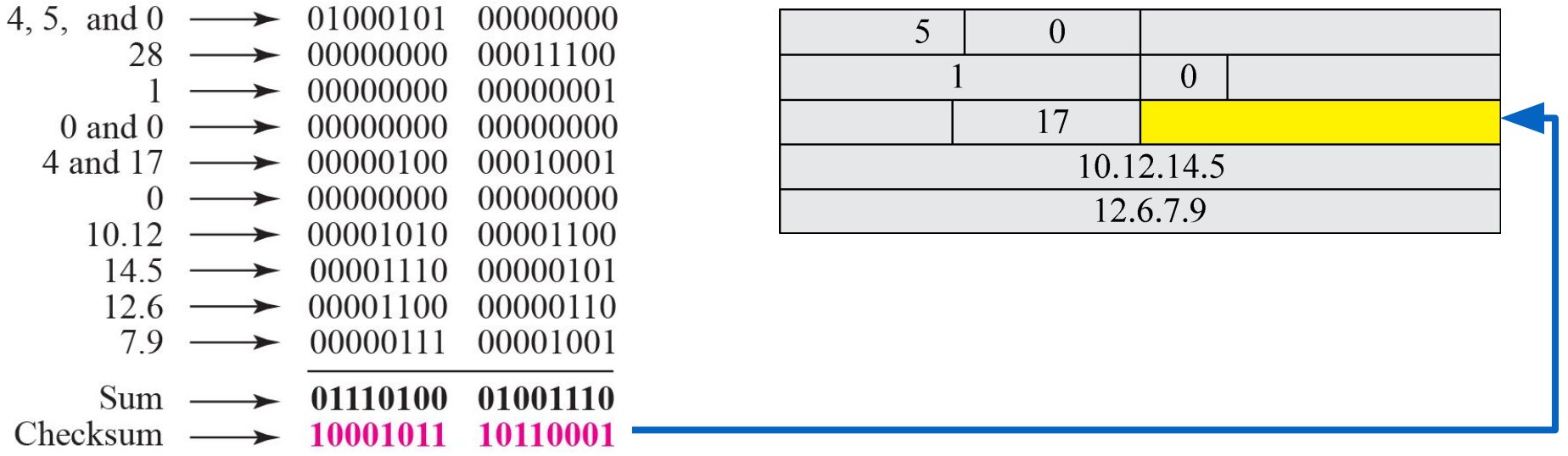
Figure 1.23 *Checksum in one's complement arithmetic*



Example 1.17

Figure 1.24 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.

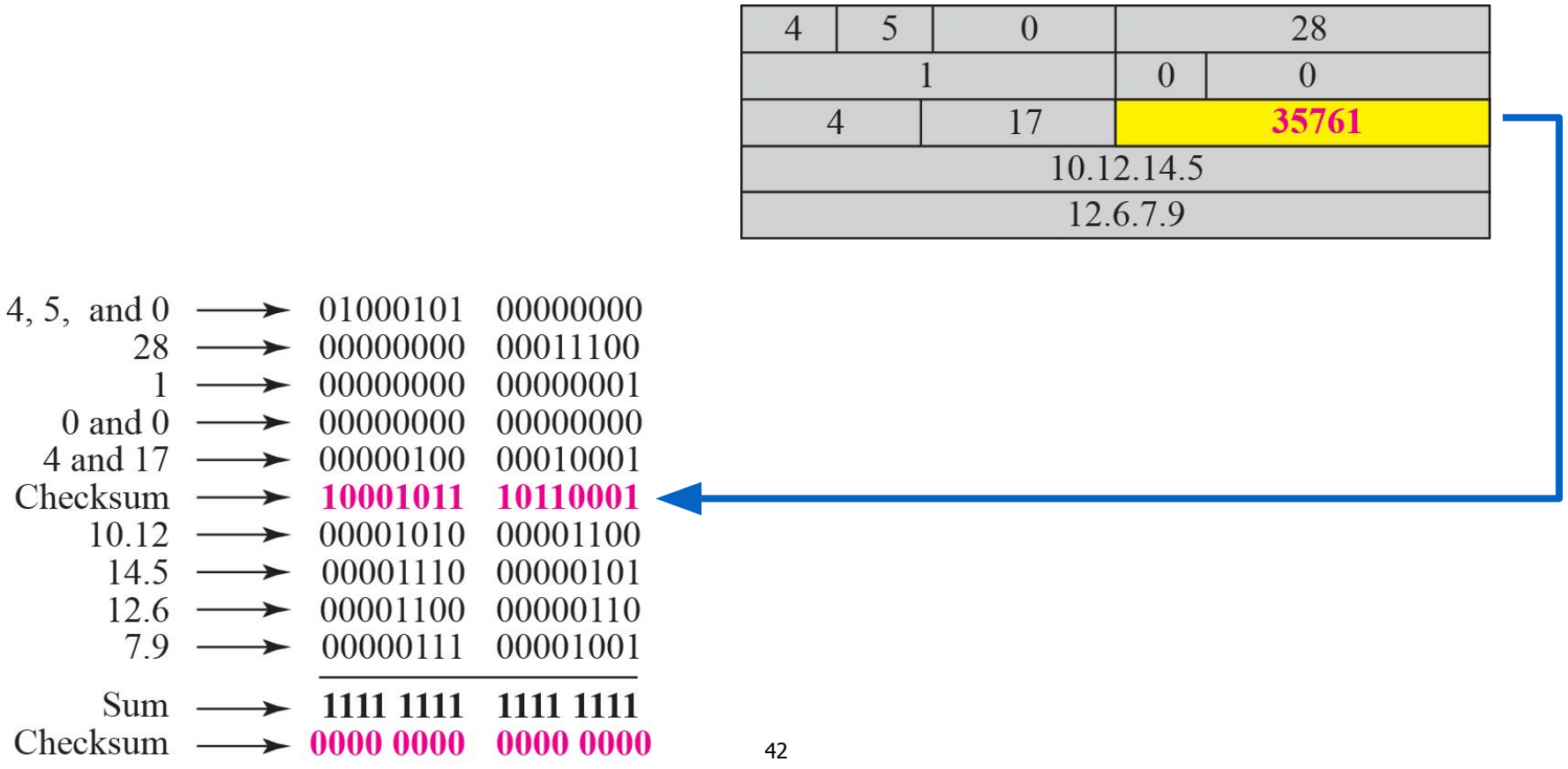
Figure 1.24 ***Example of checksum calculation at the sender***



Example 1.18

Figure 1.25 shows the checking of checksum calculation at the receiver site (or intermediate router) assuming that no errors occurred in the header. The header is divided into 16-bit sections. All the sections are added and the sum is complemented. Since the result is 16 0s, the packet is accepted.

Figure 1.25 ***Example of checksum calculation at the receiver***



Address Resolution Protocol

OBJECTIVES:

- To make a distinction between logical address (IP address) and physical address (MAC address).
- To describe how the mapping of a logical address to a physical address can be static or dynamic.
- To show how the address resolution protocol (ARP) is used to dynamically map a logical address to a physical address.
- To show that the proxy ARP can be used to create a subnetting effect.
- To show that an ARP software package can be made of five components.

ADDRESS MAPPING

The hosts and routers are recognized at the network level by their logical addresses.

A logical address is an internetwork address and unique universally.

It is called a logical address because it is usually implemented in software.

Every protocol that deals with interconnecting networks requires logical addresses.

The logical addresses in the TCP/IP protocol suite are called IP addresses and are 32 bits long.

A physical address is a local address, because it is usually (but not always) implemented in hardware.

Examples of physical addresses are 48-bit MAC addresses in the Ethernet protocol, which are imprinted on the NIC installed in the host or router.

ADDRESS MAPPING

Static Mapping

Static mapping means creating a table that associates a logical address with a physical address.

This table is stored in each machine on the network.

Each machine that knows, for example, the IP address of another machine but not its physical address can look it up in the table.

Dynamic Mapping

Each time a machine knows the logical address of another machine, it can use a protocol to find the physical address.

Two protocols have been designed to perform dynamic mapping: Address Resolution Protocol (ARP) and Reverse Address Resolution Protocol (RARP).

ARP maps a logical address to a physical address;

RARP maps a physical address to a logical address.

Address Resolution Protocol

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver.
But the IP datagram must be encapsulated in a frame to be able to pass through the physical network.
This means that the sender needs the physical address of the receiver.
A mapping corresponds a logical address to a physical address.
ARP accepts a logical address from the IP protocol, maps the address to the corresponding physical address and pass it to the data link layer.

Anytime a host or a router has an IP datagram to send to another host or router, it has the logical (IP) address of the receiver.
But the IP datagram must be encapsulated in a frame to be able to pass through the physical network.
This means that the sender needs the physical address of the receiver.
A mapping corresponds a logical address to a physical address.

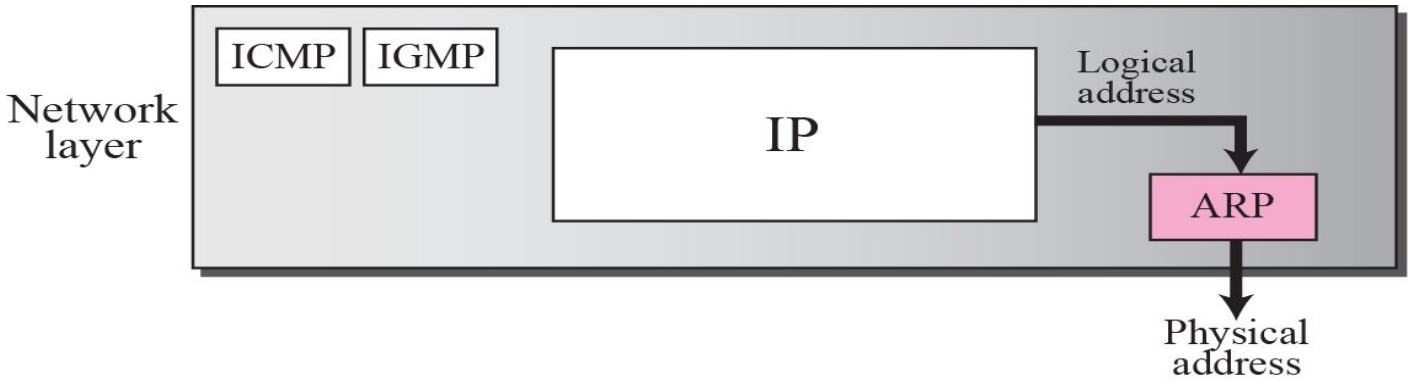
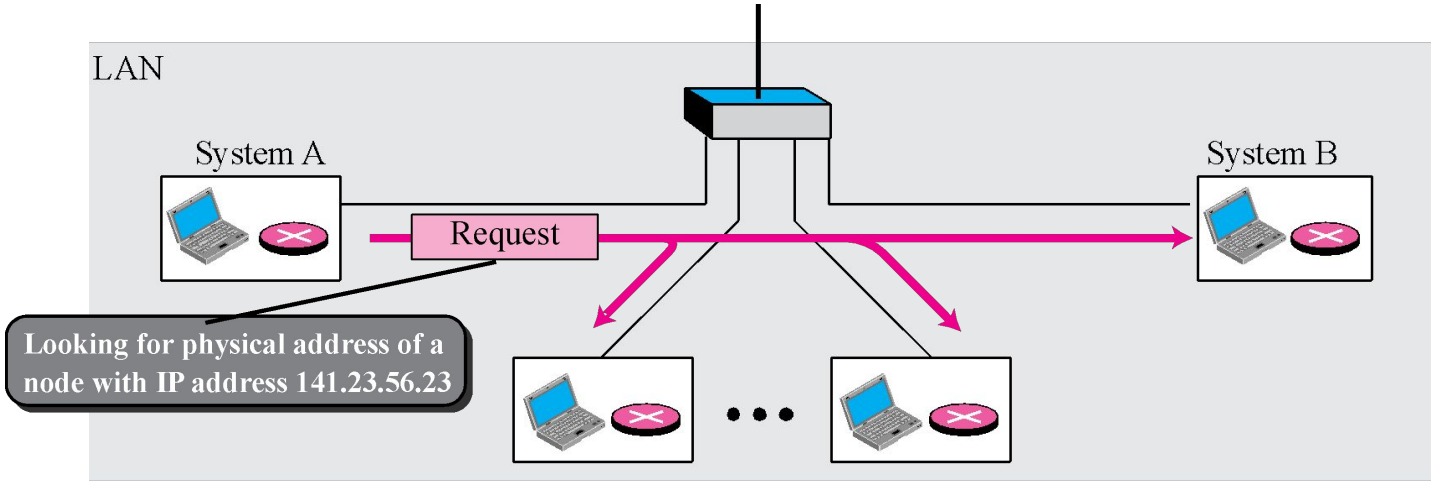
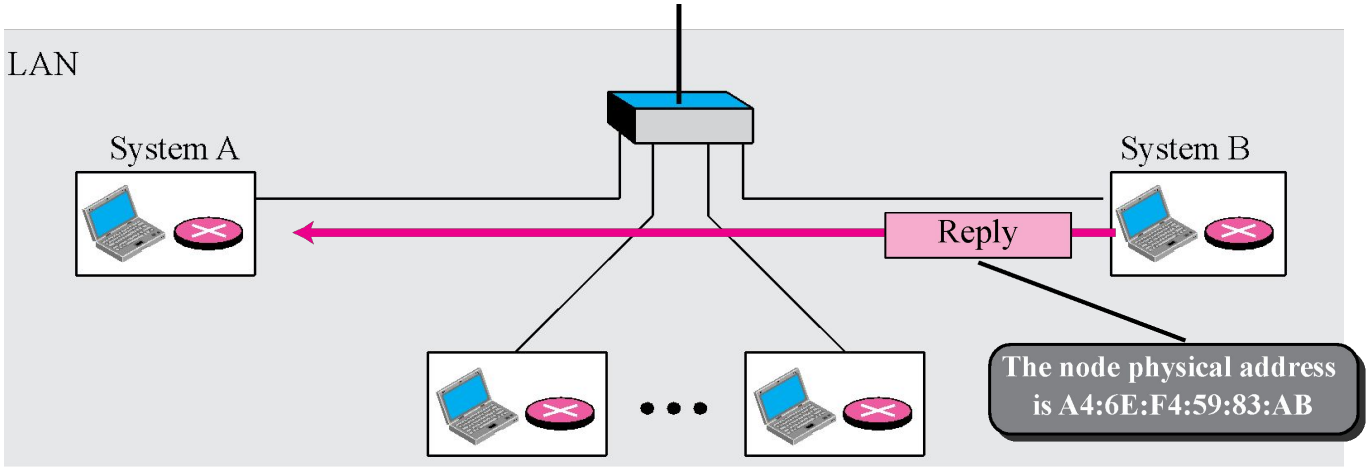


Figure 1.25 ***Position of ARP in TCP/IP protocol suite***

Figure 1.26 *ARP operation*

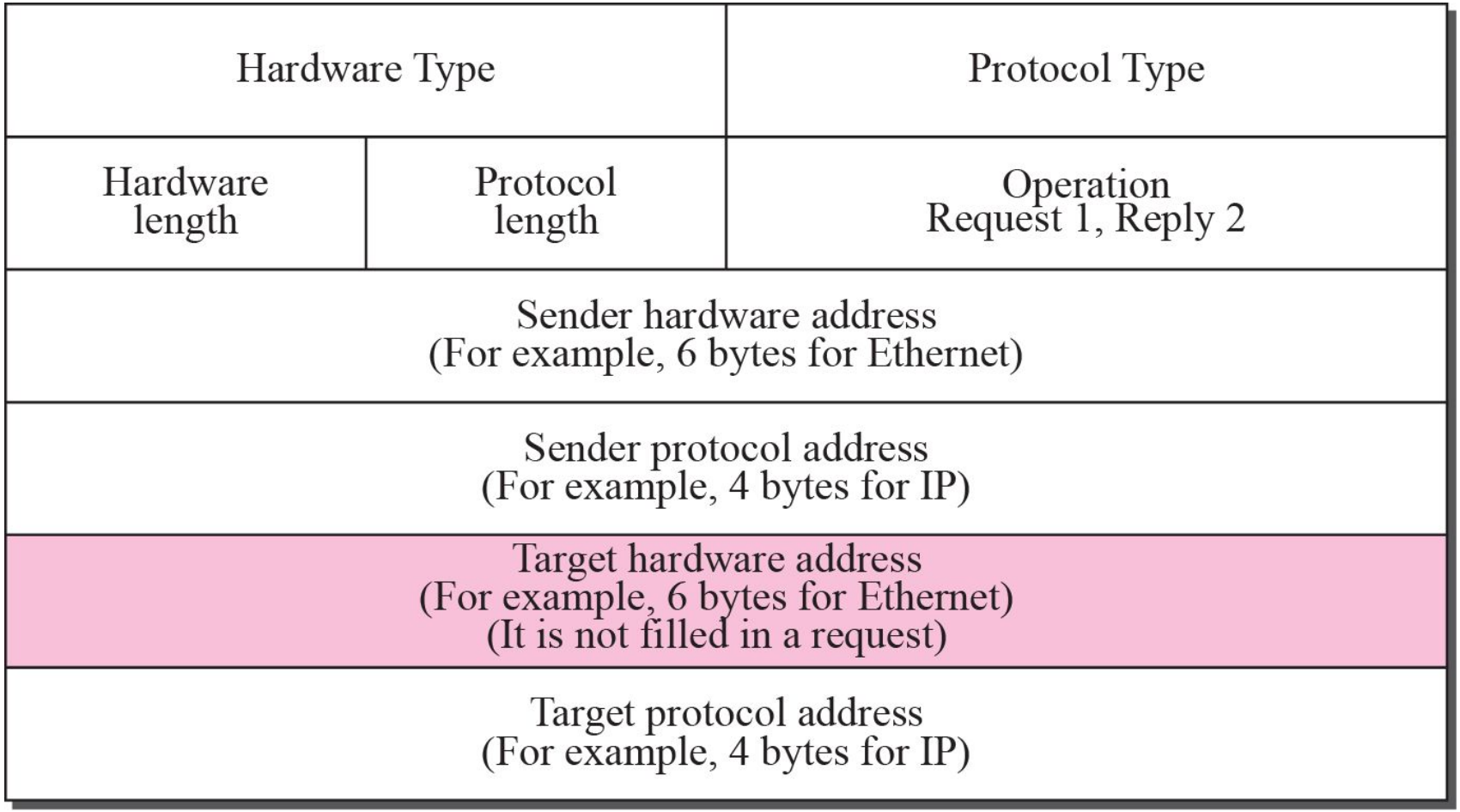


a. ARP request is **broadcast**



b. ARP reply is unicast

Figure 1.27 ***ARP packet***



ARP Packet Format

The fields are as follows:

Hardware type: This is a 16-bit field defining the type of the network on which ARP is running.

Each LAN has been assigned an integer based on its type. For example, Ethernet is given the type 1. ARP can be used on any physical network.

Protocol type This is a 16-bit field defining the protocol.

For example, the value of this field for the IPv4 protocol is 080016.

ARP can be used with any higher-level protocol.

Hardware length: This is an 8-bit field defining the length of the physical address in bytes.

For example, for Ethernet the value is 6.

Protocol length: This is an 8-bit field defining the length of the logical address in bytes.

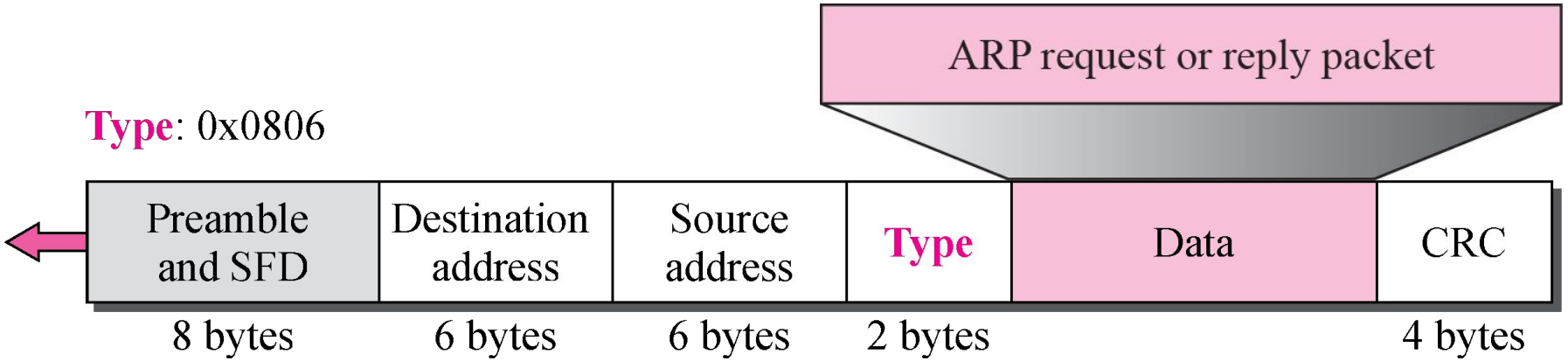
For example, for the IPv4 protocol the value is 4.

Operation: This is a 16-bit field defining the type of packet. Two packet types are defined: ARP request (1), ARP reply (2).

ARP Packet Format Cont.

Sender hardware address: This is a variable-length field defining the physical address of the sender.
For example, for Ethernet this field is 6 bytes long.
Sender protocol address: This is a variable-length field defining the logical (for example, IP) address of the sender.
For the IP protocol, this field is 4 bytes long.
Target hardware address: This is a variable-length field defining the physical
address of the target.
For example, for Ethernet this field is 6 bytes long.
For an ARP request message, this field is all 0s because the sender does not know the physical address of the target.
Target protocol address: This is a variable-length field defining the logical (for
example, IP) address of the target.
For the IPv4 protocol, this field is 4 bytes long.

Figure 1.28 ***Encapsulation of ARP packet***



Encapsulation

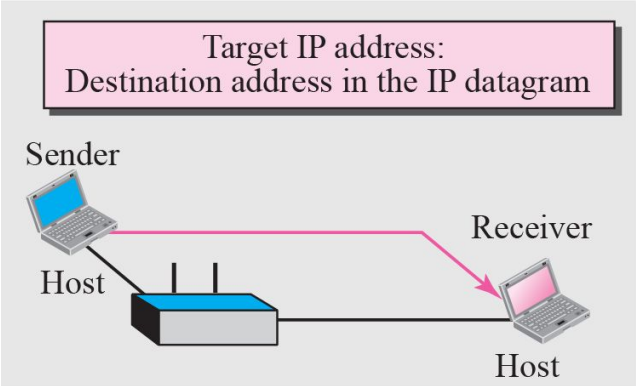
An ARP packet is encapsulated directly into a data link frame.
For example, an ARP packet is encapsulated in an Ethernet frame.
Note that the type field indicates that the data carried by the frame is an ARP packet.

Operation

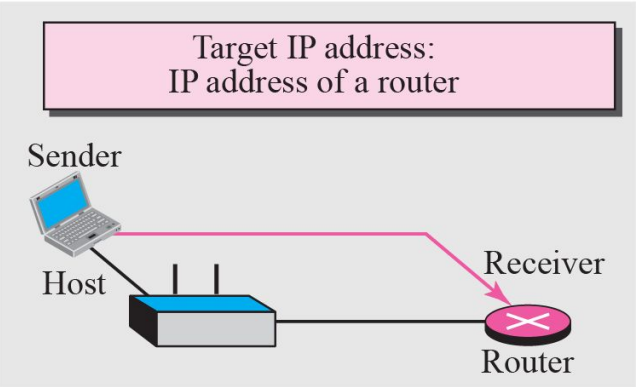
Let us see how ARP functions on a typical internet.
First we discuss the four cases in which a host or router needs to use ARP.

Figure 1.29 *Four cases using ARP*

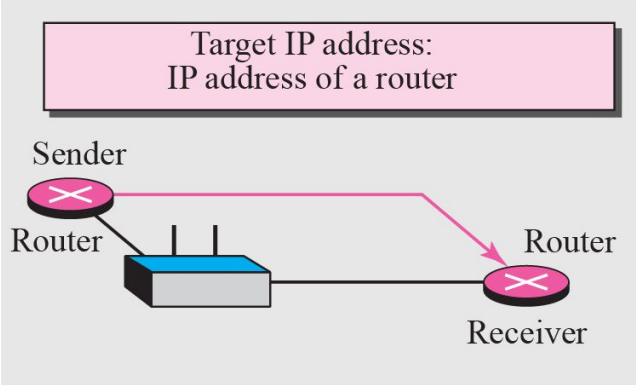
Case 1: A host has a packet to send to a host on the same network.



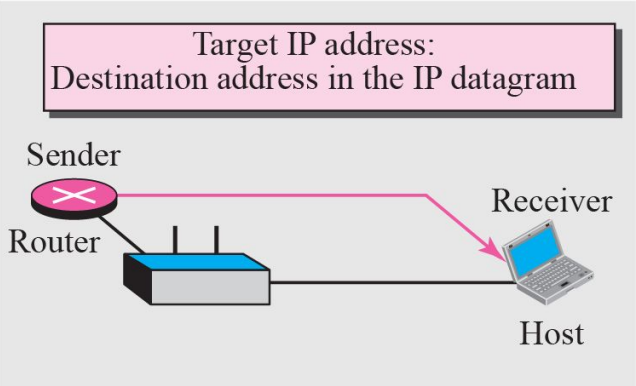
Case 2: A host has a packet to send to a host on another network.



Case 3: A router has a packet to send to a host on another network.



Case 4: A router has a packet to send to a host on the same network.



Example 8.1

A host with IP address 130.23.43.20 and physical address B2:34:55:10:22:10 has a packet to send to another host with IP address 130.23.43.25 and physical address A4:6E:F4:59:83:AB. The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

Solution

Figure 1.30 shows the ARP request and reply packets. Note that the ARP data field in this case is 28 bytes, and that the individual addresses do not fit in the 4-byte boundary. That is why we do not show the regular 4-byte boundaries for these addresses. Also note that the IP addresses are shown in hexadecimal.

Figure 1.30 ***Example 1.26***

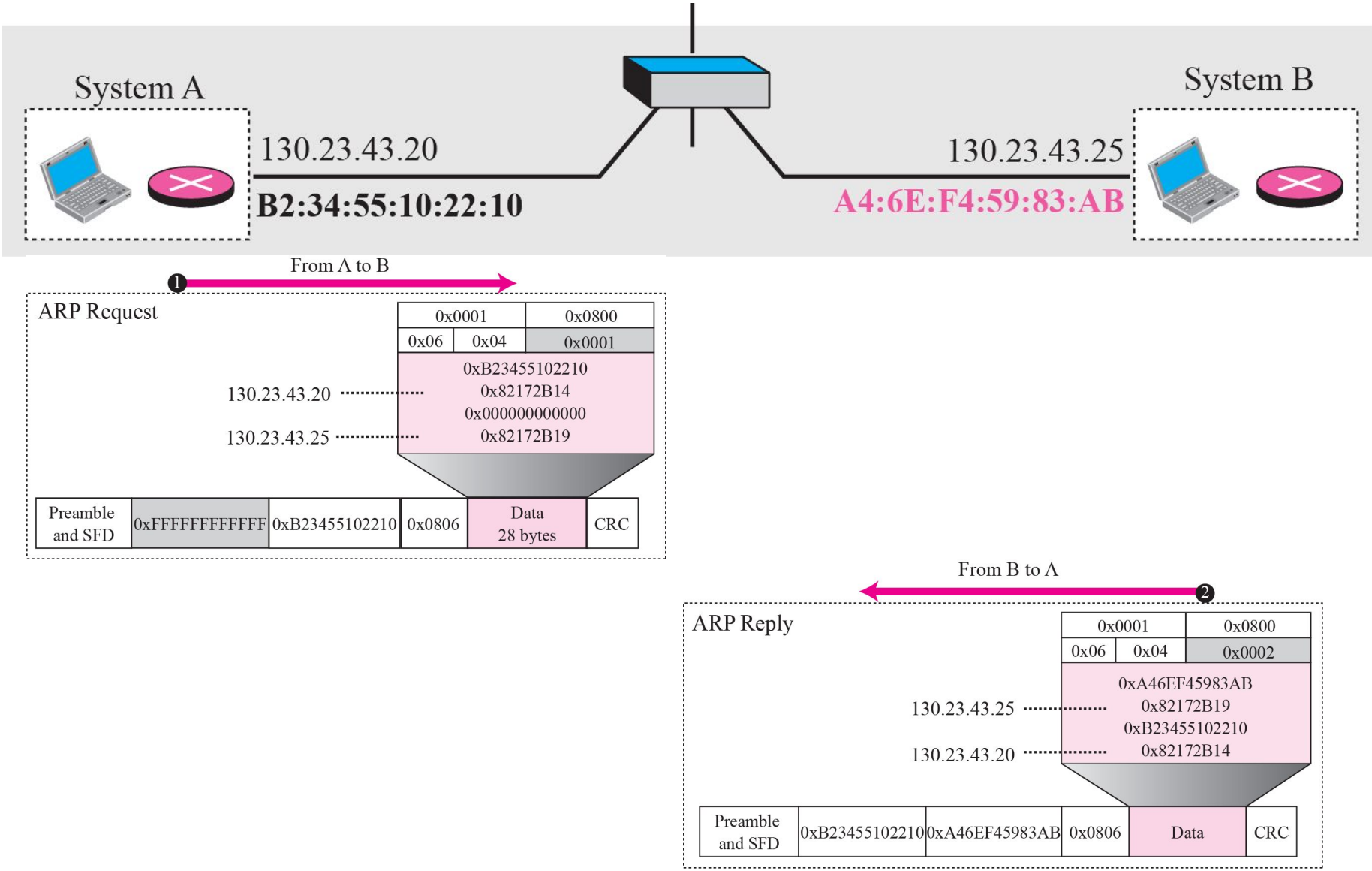
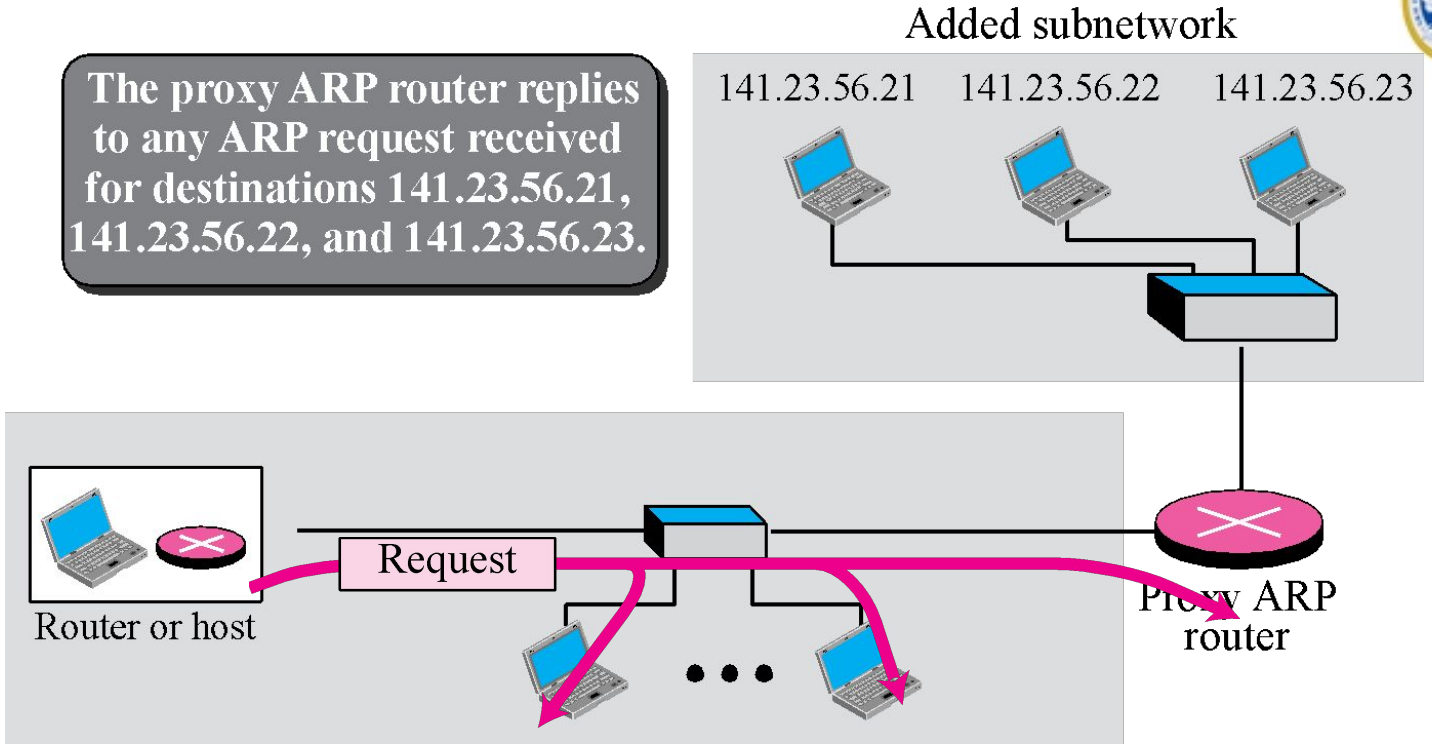


Figure 1.31*Proxy ARP*

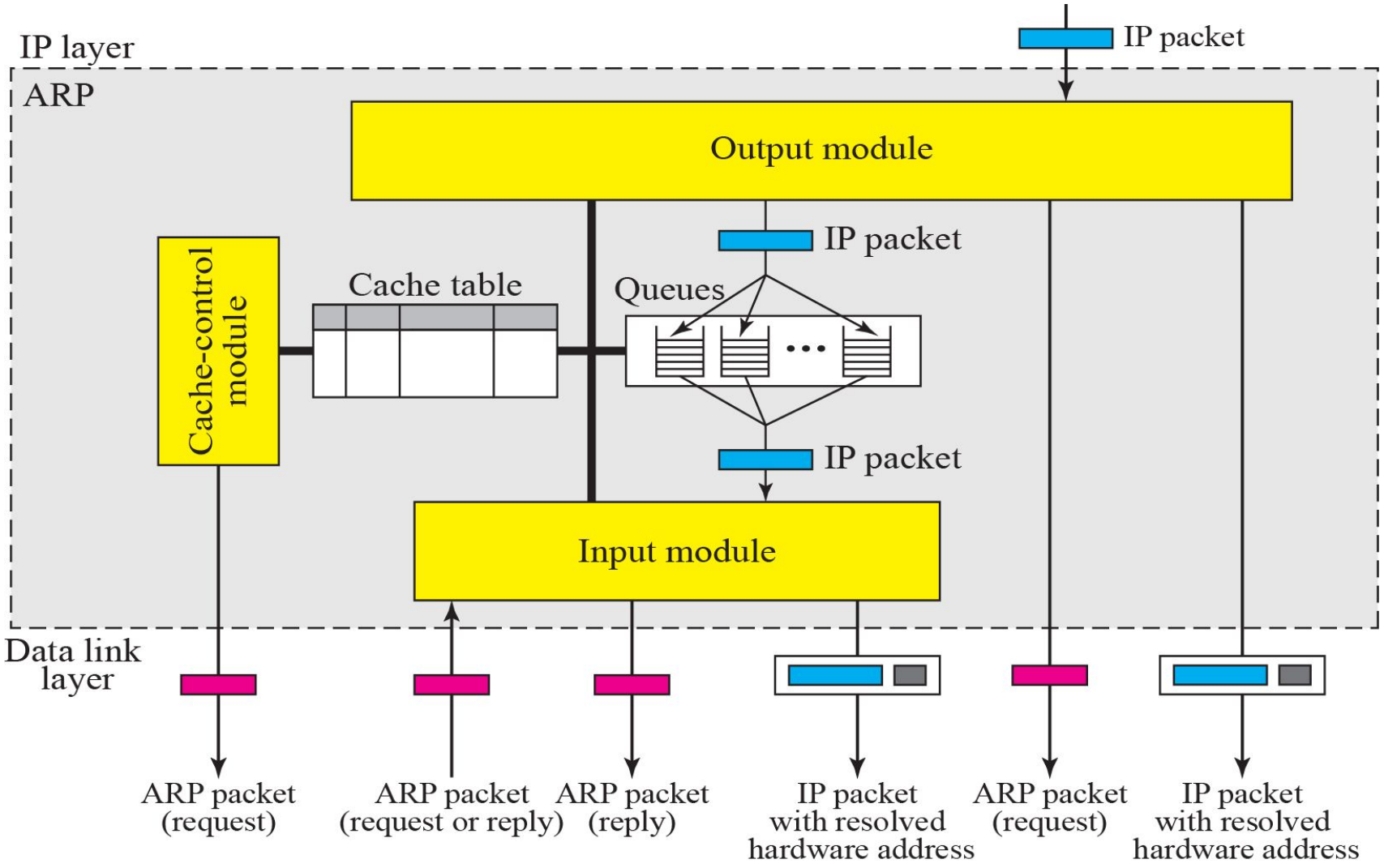


A proxy ARP is an ARP that acts on behalf of a set of hosts. Whenever a router running a proxy ARP receives an ARP request looking for the IP address of one of these hosts, the router sends an ARP reply announcing its own hardware (physical) address. After the router receives the actual IP packet, it sends the packet to the appropriate host or router.

ARP PACKAGE

we give an example of a simplified ARP software package.
The purpose is to show the components of a hypothetical ARP package and the relationships between the components.
We can say that this ARP package involves five components:
1) a cache table
2) Queues
3) an output module
4) an input module
5) a cache-control module.

Figure 1.32 ***ARP components***



ARP PACKAGE Cont.

Cache Table

A sender usually has more than one IP datagram to send to the same destination.

It is inefficient to use the ARP protocol for each datagram destined for the same host or router.

The solution is the cache table.

When a host or router receives the corresponding physical address for an IP datagram, the address can be saved in the cache table.

The cache table is implemented as an array of entries. In our package, each entry contains the following fields:

State: This column shows the state of the entry. It can have one of three values: FREE, PENDING, or RESOLVED.

Hardware type

Protocol type

Hardware length.

ARP PACKAGE Cont.

Protocol length

Interface number: A router (or a multihomed host) can be connected to different networks, each with a different interface number.

Each network can have different hardware and protocol types.

Queue number: ARP uses numbered queues to enqueue the packets waiting for address resolution.

Packets for the same destination are usually enqueued in the same queue.

Attempts: This column shows the number of times an ARP request is sent out for this entry.

Time-out: This column shows the lifetime of an entry in seconds.

Hardware address: This column shows the destination hardware address. It remains empty until resolved by an ARP reply.

Protocol address

Output Module

Table 8.2 *Output Module*

```
1   ARP_Output_Module ( )
2   {
3      Sleep until an IP packet is received from IP software.
4      Check cache table for an entry corresponding to the
5            destination of IP packet.
6      If (entry is found)
7      {
8           If (the state is RESOLVED)
9           {
10             Extract the value of the hardware address from the entry.
11             Send the packet and the hardware address to data
12                   link layer.
13             Return
14          } // end if
15          If (the state is PENDING)
16          {
17             Enqueue the packet to the corresponding queue.
18             Return
19        }//end if
20     }//end if
21     If (entry is not found)
22     {
23        Create a cache entry with state set to PENDING and
24              ATTEMPTS set to 1.
25        Create a queue.
26        Enqueue the packet.
27        Send an ARP request.
28        Return
29     }//end if
30  } //end module
```

Input Module

Table 8.3 *Input Module*

```
ARP_Input_Module ( )
{
    Sleep until an ARP packet (request or reply) arrives.
    Check the cache table to find the corresponding entry.
    If (found)
    {
        Update the entry.
        If (the state is PENDING)
        {
            While (the queue is not empty)
            {
                Dequeue one packet.
                Send the packet and the hardware address.
            }//end if
        }//end if
    }//end if
    If (not found)
    {
        Create an entry.
        Add the entry to the table.
    }//end if
    If (the packet is a request)
    {
        Send an ARP reply.
    }//end if
    Return
}//end module
```

Cache Control Module

Table 8.4 *Cache-Control Module*

```
1   ARP_Cache_Control_Module ( )
2   {
3        Sleep until the periodic timer matures.
4        Repeat for every entry in the cache table
5        {
6           If (the state is FREE)
7           {
8              Continue.
9           }//end if
10           If (the state is PENDING)
11           {
```

Table 8.4 *Cache-Control Module (continued)*

```
12              Increment the value of attempts by 1.
13              If (attempts greater than maximum)
14              {
15                   Change the state to FREE.
16                   Destroy the corresponding queue.
17              }// end if
18              else
19              {
20                 Send an ARP request.
21               }//end else
22              continue.
23            }//end if
24            If (the state is RESOLVED)
25            {
26                Decrement the value of time-out.
27                If (time-out less than or equal 0)
28                {
29                    Change the state to FREE.
30                    Destroy the corresponding queue.
31                }//end if
32            }//end if
33        }//end repeat
34        Return.
35    }//end module
```

Example 8.2

The ARP output module receives an IP datagram (from the IP layer) with the destination address 114.5.7.89. It checks the cache table and finds that an entry exists for this destination with the RESOLVED state (R in the table). It extracts the hardware address, which is 457342ACAE32, and sends the packet and the address to the data link layer for transmission. The cache table remains the same.

Table 8.5 *Original cache table used for examples*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
F					
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

Example 8.3

Twenty seconds later, the ARP output module receives an IP datagram (from the IP layer) with the destination address 116.1.7.22. It checks the cache table and does not find this destination in the table. The module adds an entry to the table with the state PENDING and the Attempt value 1. It creates a new queue for this destination and enqueues the packet. It then sends an ARP request to the data link layer for this destination. The new cache table is shown in Table 8.6.

Table 8.6 *Updated cache table for Example 8.3*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
P	23	1		116.1.7.22	
R	9		60	19.1.7.82	4573E3242ACA
P	18	3		188.11.8.71	

Example 8.4

Fifteen seconds later, the ARP input module receives an ARP packet with target protocol (IP) address 188.11.8.71. The module checks the table and finds this address. It changes the state of the entry to RESOLVED and sets the time-out value to 900. The module then adds the target hardware address (E34573242ACA) to the entry. Now it accesses queue 18 and sends all the packets in this queue, one by one, to the data link layer. The new cache table is shown in Table 8.7.

Table 8.7 *Updated cache table for Example 8.4*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		900	180.3.6.1	ACAE32457342
P	2	2		129.34.4.8	
P	14	5		201.11.56.7	
R	8		450	114.5.7.89	457342ACAE32
P	12	1		220.55.5.7	
P	23	1		116.1.7.22	
R	9		60	19.1.7.82	4573E3242ACA
R	18		900	188.11.8.71	E34573242ACA

Twenty-five seconds later, the cache-control module updates every entry. The time-out values for the first three resolved entries are decremented by 60. The time-out value for the last resolved entry is decremented by 25. The state of the next-to-the last entry is changed to FREE because the time-out is zero. For each of the three pending entries, the value of the attempts field is incremented by one. After incrementing, the attempts value for one entry (the one with IP address 201.11.56.7) is more than the maximum; the state is changed to FREE, the queue is deleted, and an ICMP message is sent to the original destination See Table 8.8.

Table 8.8 *Updated cache table for Example 8.5*

State	*Queue*	*Attempt*	*Time-Out*	*Protocol Addr.*	*Hardware Addr.*
R	5		840	180.3.6.1	ACAE32457342
P	2	3		129.34.4.8	
F					
R	8		390	114.5.7.89	457342ACAE32
P	12	2		220.55.5.7	
P	23	2		116.1.7.22	
F					
R	18		875	188.11.8.71	E34573242ACA

Reverse Address Resolution Protocol

RARP Procedure

Steps to Achieve the IP Address from RARP Server:

1. **Source Device "Generates RARP Request Message"** – The source device generates a RARP Request message. The Source puts its own data link-layer address as both the Sender Hardware Address and also the Target Hardware Address. It leaves both the Sender Protocol Address and the Target Protocol Address blank.
2. **Source Device "Broadcasts RARP Request Message"** – The source broadcasts the ARP Request message on the local network.
3. **Local Devices "Process RARP Request Message**" – The message is received by each device on the local network and processed. Devices that are not configured to act as RARP servers ignore the message.

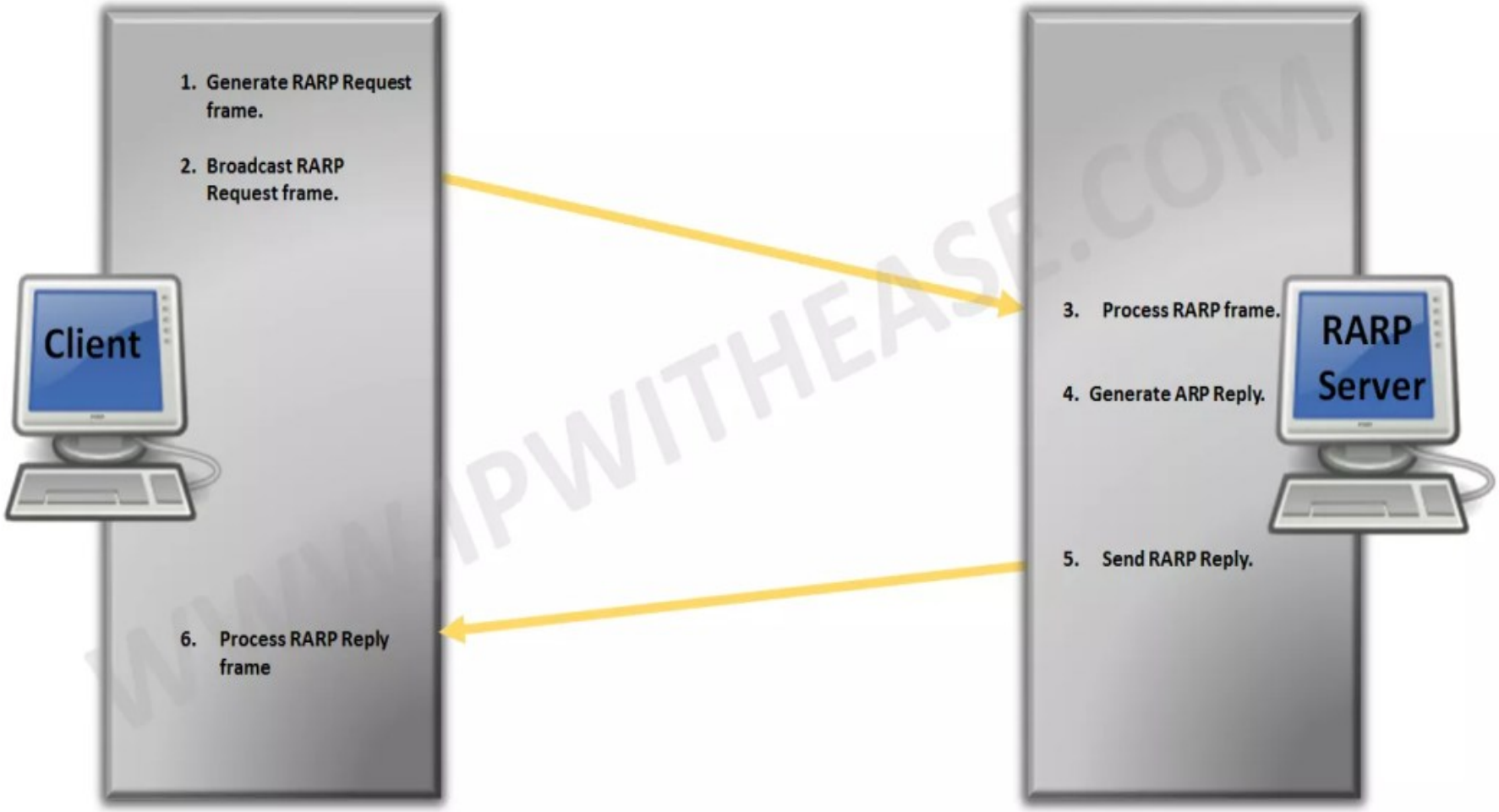
RARP Procedure

4. RARP Server Generates RARP Reply Message: Any device on the network that is a RARP server responds to the broadcast from the source device. It generates a RARP Reply and sets the Sender Hardware Address and Sender Protocol Address to its own hardware and IP address of course. It looks up in a table the hardware address of the source, determines that device's IP address assignment, and puts it into the Target Protocol Address field.

5. RARP Server Sends RARP Reply Message: The RARP server sends the RARP Reply message unicast to the device looking to be configured.

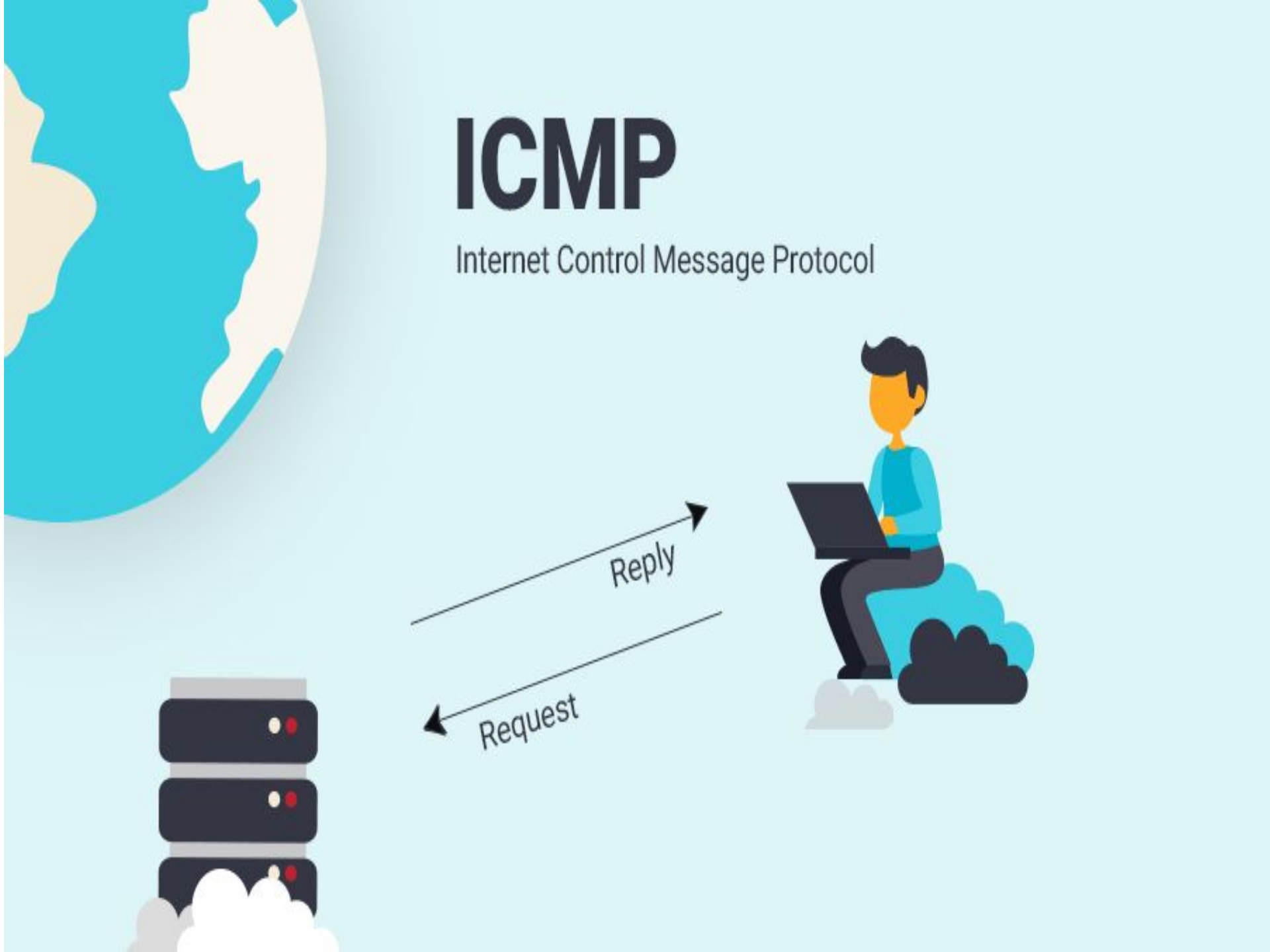
6. Source Device Processes RARP Reply Message: The source device processes the reply from the RARP server. It then configures itself using the IP address in the Target Protocol Address supplied by the RARP server

RARP Procedural Steps



ICMP

Internet Control Message Protocol

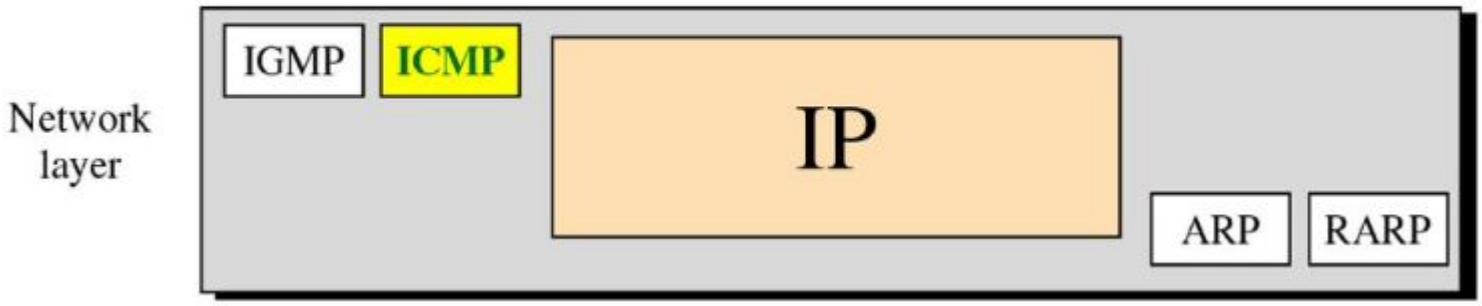


Introduction to Internet Control Message Protocol (ICMP)

- IP protocol has no error-reporting or error-correcting mechanism
  - When errors occur, no built-in mechanism to notify the original host
- IP protocol also lacks a mechanism for host and management queries
  - A host sometimes needs to determine if a router or another host is alive
  - Network manager needs information from another host and router

Introduction to Internet Control Message Protocol (ICMP)

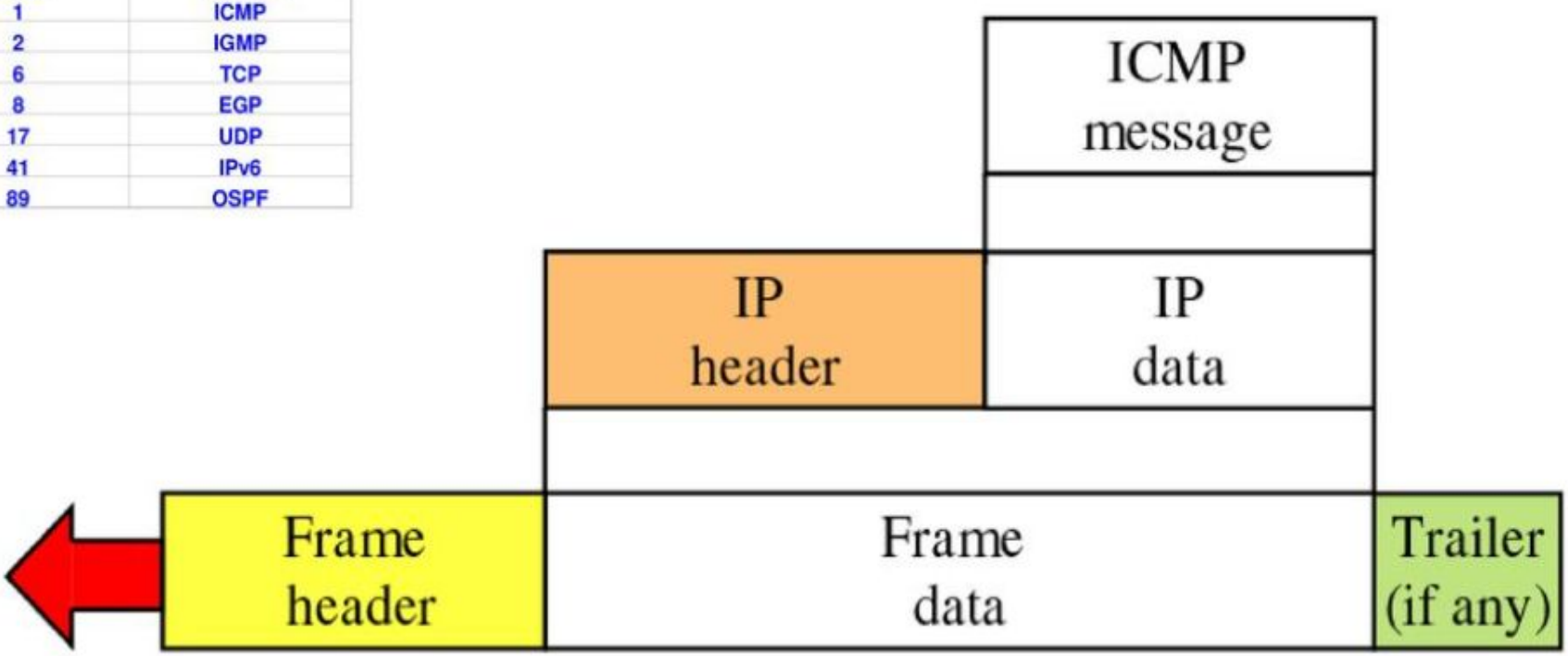
- Position of ICMP in the network layer



Introduction to Internet Control Message Protocol (ICMP)

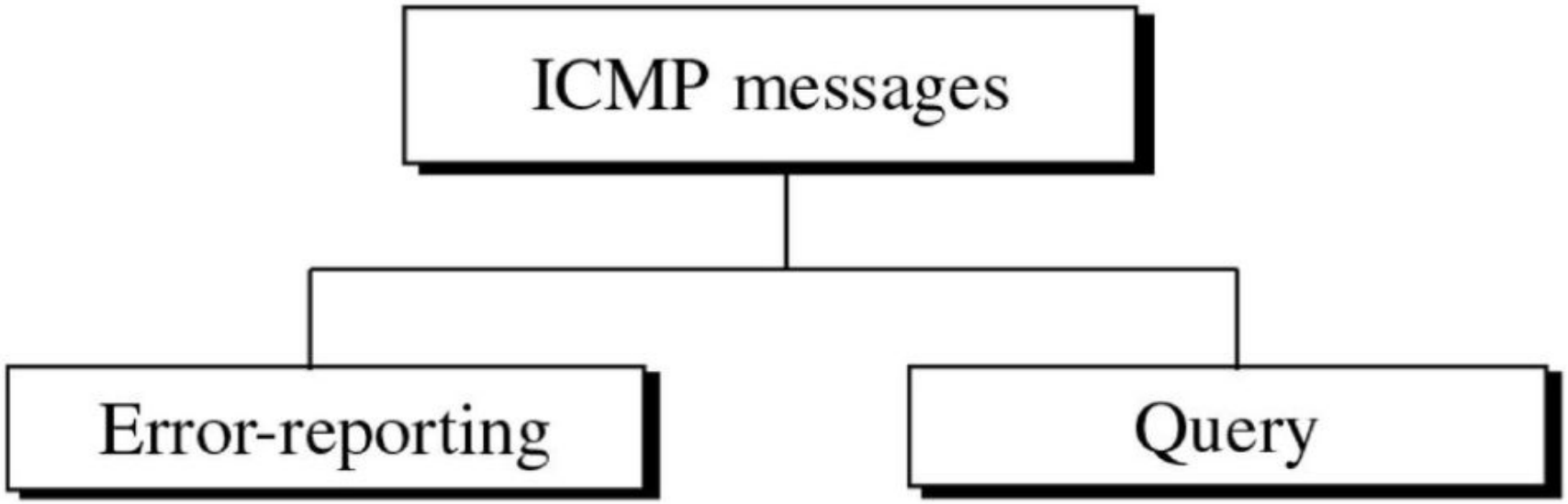
- ICMP encapsulation
  - The value of the protocol field in the IP datagram : 1 (table 7.3)

Value	Protocol
1	ICMP
2	IGMP
6	TCP
8	EGP
17	UDP
41	IPv6
89	OSPF



Types of Messages

- Category of ICMP messages



ICMP MESSAGE TYPES

Type Field	ICMP Message Type
0	Echo Reply
3	Destination Unreachable
4	Source Quench
5	Redirect (change a route)
6	Alternate Host Address
8	Echo Request
9	Router Advertisement
10	Router Solicitation
11	Time Exceeded for a Datagram
12	Parameter Problem on a Datagram
13	Timestamp Request
14	Timestamp Reply
15	Information Request
16	Information Reply
17	Address Mask Request
18	Address Mask Reply

Type Field	ICMP Message Type
30	Traceroute
31	Datagram Conversion Error
32	Mobile Host Redirect
33	IPv6 Where-Are-You
34	IPv6 I-Am-Here
35	Mobile Registration Request
36	Mobile Registration Reply
37	Domain Name Request
38	Domain Name Reply
39	SKIP
40	Photuris

Types of Messages

- CMP messages
  - Error reporting messages

Type	Message
3	Destination unreachable
4	Source quench
11	Time Exceeded
12	Parameter problem
5	Redirection

Types of Messages

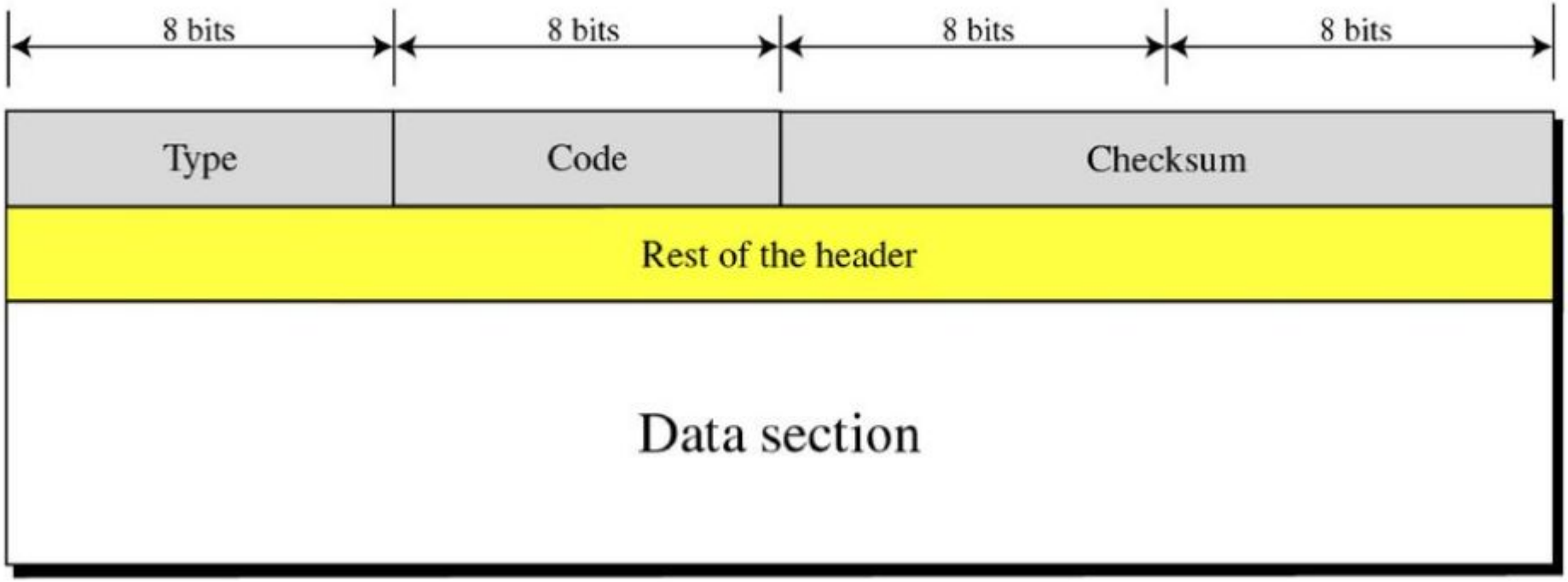
- ICMP messages
  - Query messages

Type	Message
8 or 0	Echo request or reply
13 or 14	Timestamp request and reply
17 or 18	Address mask request and reply
10 or 9	Router solicitation and advertisement

Message Format

- **Having 8 byte header and variable-size data section**
  - ICMP type : defining the type of the message
  - Code field : specifying the reason for the particular message type
  - Checksum field (for header and message)
  - **Data section**
    - In error message, carrying information for finding the original packet which caused the error
    - In query message, carrying extra information based on the type of the query

Message Format



Error Reporting

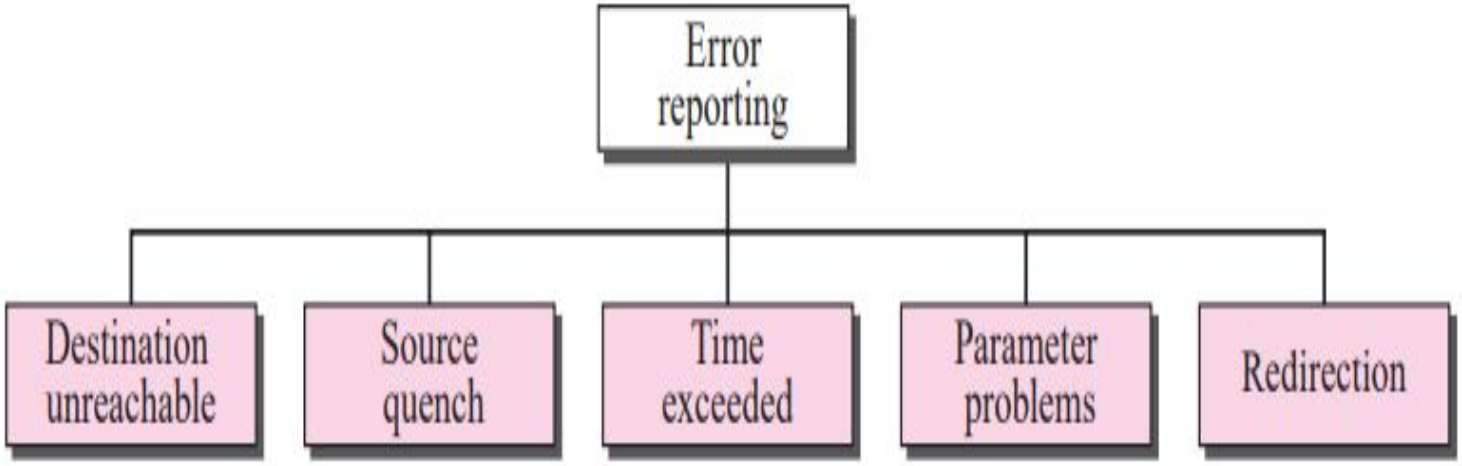
- Error checking and control
- Not correcting errors : it is left to the higher level protocols
- Always reporting error messages to the original source

Error Reporting (cont'd)

- Error checking and control
- Not correcting errors : it is left to the higher level protocols
- Always reporting error messages to the original source

Error Reporting (cont'd)

- Error-reporting messages



Error Reporting (cont'd)

Important points about ICMP error messages

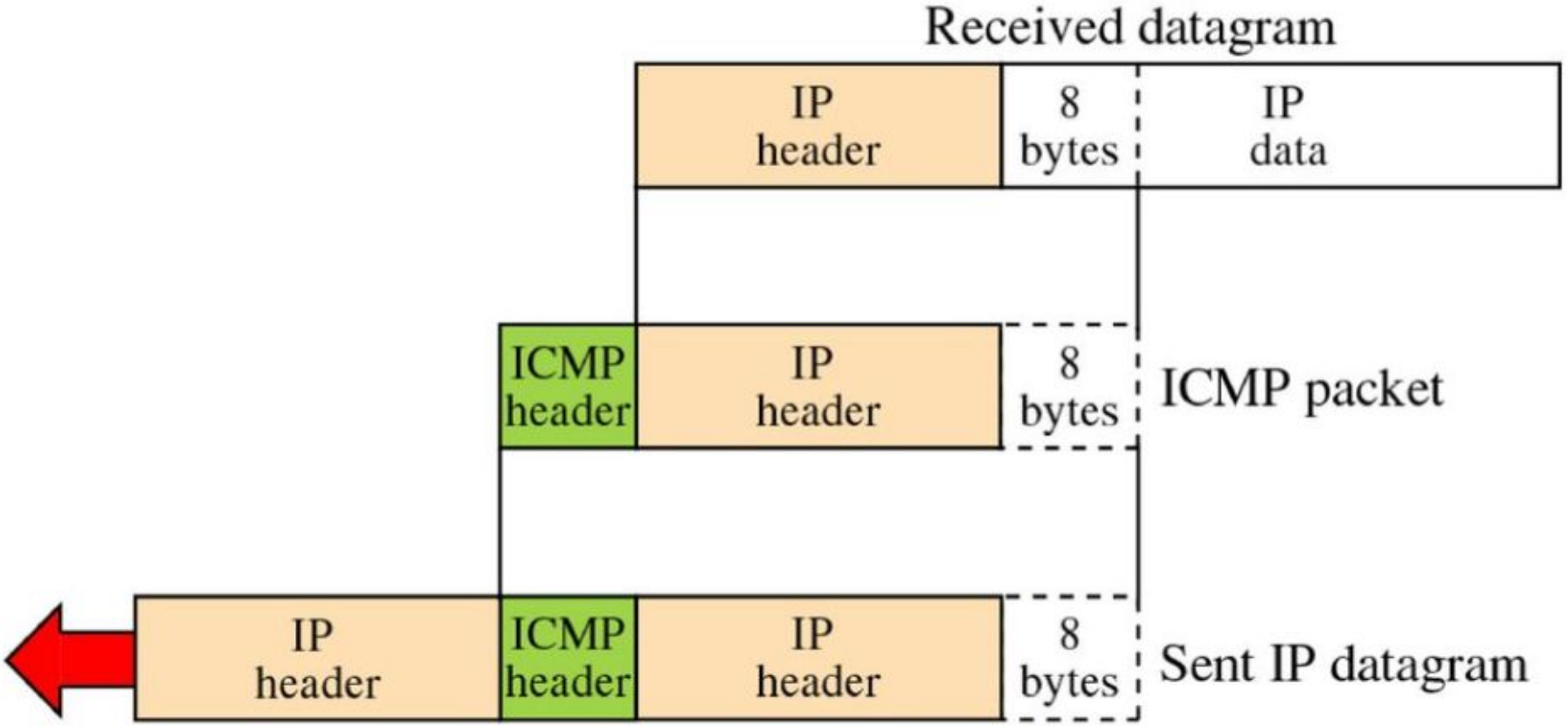
- No ICMP error message will be generated in response to a datagram carrying an
- ICMP error message.
- No ICMP error message will be generated for a fragmented datagram that is not
- the first fragment.
- No ICMP error message will be generated for a datagram having a multicast
- address.
- No ICMP error message will be generated for a datagram having a special address
- such as 127.0.0.0 or 0.0.0.0.

Error Reporting (cont'd)

- All error messages
  - containing a data section that includes the IP header of the original datagram + the first 8 bytes of data in that IP datagram
  - 8 bytes of data : port # (UDP and TCP) and sequence # (TCP)
    - Used for informing to the protocols (TCP or UDP) about the error situation

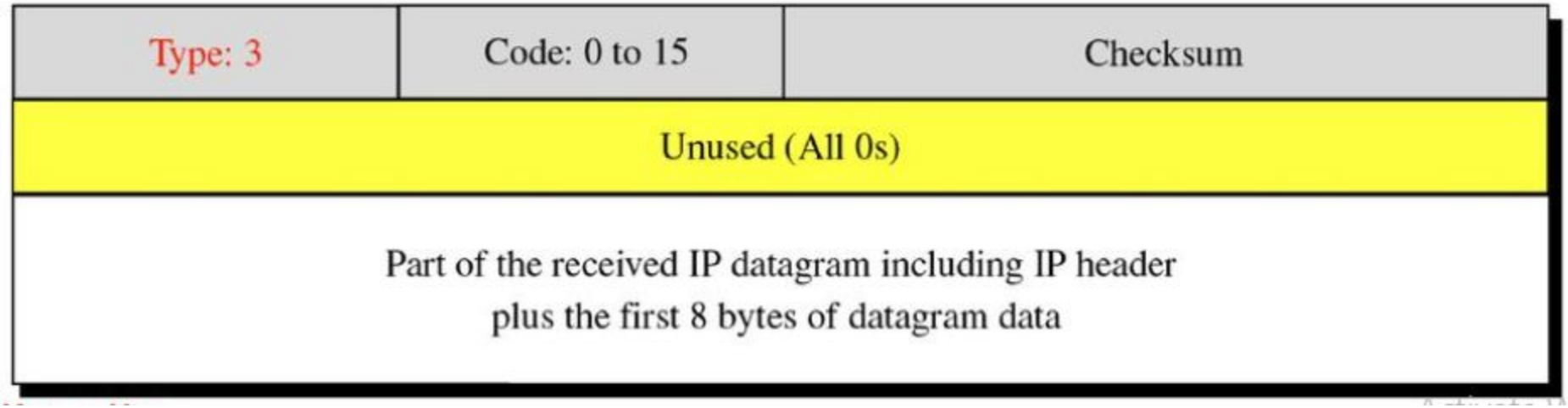
Error Reporting (cont'd)

- Contents of data field for the error messages



Error Reporting (cont'd)

- Destination Unreachable
- When a router cannot route a datagram or a host cannot deliver a datagram, the datagram is discarded.
- Then, the router or the host sends a destination unreachable message back to the source that initiated the datagram.
- Destination unreachable format



Error Reporting (cont'd)

Code 0 : network is unreachable, due to hardware failure, can only be generated by a router

Code 1 : host is unreachable, due to hardware failure, can only be generated by a router

Code 2 : protocol such as UDP, TCP or OSPF is not running at the moment.

- generated only by the destination

Code 3 : the application program (process) that the datagram is destined for is not running at the moment

Code 4 : Fragmentation is required, but the DF (do not fragment) field has been set

Code 5 : Source routing cannot be accomplished

Code 6 : The destination network is unknown.

- A router has no information about the destination network

Error Reporting (cont'd)

Code 7 : The destination host is unknown.

- the router is unaware of the existence of the destination

Code 8 : The source host is isolated

Code 9 : Communication with the destination network is administratively prohibited

Code 10 : Communication with the destination host is administratively prohibited

Code 11 : the network is unreachable for the specified type of service

Code 12 : The host is unreachable for the specified type of service

Error Reporting (cont'd)

Code 13 : The host is unreachable because the administration has put a filter on it

Code 14 : The host is unreachable because the host precedence is violated. The requested precedence is not permitted for the destination

Code 15 : The host is unreachable because its precedence was cut off. This message is generated when the network operators have imposed a minimum level of precedence for the operation of the network

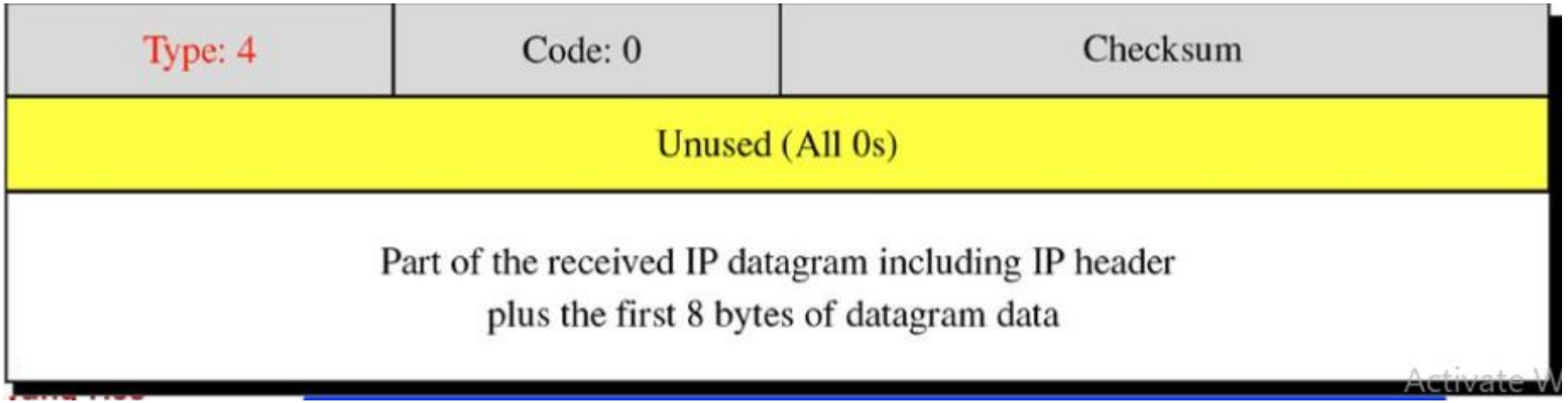
Error Reporting (cont'd)

- Destination-unreachable messages with codes 2 or 3 can be created only by the destination host. Other destination-unreachable message can be created only by routers.
- A router can not detect all problems that prevent the delivery of a packet.
- The case that a datagram is traveling through an Ethernet network.
- Ethernet does not provide any acknowledgement mechanism.

Error Reporting (cont'd)

Source Quench

- is designed to add a kind of flow control to the IP
  - IP does not have a flow-control mechanism embedded in the protocol
- when a router or host discards a datagram due to congestion, it sends a source-quench message to the sender of the datagram
  - making slow down the sending process



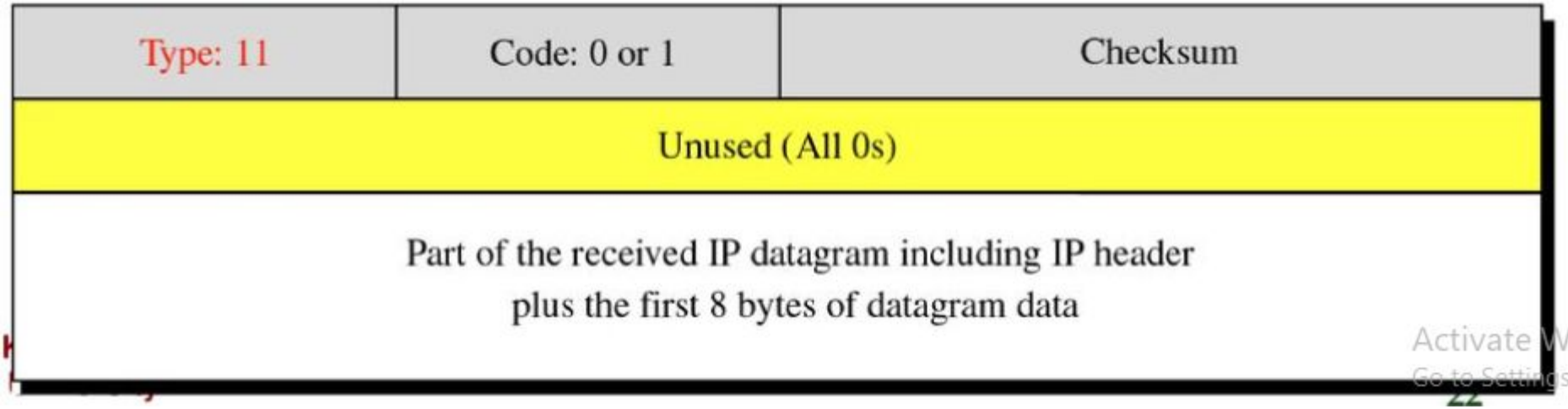
Error Reporting (cont'd)

Time exceeded

- Whenever a router receives a datagram whose time-to-live field has the value of zero, it discards the datagram and sends a time-exceeded message to the original source
- When the final destination does not receive all of the fragments in a set time, it discards the received fragments and sends a time-exceeded message to the original source

Error Reporting (cont'd)

- In a time-exceeded message, code 0 is used only by routers to show that the value of the time-to-live field is zero. Code 1 is used only by the destination host to show that not all of the fragments have arrived within a set time
- Time-exceeded message format



Error Reporting (cont'd)

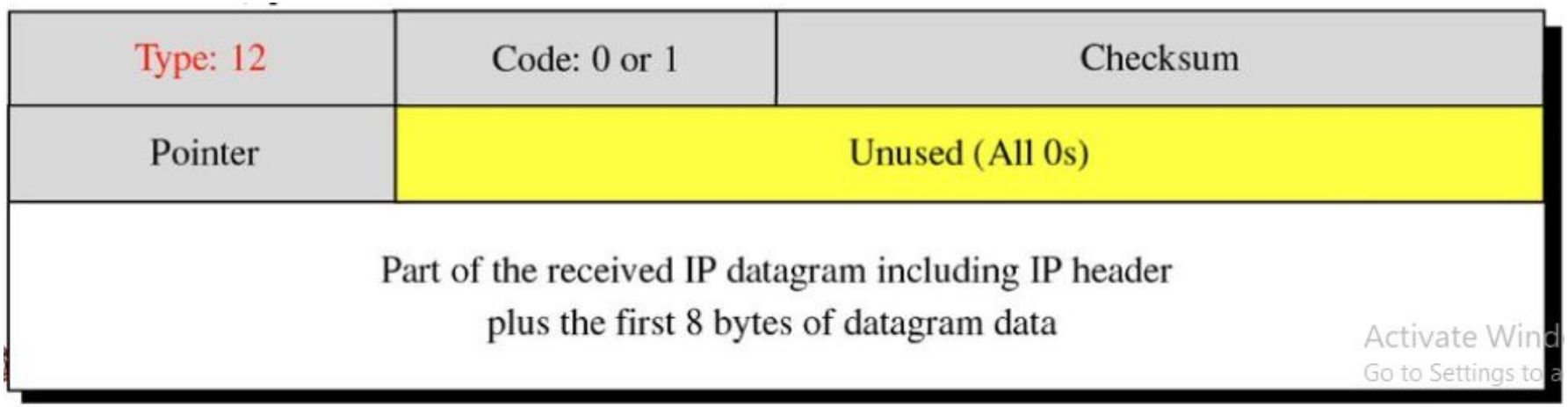
Parameter-problem

- A parameter-problem message caused by ambiguity in the header part can be created by a router or the destination host

Code 0 : error or ambiguity in one of the header fields

- the value in the pointer field points to the byte with the problem

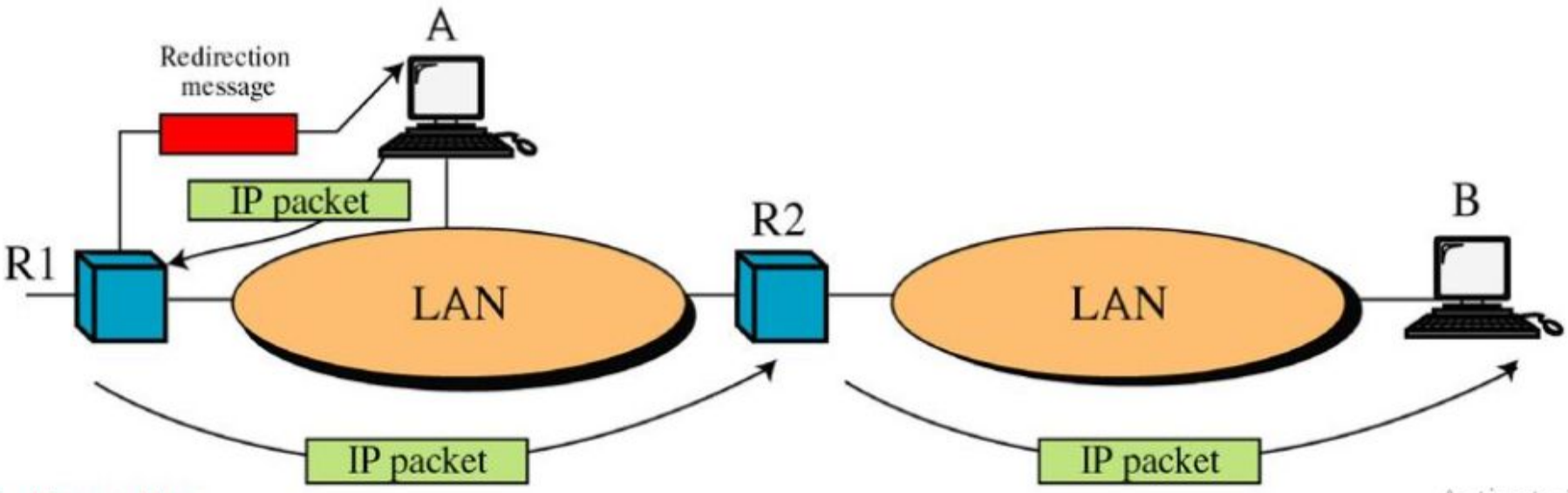
Code 1 : the required part of an option is missing. In this case, pointer is not used



Error Reporting (cont'd)

Redirection

- A host usually starts with a small routing table that is gradually augmented and updated. One of the tools to accomplish this is the redirection message.
- A redirection message is sent from a router to a host on the same local network.



Error Reporting (cont'd)

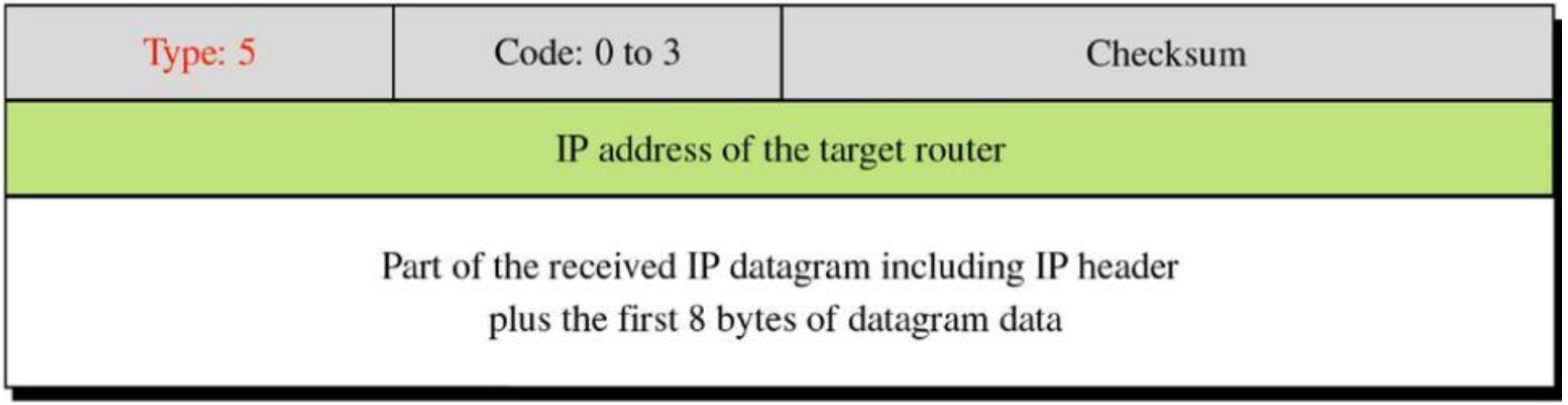
Redirection message format

Code 0 : redirection for the network-specific route

Code 1 : redirection for the host-specific route

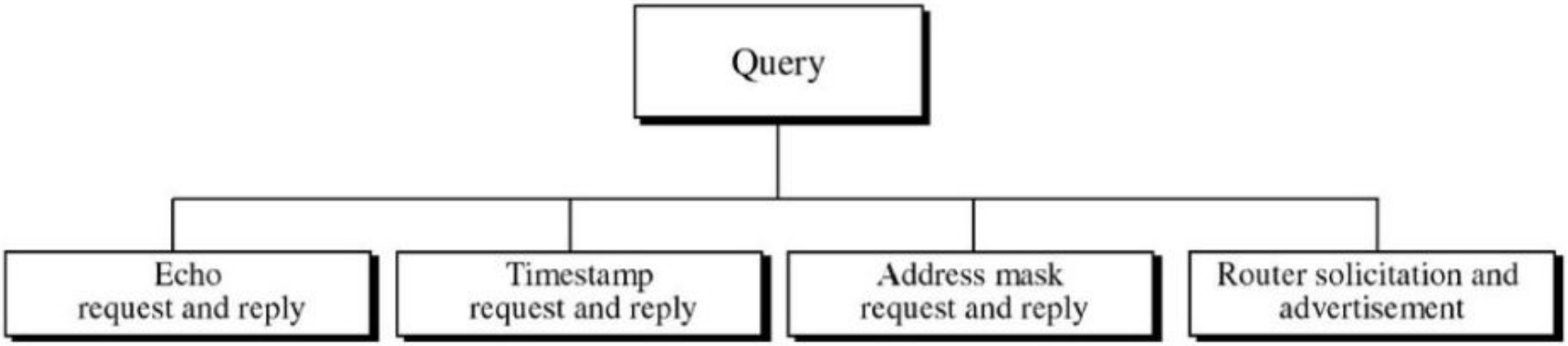
Code 2 : redirection for network-specific route based on specific type of service

Code 3 : redirection for the host-specific route based on the specified type of servic



Query

- Diagnosing some network problems
- 4 different pairs of messages



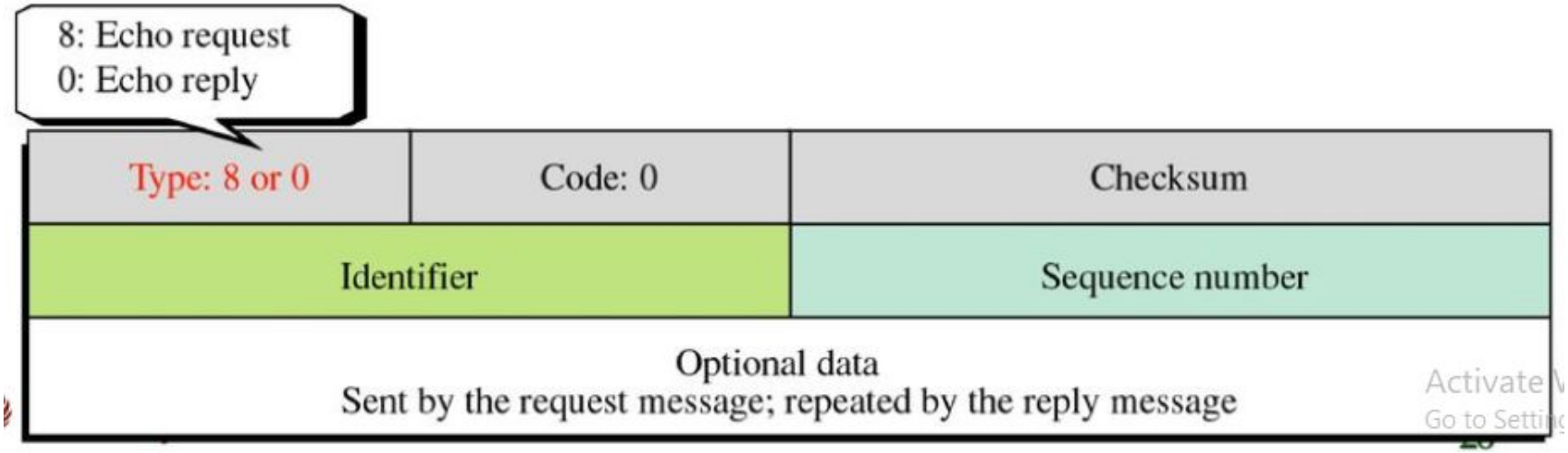
Query (cont'd)

Echo Request and Reply messages

- designed for diagnostic purpose
- the combination of echo-request and echo-reply messages determines whether 2 systems (hosts or routers) can communicate with each other
- An echo-request message can be sent by a host or router. An echo-reply message is sent by the host or router which receives an echo-request message
- Echo-request and echo-reply message can be used by network managers to check the operation of the IP protocol

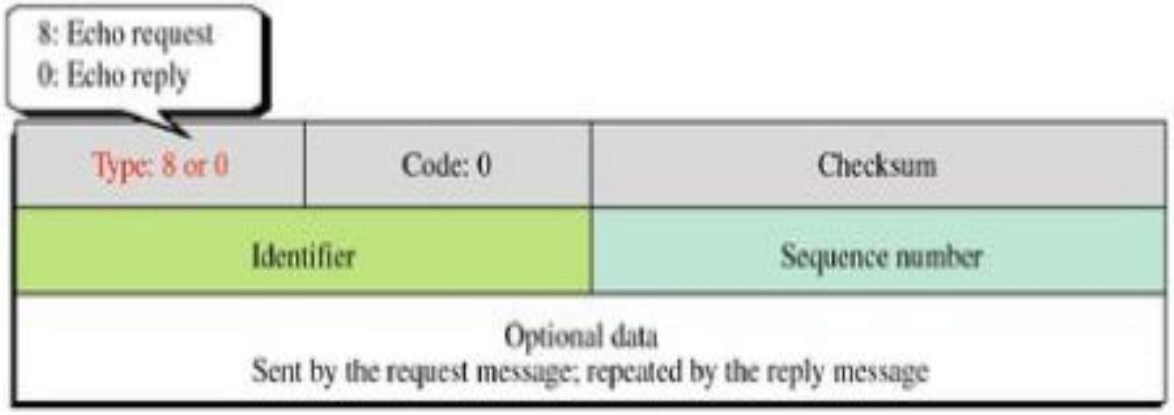
Query (cont'd)

- Echo-request and echo-reply messages can test the reachability of a host. This is usually done by invoking the ping command
- Identifier and sequence number fields are not formally defined by the protocol and can be used by the sender
- Echo-request and echo-reply message



Query (cont'd)

- The identifier field The sequence number field
  - defines a group of problems
  - ex) process ID that originated the request
- The sequence number field
  - keeps track of the particular echo request messages sent
- At the user level
  - Invoking the packet Internet groper (ping) command



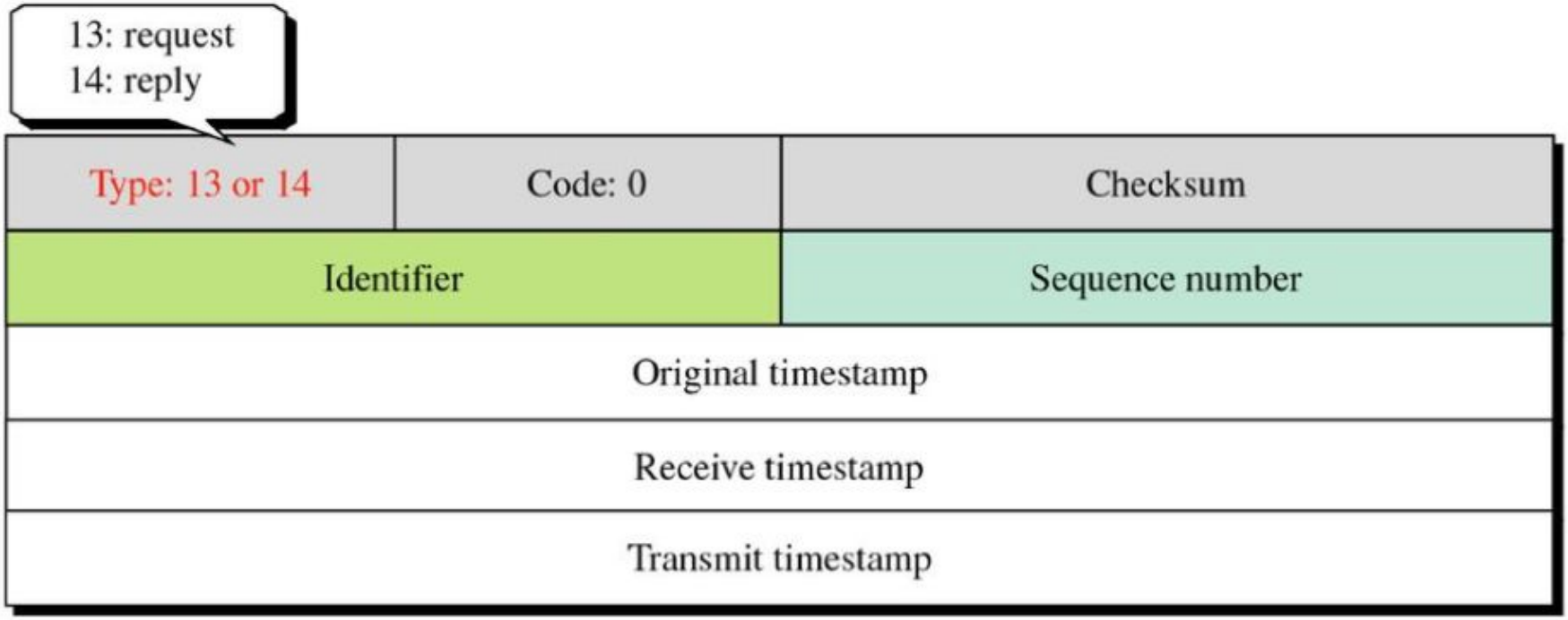
Query (cont'd)

Timestamp Request and Reply

- 2 machines (routers or hosts) can use the timestamp-request and timestamp-reply messages to determine the round-trip time needed for an IP datagram to travel between them
- can used to synchronize the clocks in two machines
- Three timestamp fields are each 32 bits long
  - holding a number representing time measured in milliseconds from midnight in Universal Time
  - Cannot exceed 86,400,000 = 24 x 60 x 60 x 1,000

Query (cont'd)

Timestamp-request and reply message format



- original timestamp field : clock at departure time
- receive timestamp field : at the time the request was received
- transmit timestamp field : at the time the reply message departs

Query (cont'd)

The formulas for computing the one-way or round-trip time required for a datagram to go from a source to a destination and then back again.

- Sending time = value of receive timestamp – value of original time stamp
- Receiving time = time the packet returned – value of transmit timestamp
- Round-trip time = sending time + receiving time

Query (cont'd)

Timestamp-request and timestamp reply message can be used to measure the round-trip time between a source and a destination machine even if their clocks are not synchronized

Example

Value of original timestamp : 46

Value of receive timestamp : 59

Value of transmit timestamp : 60

Time the packet arrived : 67

Sending time = 13 ms

Receiving time = 7 ms

Round-trip time = 20 ms

Query (cont'd)

Synchronizing clocks between two machines

Time difference = receive timestamp – (original timestamp field + oneway time duration)

In previous example,

Time difference = 59 – (46+10) = 3

Query (cont'd)

Address Mask Request and Reply

- for differentiating among network address, subnetwork address and host ID
- example, a host may know its 32-bit IP address as

10011111.00011111.11100010.10101011

- left 20 bits are network and subnetwork addresses and remaining 12 bits are Host ID. In this case, following mask

11111111.11111111.11110000.00000000

NetId and subnetid → 10011111.00011111.1110

Host ID → 0010.10101011

Query (cont'd)

To obtain its mask,

- A host sends an address-mask-request message to a router on the LAN. (unicast or broadcast)
- If the host knows the address of the router, it sends the request directly to the router, if not, it broadcasts the message.

Query (cont'd)

- Masking is needed for diskless stations at start-up time.
- When a diskless station comes up for the first time
  - it may ask for its full IP address using RARP protocol
  - after receiving its IP address, it may use the address mask request and reply to find out which part of the address defines the subnet

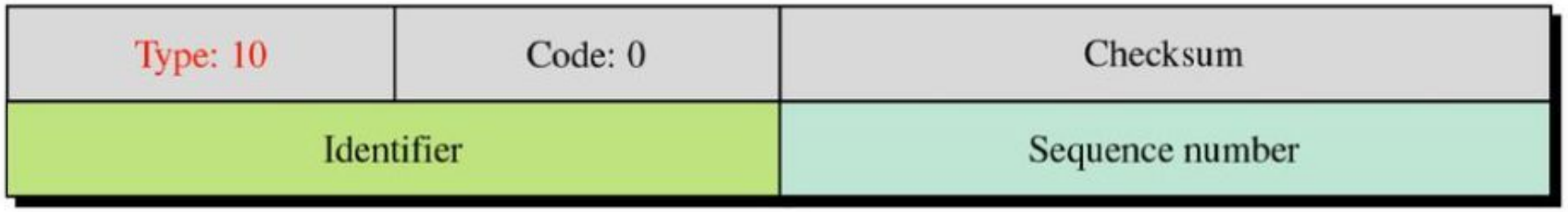
Query (cont'd)

Router Solicitation and Advertisement

- A host that wants to send data to a host on another network needs to know the address of routers connected to its own network.
  - the host should know if the routers are alive and functioning
  - A host can broadcast (or multicast) a router-solicitation message.
  - The router or routers that receive the solicitation message broadcast their routing information using the router-advertisement message.
    - A router can also periodically advertise router-advertisement messages even if no host has solicited

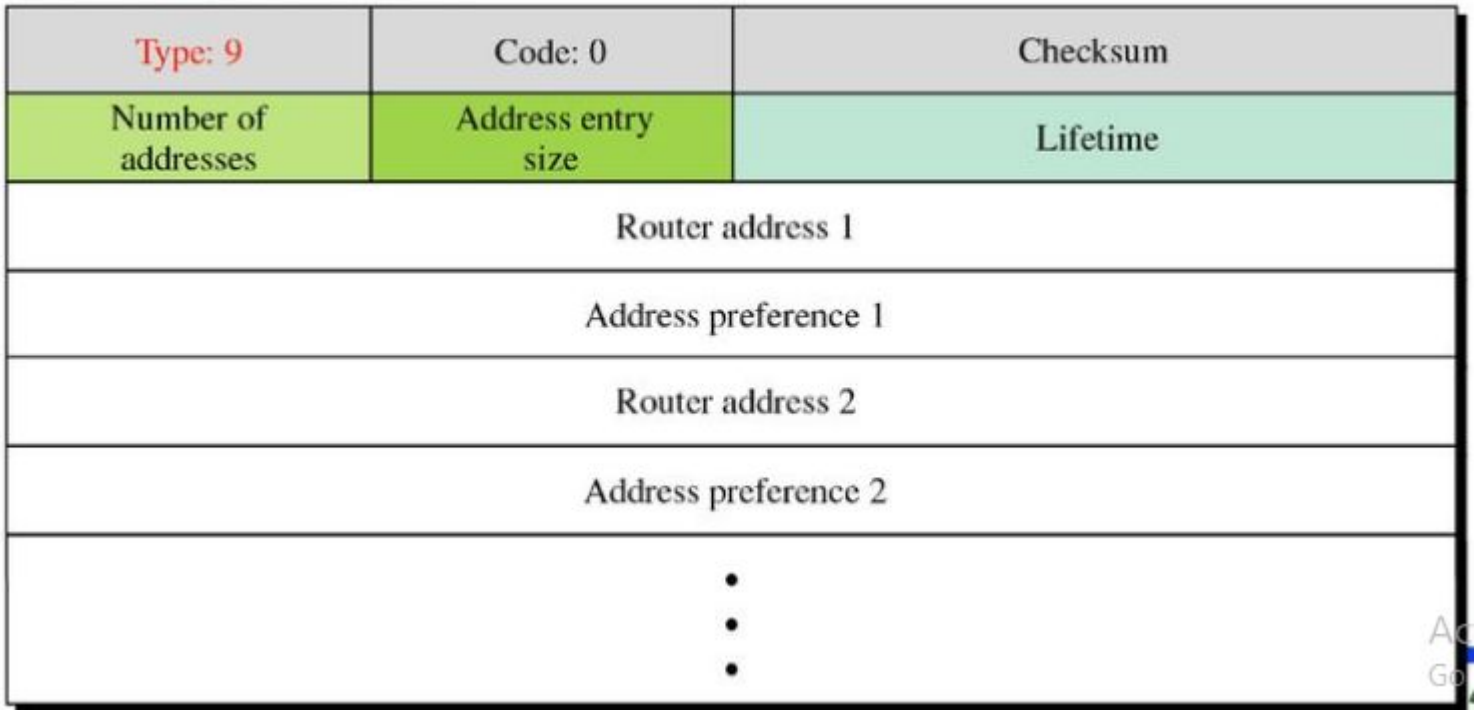
Query (cont'd)

- Router-solicitation message format



Query (cont'd)

- Router-advertisement message format
  - lifetime field : showing the number of seconds that entries are considered to be valid
  - address preference level defines the ranking of the router
    - preference level 0 : default router
    - preference level : the router should never be selected as the default router



Checksum

- calculating over the entire message (header and data)
- Checksum calculation
  1. Checksum field is set to zero
  2. Sum of all the 16-bit words (header and data) is calculated
  3. Sum is complemented to get the checksum
  4. Checksum is stored in the checksum field

Checksum (cont'd)

Checksum testing Example,

1. the sum of all words (header and data) is calculated
2. the sum is completed
3. if the result obtained in step 2 is 16 0s, the message is accepted; otherwise, it is rejected.

Example,

8	0	0
1		9
TEST		

8 and 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

ICMP– DEBUGGING TOOL

- To check whether host or router is alive and running
- To trace the route of a packet.
- Two tools that use ICMP for debugging: ping and traceroute

Ping

- The ping program to find if a host is alive and responding.
- Command : ping the ip of the host.(ping 152.18.1.3)
- The source host sends ICMP echo request messages (type: 8, code: 0);
- The destination, if alive, responds with ICMP echo reply messages.

ICMP– DEBUGGING TOOL

- **Ping cont…**
- Starts the sequence number from 0; this number is incremented by one each time a
- new message is sent.
- ping can calculate the round-trip time.
- Inserts the sending time in the data section of the message.
- When packet arrives it subtracts the arrival time from the departure time to get the

Round-Trip Time (RTT).

- The TTL (time to live) field is encapsulates an ICMP message as 62, which means the
- packet cannot travel more than 62 hops

ICMP– DEBUGGING TOOL

Example : $ 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=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=1 ttl=62 time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2 ttl=62 time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3 ttl=62 time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4 ttl=62 time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5 ttl=62 time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6 ttl=62 time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=7 ttl=62 time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=8 ttl=62 time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9 ttl=62 time=1.89 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=10 ttl=62 time=1.98 ms
--- fhda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103 ms
rtt min/avg/max = 1.899/1.955/2.041 ms
```

ICMP– DEBUGGING TOOL

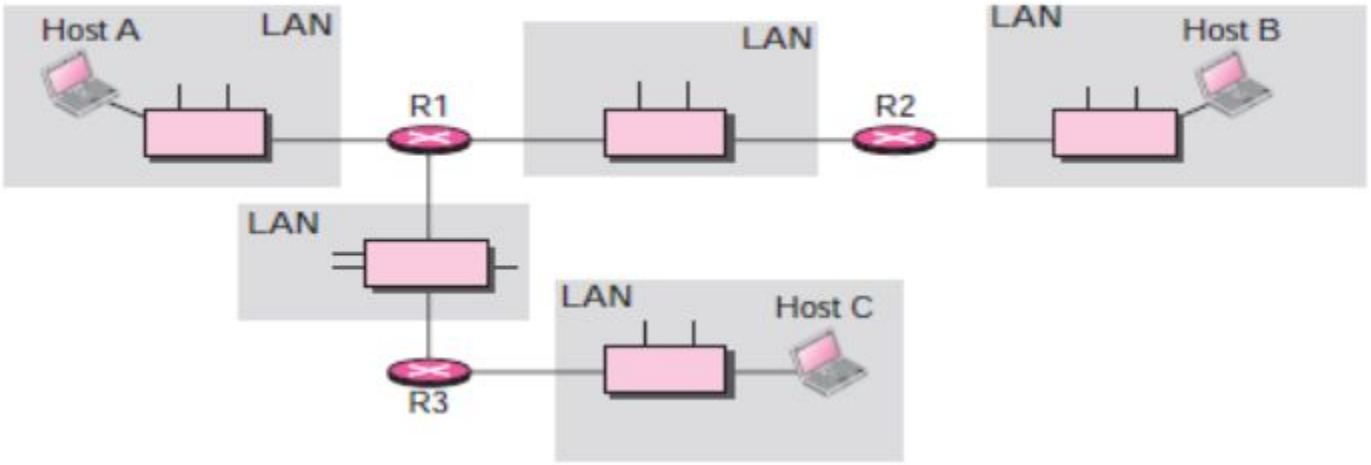
Ping cont…

- Ping data bytes as 56 and the total number of bytes as 84.
- 8 bytes ICMP header+ 20 bytes of IP header to 56 =84
- ping defines the number of bytes as 64(56 + 8).
- Interrupts message ctrl+c.
- it prints the statistics of the probes
  - number of packets sent,
  - the number of packets received.
  - the total time
  - the RTT minimum, maximum, and average.

ICMP– DEBUGGING TOOL

- TRACE ROUTE
- The traceroute program in UNIX or tracert in Windows.
- It is used to route the packets from source to destination.

Example Scenario



The Traceroute Program Operation

ICMP– DEBUGGING TOOL

Trace route cont….

- In above example Given the topology, A packet from host A to host B travels through routers R1 and R2.
- The traceroute program find the address of router R1 & RTT between host A and router R1.
- The program repeats steps a to c three times to get a better average round-trip time.
  - a. Host A sends a packet to destination B using UDP the message is
  - encapsulated in an IP packet with a TTL value of 1. The program notes the time the packet is sent

ICMP– DEBUGGING TOOL

Trace route cont.….

b. Router R1 receives the packet and decrements the value of TTL to 0. It then discards the packet (because TTL is 0).

c. In receiver the ICMP messages uses the source address of the IP packet to find the IP address of router R1 and also makes note of the time the packet has arrived.

- The traceroute program repeats the previous steps to find the address of router R2 and the round-trip time between host A and router R2.
- The round-trip time between host A and host B.

ICMP– DEBUGGING TOOL

Trace route cont.….

Example: The traceroute program to find the route from the computer voyager.deanza.edu to the server fhda.edu. The following shows the result.

```
$ traceroute fhda.edu
traceroute to fhda.edu (153.18.8.1), 30 hops max, 38 byte packets
1 Dcore.fhda.edu (153.18.31.25) 0.995 ms 0.899 ms 0.878 ms
2 Dbackup.fhda.edu (153.18.251.4) 1.039 ms 1.064 ms 1.083 ms
3 tiptoe.fhda.edu (153.18.8.1) 1.797 ms 1.642 ms 1.757 ms
```

- In the above example the destination is 153.18.8.1.
- TTL value is 30 hops.
- The packet contains 38 bytes: 20 bytes of IP header, 8 bytes of UDP header, and 10 bytes of application data.

ICMP– DEBUGGING TOOL

Trace route cont.….

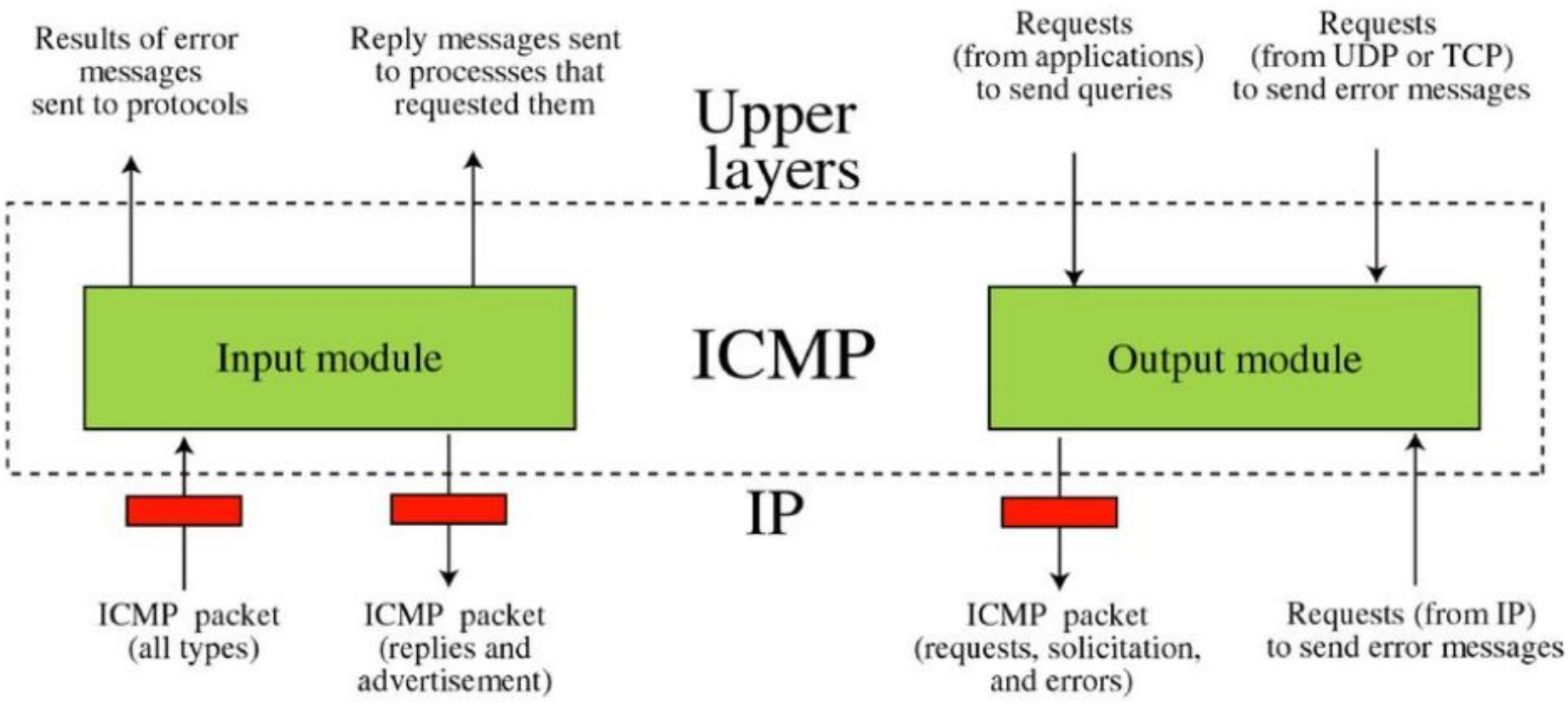
- The router is named Dcore.fhda.edu with IP address 153.18.31.254.
- The Round Trip Time
    1. 0.995 milliseconds,
    2. 0.899 milliseconds
    3. 0.878 milliseconds
- The router is named Dbackup.fhda.edu with IP address 153.18.251.4.
- The third line shows the destination host

ICMP DESIGN

To handle the ICMP sending and receiving messages

ICMP package made of two modules:

- input module
- output module



ICMP Design (cont'd)

Input module

- handling all received ICMP message
- invoked when an ICMP packet is delivered to it from the IP layer
- if the received packet is a request or solicitation, the module creates a reply or an advertisement and sends it out
- if the received packet is a redirection message, the module uses the information to update the routing table
- if the received packet is an error message, the module informs the protocol about the situation that caused the error

INPUT MODULE

Pseudocode for Input Module

```
1   ICMP_Input_module (ICMP_Packet)
2   {
3      If (the type is a request)
4      {
5           Create a reply
6           Send the reply
7      }
8      If (the type defines a redirection)
9      {
10          Modify the routing table
11     }
12     If (the type defines other error messages)
13     {
14          Inform the appropriate source protocol
15     }
16     Return
17  }
```

ICMP DESIGN

OUTPUT MODULE

- responsible for creating request, solicitation, or error messages requested by a higher level or the IP protocol.
- the module receives a demand from IP, UDP or TCP to send one of the ICMP error messages
- if the demand is from IP
  - check first that request is allowed
  - ICMP message cannot be created for four situations;
  - ICMP error message
  - Fragmented IP packet
  - Multicast IP packet
  - IP packet having IP address or 127.X.Y.Z
- May also receive a demand from an application program to send one of the ICMP request or solicitation messages

OUTPUT MODULE

Pseudocode for Output Module

Output Module

```
Receive: a demand
1. If (the demand defines an error message)
        1. If (the demand is from IP )
                1. If (the demand is forbidden)
                        1. Return.
        2. If (the type defines a redirection message)
                1. If (the station is not a router)
                        1. Return.
        3. Create the error message using the type, the code, and the IP packet.
2. If ( the demand defines a request or solicitation)
        1. Create a request or solicitation message.
3. Send the message.
4. Return.
```


UDP
User Datagram
Protocol

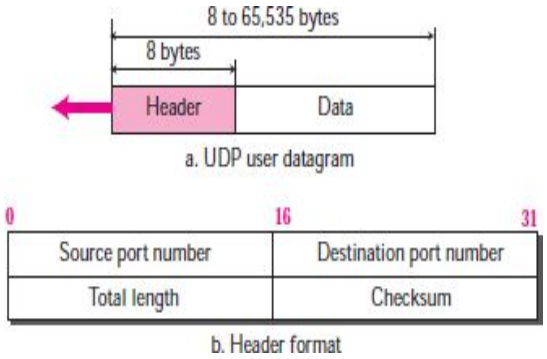
USER DATAGRAM PROTOCOL

UDP – An Introduction

- Connectionless service
- Unreliable transport protocol.
- No flow control / No Acknowledgement
- Process to Process communication
- Powerless
- It uses minimum of over heads
- No reliability is obtained using UDP
- Less interaction between sender and receiver

USER DATAGRAM PROTOCOL

USER DATAGRAM



User datagram header format

- Above picture depicts user data gram format
- UDP packets are called user datagrams(messages)
- It has fixed size header of 8 bytes

USER DATAGRAM PROTOCOL

UDP Datagram various fields

- Source Port Number:
  - 16 bits long – port ranges from 0-65535.
  - This port number will be used by the source host for identification.
- Destination Port Number:
  - 16 bits long.
  - Used by the process running on the Destination machine.
  - Application level service on end machine

USER DATAGRAM PROTOCOL

UDP Datagram various fields

- Length field s
- Length:

UDP length=IP length – IP headers Length

packet (including header).

- It is 16-bits field and minimum value is 8-byte, i.e. the size of UDP header
  - itself.
- A user datagram is encapsulated in an IP datagram.

- Checksum:
  - This field is used to detect errors over the entire user datagram (header

plus data)

USER DATAGRAM PROTOCOL

UDP Services

- Process to Process communication
- UDP provides process-to-process communication using sockets, a combination of IP addresses and port numbers. Several port numbers used by UDP.

Port	*Protocol*	*Description*
7	Echo	Echoes a received datagram back to the sender
9	Discard	Discards any datagram that is received
11	Users	Active users
13	Daytime	Returns the date and the time
17	Quote	Returns a quote of the day
19	Chargen	Returns a string of characters
53	Domain	Domain Name Service (DNS)
67	Bootps	Server port to download bootstrap information
68	Bootpc	Client port to download bootstrap information
69	TFTP	Trivial File Transfer Protocol
111	RPC	Remote Procedure Call
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management Protocol
162	SNMP	Simple Network Management Protocol (trap)

USER DATAGRAM PROTOCOL

UDP Services cont...

- Connectionless services:
  - Connectionless service.
  - Each datagram is independent even it comes from same source and delivered in same destination.
  - The datagrams are not numbered.
  - No connection establishments is done
  - cannot send a stream of data to UDP, It will chop them into differentrelated user datagrams.
- Flow control:
  - No Flow Control and no window mechanism.
  - The receiver may overflow with incoming messages.
  - UDP should provide for this service, if needed

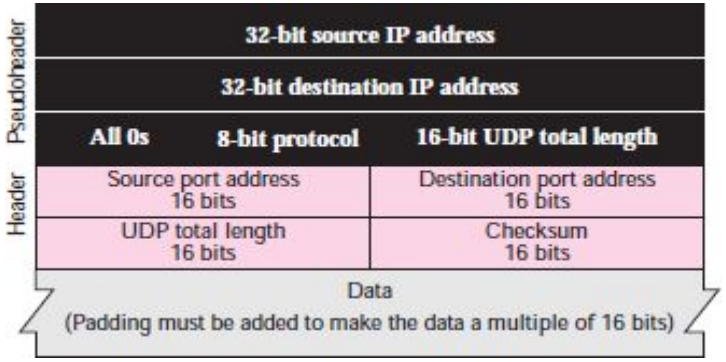
USER DATAGRAM PROTOCOL

UDP Services cont...

- Error Control:
  - No Error control mechanism except for checksum
  - Sender is unknown about the loss and duplication.
  - When error detects in receiver the datagram is discarded

Checksum

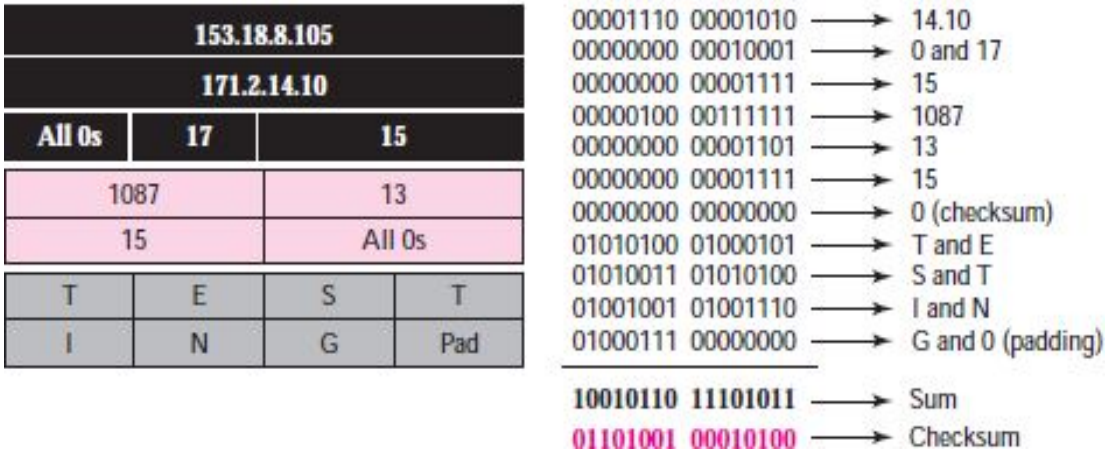
- Three sections: a pseudo header, the UDP header, and the data coming from the application layer.



Pseudoheader for checksum calculation

USER DATAGRAM PROTOCOL

UDP Services cont...



Check Sum Calculation

USER DATAGRAM PROTOCOL

UDP Services cont…

- Congestion Control
  - It is a connectionless protocol so

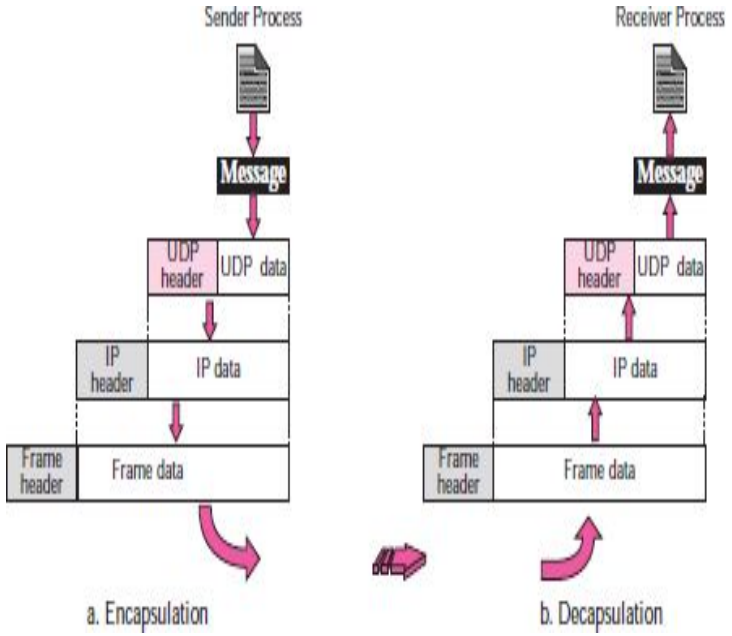
    congestion control not provided.
- Encapsulation and Decapsulation
  - Encapsulation
    - A process send message through UDP
    - Along pair of socket address and the length of data.

- UDP receives data and add UDP header then pass to IP with socket.
- IP add its own header using value 17
- in protocol filed.
- Indicating UDP protocol.
- The IP datagram is then passed to the data link layer.
- The data link layer receives the IP datagram and passes it to the physical layer.
- The physical layer encodes the

USER DATAGRAM PROTOCOL

UDP Services cont…

- ***Decapsulation:***
  - At the destination host:
    - the physical layer decodes the signals pass to data link layer.
    - The data link layer uses the header (and the trailer) to check the data.
  - If there is no error
    - the header and trailer are dropped , datagram is passed to IP.



Encapsulation and Decapsul a tion

USER DATAGRAM PROTOCOL

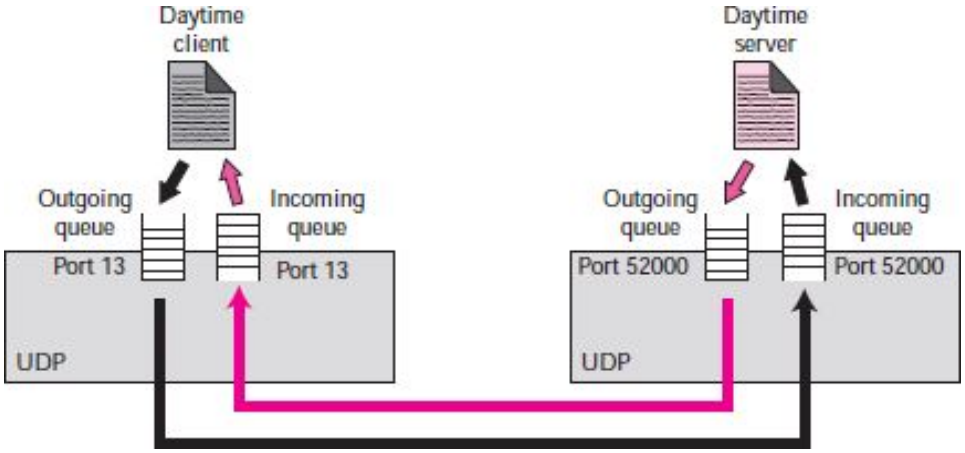
UDP Services cont...

- The header is dropped and the user datagram is passed to UDP with the sender and receiver IP addresses.
- Checksum is to check the entire user datagram.
- the header is dropped and the application data along with the sender socket address is passed to the process.
- The sender socket address is passed to the process in case it needs to respond to the message received.

USER DATAGRAM PROTOCOL

UDP Services cont…

Queuing in UDP



USER DATAGRAM PROTOCOL

UDP Services cont...

Queuing in UDP

- At the client site, when a process starts, it requests a port number from the operating system. Some implementations create both an incoming and an outgoing queue associated with each process. Other implementations create only an incoming queue associated with each process
- process wants to communicate with multiple processes, it obtains only one port number and eventually one outgoing and one incoming queue. The queues opened by the client are, in most cases, identified by ephemeral port numbers. The queues function as long as the process is running. When the process terminates, the queues are destroyed.
- The client process can send messages to the outgoing queue by using the source port number specified in the request

USER DATAGRAM PROTOCOL

UDP Services cont...

Queuing in UDP

- The client process can send messages to the outgoing queue by using the source port number specified in the request
- UDP removes the messages one by one and, after adding the UDP header, delivers them to IP. An outgoing queue can overflow
- It happens the operating system can ask the client process to wait before sending any more messages

USER DATAGRAM PROTOCOL

UDP Services cont...

Applications of UDP:

Used for simple request response communication when size of data is less hence there is lesser concern about flow and error control.

It is suitable protocol for multicasting as UDP supports packet switching. Following implementations uses UDP as a transport layer protocol:

- NTP (Network Time Protocol) DNS (Domain Name Service) BOOTP, DHCP.
- NNP (Network News Protocol) Quote of the day protocol TFTP, RTSP, RIP, OSPF.

UDP is null protocol if you remove checksum field.

USER DATAGRAM PROTOCOL

UDP Services cont...

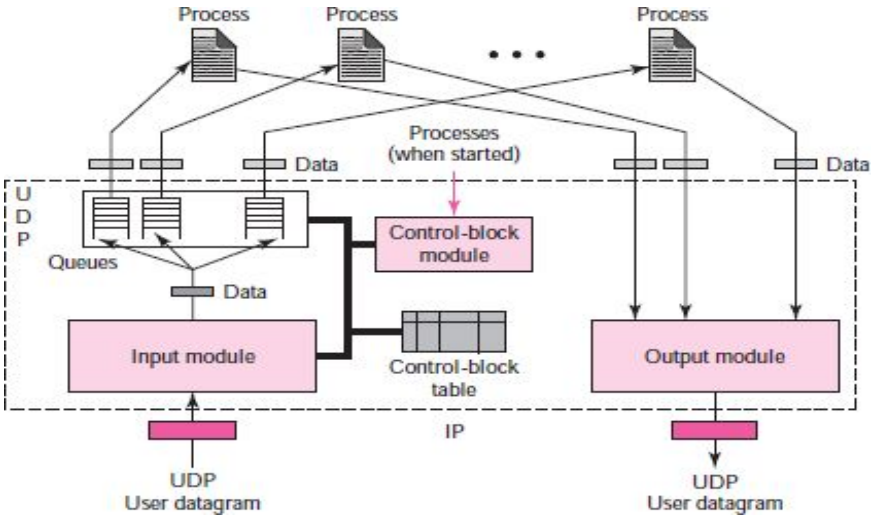
When to use UDP? Reduce the requirement of computer resources.

When using the Multicast or Broadcast to transfer.

The transmission of Real-time packets, mainly in multimedia applications

USER DATAGRAM PROTOCOL

UDP Package



UDP design

USER DATAGRAM PROTOCOL

UDP Package

- The UDP package involves five components:

Control-Block Table

- UDP has a control-block table to keep track of the open ports.
- Each entry in this table has a minimum of four fields: the state, which can be FREE or IN-USE, the process ID, the port number, and the corresponding queue number.

Input Queues

- Our UDP package uses a set of input queues, one for each process. In this design, we do not use output queues.

USER DATAGRAM PROTOCOL

UDP Package

- **Control-Block Module**
  - The control-block module is responsible for the management of the control-block table.
  - When a process starts, it asks for a port number from the operating system.
  - The operating system assigns well-known port numbers to servers and ephemeral port numbers to clients.
  - The process passes the process ID and the port number to the control-block module to create an entry in the table for the process.

```
UDP_Control_Block_Module (process ID, port number)
{
Search the table for a FREE
entry.  if (not found)
Delete one entry using a predefined
strategy.  Create a new entry with the state
IN-USE  Enter the process ID and the port
number.
 Return.
} // End module
```

USER DATAGRAM PROTOCOL

- UDP Package

  - Input Module

  - The input module receives a user datagram from the IP. It searches the control-block table to find an entry having the same port number as this user datagram.

  - If the entry is found, the module uses the information in the entry to enqueue the data.If the entry is not found, it generates an ICMP message..

```
UDP_INPUT_Module (user_datagram)
{
Look for the entry in the control_block table
If (found)
{
Check to see if a queue is allocated
If (queue is not allocated)
allocate a queue
else
enqueue the data
} //end if
else
{
Ask ICMP to send an "unreachable port" message
Discard the user datagram
} //end else
return
} // end module
```

USER DATAGRAM PROTOCOL

UDP Package

Output Module

- The output module is responsible for creating and sending user datagrams. UDP_OUTPUT_MODULE (Data)

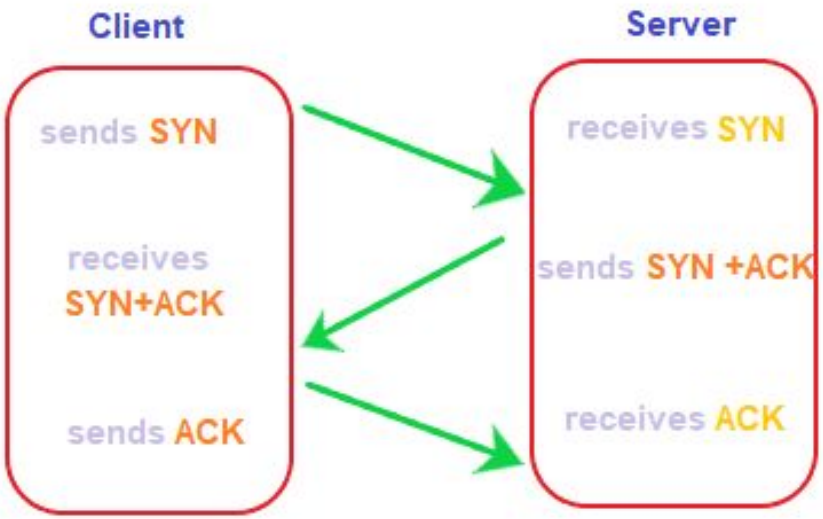
```
{
    Create a user datagram
    Send the user datagram
    Return.
}
```

UDP Features at a Glance

- UDP is connectionless
- UDP uses ports for data transmission
- UDP guarantees fast and delay-free communication
- UDP doesn’t care about data integrity or reliability
- UDP can be vulnerable to denial of service attacks (DDOS)

TCP Connection (A 3-way handshake)

- Handshake refers to the process to establish connection between the client and server. Handshake is simply defined as the process to establish a communication link.
- To transmit a packet, TCP needs a three way handshake before it starts sending data.
- The reliable communication in TCP is termed as par (positive acknowledgement re-transmission).
- When a sender sends the data to the receiver, it requires a positive acknowledgement from the receiver confirming the arrival of data.
- If the acknowledgement has not reached the sender, it needs to resend that data.
- The positive acknowledgement from the receiver establishes a successful connection.



- Here, the server is the server and client is the receiver.
- The above diagram shows 3 steps for successful connection.
- A 3-way handshake is commonly known as SYN-SYN-ACK and requires both the client and server response to exchange the data.
- SYN means **synchronize Sequence Number** and ACK means **acknowledgment**.
- Each step is a type of handshake between the sender and the receiver.

The three handshakes are discussed in the below steps:

Step 1: SYN

- SYN is a segment sent by the client to the server.
- It acts as a **connection request** between the client and server. It informs the server that the client wants to establish a connection.
- Synchronizing sequence numbers also helps synchronize sequence numbers sent between any two devices, where the same SYN segment asks for the sequence number with the connection request.

Step 2: SYN-ACK

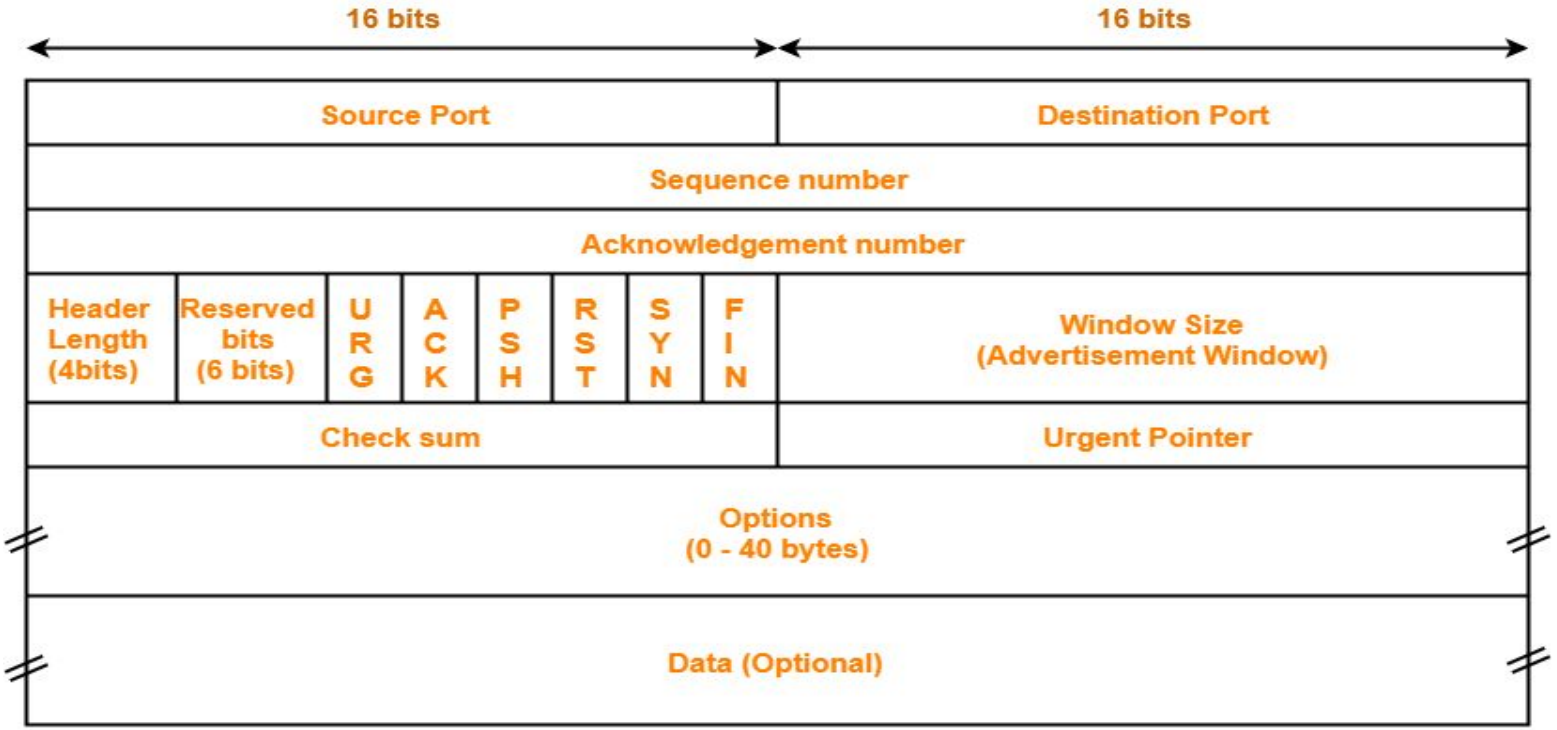
- It is an SYN-ACK segment or an SYN + ACK segment sent by the server.
- The ACK segment informs the client that the server has received the connection request and it is ready to build the connection.
- The SYN segment informs the sequence number with which the server is ready to start with the segments.

Step 3: ACK

- ACK (Acknowledgment) is the last step before establishing a successful TCP connection between the client and server.
- The ACK segment is sent by the client as the response of the received ACK and SN from the server. It results in the establishment of a reliable data connection.
- After these three steps, the client and server are ready for the data communication process

TCP Header

The following diagram represents the TCP header format-



TCP Header

1. Source Port-

- Source Port is a 16 bit field.
- It identifies the port of the sending application.

2. Destination Port-

- Destination Port is a 16 bit field.
- It identifies the port of the receiving application.

3. Sequence Number-

- Sequence number is a 32 bit field.
- TCP assigns a unique sequence number to each byte of data contained in the TCP segment.
- This field contains the sequence number of the first data byte.

Acknowledgement Number-

- Acknowledgment number is a 32 bit field.
- It contains sequence number of the data byte that receiver expects to receive next from the sender.
- It is always sequence number of the last received data byte incremented by 1.

Header Length-

- Header length is a 4 bit field.
- It contains the length of TCP header.
- It helps in knowing from where the actual data begins.

Reserved Bits-

- The 6 bits are reserved.
- These bits are not used.

URG Bit-

URG bit is used to treat certain data on an urgent basis.

- When URG bit is set to 1,
- It indicates the receiver that certain amount of data within the current segment is urgent.
- Urgent data is pointed out by evaluating the urgent pointer field.
- The urgent data has be prioritized.
- Receiver forwards urgent data to the receiving application on a separate channel.

ACK Bit-

ACK bit indicates whether acknowledgement number field is valid or not.

- When ACK bit is set to 1, it indicates that acknowledgement number contained in the TCP header is valid.
- For all TCP segments except request segment, ACK bit is set to 1.
- Request segment is sent for connection establishment during **Three Way Handshake**.

PSH Bit-

PSH bit is used to push the entire buffer immediately to the receiving application.

- When PSH bit is set to 1,
- All the segments in the buffer are immediately pushed to the receiving application.
- No wait is done for filling the entire buffer.
- This makes the entire buffer to free up immediately.

RST Bit- RST bit is used to reset the TCP connection.

- When RST bit is set to 1,
- It indicates the receiver to terminate the connection immediately.
- It causes both the sides to release the connection and all its resources abnormally.
- The transfer of data ceases in both the directions.
- It may result in the loss of data that is in transit.

This is used only when-

- There are unrecoverable errors.
- There is no chance of terminating the TCP connection normally.

SYN Bit- SYN bit is used to synchronize the sequence numbers.

- When SYN bit is set to 1,
- It indicates the receiver that the sequence number contained in the TCP header is the initial sequence number.
- Request segment sent for connection establishment during Three way handshake contains SYN bit set to 1.

FIN Bit- FIN bit is used to terminate the TCP connection.

- When FIN bit is set to 1,
- It indicates the receiver that the sender wants to terminate the connection.
- FIN segment sent for **TCP Connection Termination** contains FIN bit set to 1.

Window Size-

- Window size is a 16 bit field.
- It contains the size of the receiving window of the sender.
- It advertises how much data (in bytes) the sender can receive without acknowledgement.
- Thus, window size is used for **Flow Control**.

Checksum-

- Checksum is a 16 bit field used for error control.
- It verifies the integrity of data in the TCP payload.
- Sender adds CRC checksum to the checksum field before sending the data.
- Receiver rejects the data that fails the CRC check.

Urgent Pointer-

- Urgent pointer is a 16 bit field.
- It indicates how much data in the current segment counting from the first data byte is urgent.
- Urgent pointer added to the sequence number indicates the end of urgent data byte.
- This field is considered valid and evaluated only if the URG bit is set to 1.

Options-

- Options field is used for several purposes.
- The size of options field vary from 0 bytes to 40 bytes.

Options field is generally used for the following purposes-

- Time stamp
- Window size extension
- Parameter negotiation
- Padding

Time Stamp-

- When wrap around time is less than life time of a segment,
- Multiple segments having the same sequence number may appear at the receiver side.
- This makes it difficult for the receiver to identify the correct segment.
- If time stamp is used, it marks the age of TCP segments.
- Based on the time stamp, receiver can identify the correct segment.

Window Size Extension-

- Options field may be used to represent a window size greater than 16 bits.
- Using window size field of TCP header, window size of only 16 bits can be represented.
- If the receiver wants to receive more data, it can advertise its greater window size using this field.
- The extra bits are then appended in Options field.

Parameter Negotiation-

- Options field is used for parameters negotiation.
- Example- During connection establishment,
- Both sender and receiver have to specify their maximum segment size.
- To specify maximum segment size, there is no special field.
- So, they specify their maximum segment size using this field and negotiates.

Padding-

- Addition of dummy data to fill up unused space in the transmission unit and make it conform to the standard size is called as padding.
- Options field is used for padding.

Error Control in TCP

TCP protocol has methods for finding out corrupted segments, missing segments, out-of-order segments and duplicated segments.

Error control in TCP is mainly done through the use of **three simple techniques** :

- **Checksum –** Every segment contains a checksum field which is used to find corrupted segments. If the segment is corrupted, then that segment is discarded by the destination TCP and is considered lost.
- **Acknowledgement –** TCP has another mechanism called acknowledgement to affirm that the data segments have been delivered. Control segments that contain no data but have sequence numbers will be acknowledged as well but ACK segments are not acknowledged.

Retransmission

- When a segment is missing, delayed to deliver to a receiver, corrupted when it is checked by the receiver then that segment is retransmitted again.
- Segments are retransmitted only during two events: when the sender receives three duplicate acknowledgements (ACK) or when a retransmission timer expires.

Retransmission after RTO:

- TCP always preserves one retransmission time-out (RTO) timer for all sent but not acknowledged segments.
- When the timer runs out of time, the earliest segment is retransmitted. Here no timer is set for acknowledgement.
- In TCP, the RTO value is dynamic in nature and it is updated using the round trip time (RTT) of segments.
- RTT is the time duration needed for a segment to reach the receiver and an acknowledgement to be received by the sender.

Retransmission after Three duplicate ACK segments:

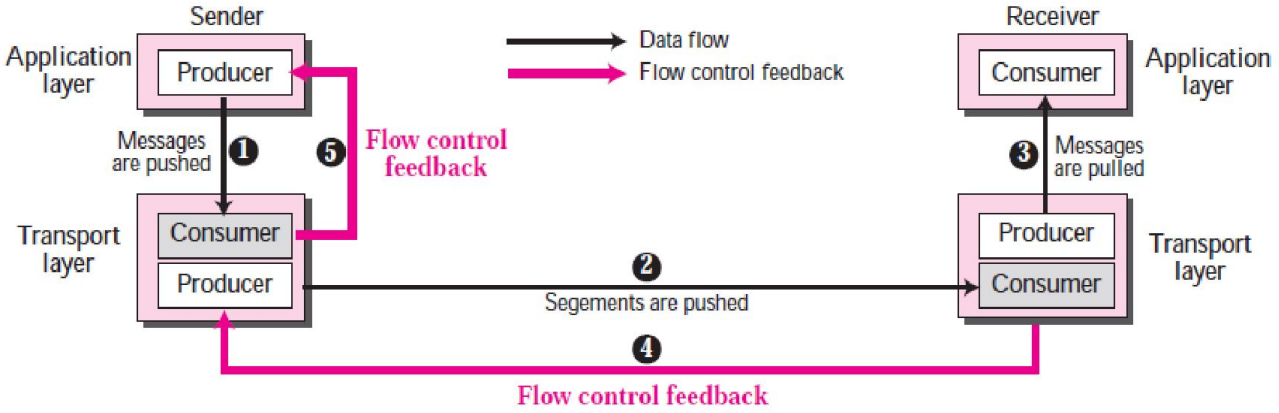
- RTO method works well when the value of RTO is small.
- If it is large, more time is needed to get confirmation about whether a segment has been delivered or not.
- Sometimes one segment is lost and the receiver receives so many out-of-order segments that they cannot be saved.
- In order to solve this situation, three duplicate acknowledgement method is used and missing segment is retransmitted immediately instead of retransmitting already delivered segment.
- This is a fast retransmission because it makes it possible to quickly retransmit lost segments instead of waiting for timer to end.

TCP Flow Control

Transmission Control Protocol (TCP)

TCP Flow Control

- Creates a balance between rate of data production and the rate of data consumption
- ***Assumption:*** Channel between sender & receiver is error-free



Data Flow and Flow Control Feedbacks in TCP

Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

1) Messages are pushed from the Sending application to TCP Client
2) Message segment from TCP Client is pushed to TCP Server
3) Messages are pulled by receiving application from TCP Server
4) Flow control feedback is sent from TCP server to TCP client
5) TCP client forwards the flow control feedback to sending application

- ***Opening and Closing Windows***
  - Buffer size of sender & receiver is fixed during connection establishment
  - Window sizes of Sender / Receiver is controlled and adjusted by TCP Server
  - Opening / Closing / Shrinking of client window is controlled by receiver

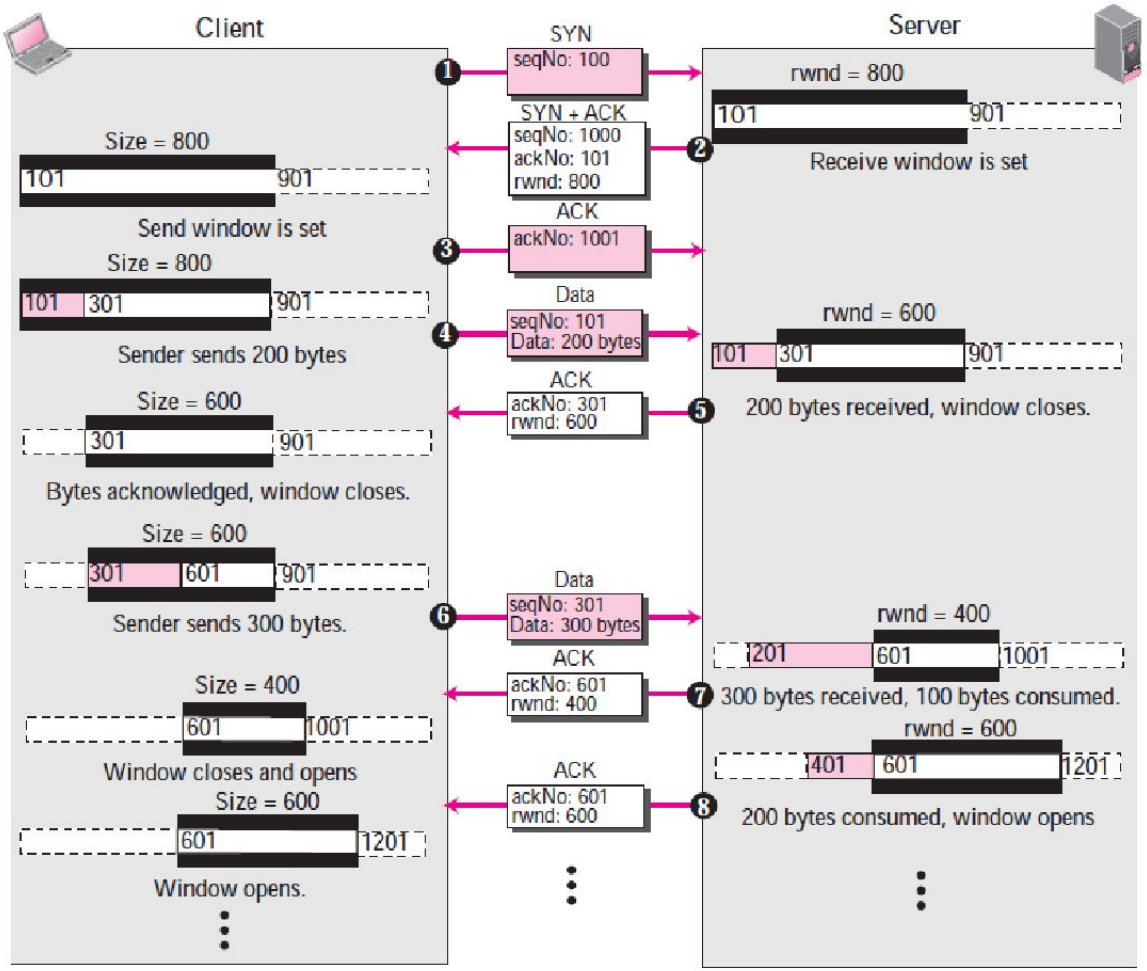
Transmission Control Protocol (TCP)

TCP Flow Control Contd…

Scenario – TCP Flow Control

- ***Segment 1***
  - Connection request – (SYN) segment from client to server
  - Initial Sequence Number (ISN): 100 (Next byte to Arrive : 101)
  - Server allots buffer size = 800 (Assumption)
  - Server allots Window size (rwnd) = 800
- ***Segment 2***
  - (ACK + SYN) segment from Server to Client
  - Set ACK no = 101 & Buffer Size = 800 bytes

Flow Control – A Scenario



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

Scenario – TCP Flow Control

- ***Segment 3***
  - ACK segment sent from client to server
- ***Segment 4***
  - Client sets window size to 800 (since rwnd from server is 800)
  - Client process pushes 200 bytes of data to TCP client
  - TCP client creates data segment with bytes (101-300) and sends to Server
  - Client window adjusted
    - Shows 200 bytes as sent but no ACK received from Server

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

Scenario – TCP Flow Control

- Shows 200 bytes as sent but no ACK received from Server
- Server stores 200 bytes in buffer & closes receive window
- Server indicates the next expected byte as 301

- ***Segment 5***
  - Server acknowledges receipt of 200 bytes from client & reduces rwnd to 600
  - Client receives acknowledgement and resizes window size to 600
  - Client closes the window (101-300) from left to right
  - Client indicates the next byte to send as 301

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

Scenario – TCP Flow Control

- *Segment 6*
  - Client pushes 300 more bytes to the server (Seq. No = 301 & Data = 300 bytes)
  - Server stores 300 bytes in buffer
  - 100 bytes of data are pulled by the Client process
  - So, Window size is reduced by 100 bytes to the left & opened by 100 bytes to the right
  - Overall, TCP client window size is reduced by 200 bytes
  - Now, Receiver window size (rwnd) = 400

Transmission Control Protocol (TCP)

TCP Flow Control Contd...

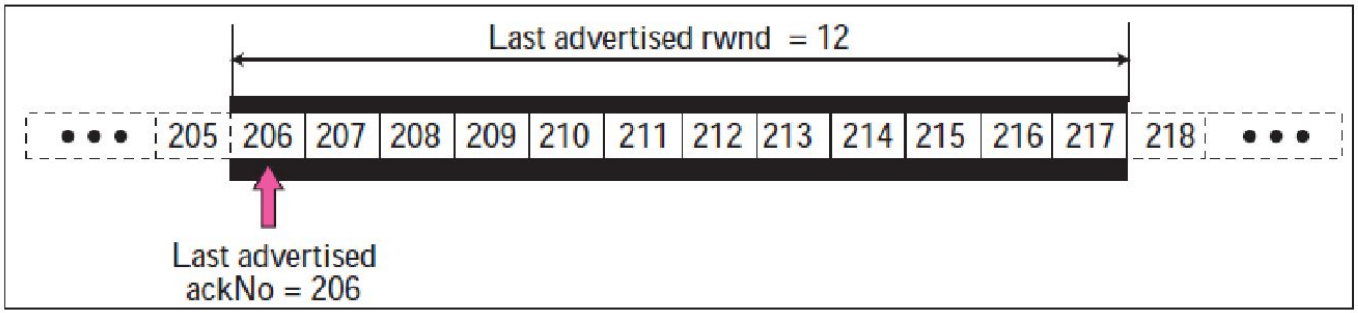
Scenario – TCP Flow Control

- *Segment 7*
  - TCP Server acknowledges receipt of 300 bytes and sets window size (rwnd) = 400 & TCP Client reduces window size to 400
  - Sender windows closes by 300 bytes from the left and opens by 100 bytes to the right
- This process continues until all the data segments are sent from server to client and connection gets closed

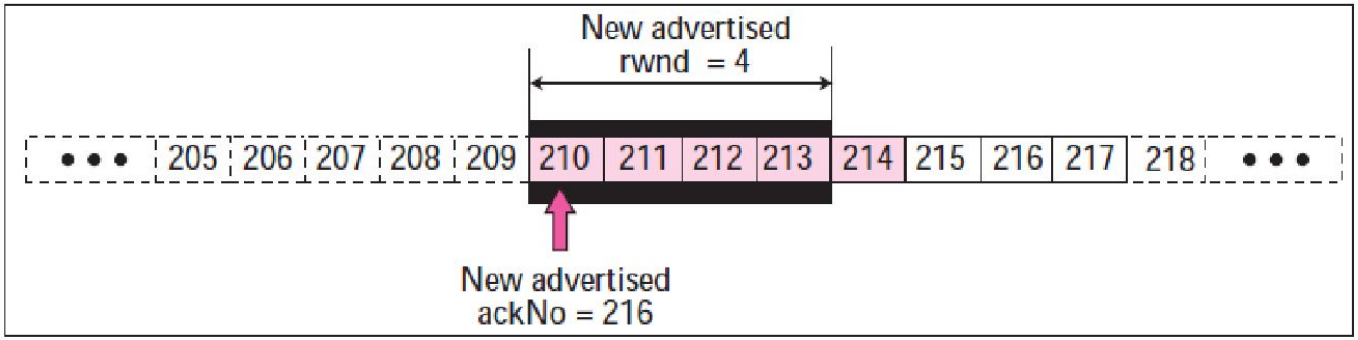
Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Shrinking of Windows***
  - Shrinking : Decreasing window size i.e., right wall moves towards the left
  - Sender window may shrink based on rwnd value defined by receiver
  - Receiver window cannot shrink
  - ***Illustration***
    - Upto 205 bytes of data received and acknowledged by sender
    - Last advertised rwnd = 12 (Window size) & last advertised ACK No = 206
      - Data can be sent from byte 206 to byte 217 (Since rwnd = 12)
    - New advertised rwnd = 12 (Window size) & new advertised ACK No = 210



a. The window after the last advertisement



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Shrinking of Windows Contd…***
  - Shrinking of window has occurred from byte 217 to byte 213 (Window had moved from right to left
  - Shrinking can be prevented by the relation given below:

New ACK number + new rwnd > = Lask ACK Number + Last rwnd

  - To prevent shrinking,
    - Wait until enough buffer locations are available in its window

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Silly Window Syndrome***
  - Occurs when either the sending process creates data slowly (or) the receiving process consumes data slowly (or) both
  - Silly window syndrome sends / receives data in small segments thus resulting in poor efficiency
  - ***Example***
    - A 42 byte TCP datagram is needed to send Segment with 2 bytes of data
    - Overhead : 42 / 2 ⇒ Network capacity used inefficiently

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Silly Window Syndrome Created by Sender***
  - Sending TCP creates syndrome since sending application program creates data slowly i.e., 1 byte at an instance
  - ***Suggestions***
    - Prevent sending TCP from transmitting data byte by byte
    - Sending TCP made to wait & collect data to send data in larger block
    - ***Disadvantage:*** Waiting too long delays the process
  - ***Solution***
    - Nagle's Algorithm

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Nagle's Algorithm for Silly Window Syndrome Created by Sender***
  i. First segment of data sent by sending TCP irrespective of the size of segment
  ii. Data is accumulated in buffer by sending TCP until acknowledgment is received from receiving TCP (or) enough data accumulated to send a segment
  iii. Repeat step 2 until transmission completes
- Nagle's algorithm works based on speed of application program and the network speed
  - Faster the application program larger the segment size
- ***Advantage:*** Simple to implement

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Silly Window Syndrome Created by Receiver***
  - Syndrome created when serving application consumes slowly
  - ***Example***
    - 1 KB data blocks created by sending application
    - 1 byte of consumed at a time by receiving application
    - Once sending window buffer is full, window size (rwnd) becomes 0
  - ***Solution 1:*** Clarks's Solution
    - Send ACK as data segment arrives
    - Set rwnd = 0 iff receive buffer is half empty (or) accommodate max size of segment

Transmission Control Protocol (TCP)

TCP Flow Control Contd…

- ***Silly Window Syndrome Created by Receiver Contd…***
  - ***Solution 2:*** Delayed Acknowledgment
    - Acknowledgement is withheld when segment arrives
    - Receiver waits for space in incoming buffer before acknowledging
    - Delayed ACK prevents sending TCP from sliding
    - Delayed ACK reduces network traffic
    - Note: ACK not to be delayed by more than 500 ms

TCP Error Control

Transmission Control Protocol (TCP)

TCP Error Control

- TCP – Reliable Transport layer Protocol
- Entire data stream to be delivered without error / loss / duplication
- Reliability is provided by TCP using Error Control
- Error Control includes:
  - Finding and resending corrupted segments
  - Resending lost segments
  - Storing out-of-order segments till missed segments arrive
  - Discarding duplicate segments
- Error control achieved by: Checksum, Acknowledgement and Time-out

Transmission Control Protocol (TCP)

TCP Error Control Contd…

a) Checksum

- TCP uses 16-bit mandatory checksum field
- Checksum field associated with each segment
  - Checks for corrupted segment
- Invalid Checksum: Segment discarded by receiving TCP & considered lost

Transmission Control Protocol (TCP)

TCP Error Control Contd…

b) Acknowledgement

- Acknowledgement segments (ACK) carry no data & confirms data segment receipt
- Types : Cumulative and Selective Acknowledgment
- ***Cumulative Acknowledgment (ACK)***
  - 32-bit ACK field used
  - Acknowledges segments cumulatively (sets ACK flag to 1)
  - No feedback provided for discarded, lost or duplicate segments
- ***Selective Acknowledgement (SACK)***
  - Reports out of order & duplicate segments

Transmission Control Protocol (TCP)

TCP Error Control Contd…

b) Acknowledgement Contd…

- No provision for SACK in TCP header
- SACK included as part of options field in the TCP header

- ***Rules for Generating Acknowledgments***

1) When data is sent from sender to receiver, ACK provides the next Seq. No expected to be received
   - This results in less traffic and less segments between sender and receiver
2) In case of one in-order segment remaining, receiver needs to delay sending ACK segment. Network traffic is thus reduced.

Transmission Control Protocol (TCP)

TCP Error Control Contd…

b) Acknowledgement Contd…

3) At no point of time there should be more than two in-order segments unacknowledged. (avoid unnecessary retransmission)
4) Receiver acknowledges (ACK) an higher out-of-order sequence number immediately leading to fast retransmission of next segment
5) Receiver sends ACK when a missing segment arrives. Segments reported as missing are thus informed to the receiver
6) Receiver discards duplicate segment & sends ACK indicating the next in-order segment. Lost ACK segment problems are thus solved.

Transmission Control Protocol (TCP)

TCP Error Control Contd…

c) Retransmission of Segments

- When retransmission occurs?
  - Expiry of retransmission timer (or)
  - Sender receives more than 2 duplicate ACK's for 1st segment
- ***Retransmission after RTO***
  - One retransmission time-out (RTO) maintained by sending TCP for each connection
  - In case of time-out, timer is restarted by TCP & first segment of Queue is sent
  - This version of TCP is called ***Tahoe***

Transmission Control Protocol (TCP)

TCP Error Control Contd…

c) Retransmission of Segments Contd…

- ***Retransmission after 3 Duplicate Segments***
  - Also called as Fast Retransmission & followed by most implementations
  - TCP version called as ***Reno***
  - If 3 identical duplicate ACK's along with the original ACK are received for a segment, the next segment is retransmitted
  - Note: Retransmission does not wait for time-out in this case

Transmission Control Protocol (TCP)

TCP Error Control Contd…

d) Out-of-Order Segments

- Out-of-Order segments are not discarded by TCP
- TCP flags such segments as out-of-order and store them temporarily until missing segments arrive
- TCP makes sure that data segments are delivered in sequence to the process

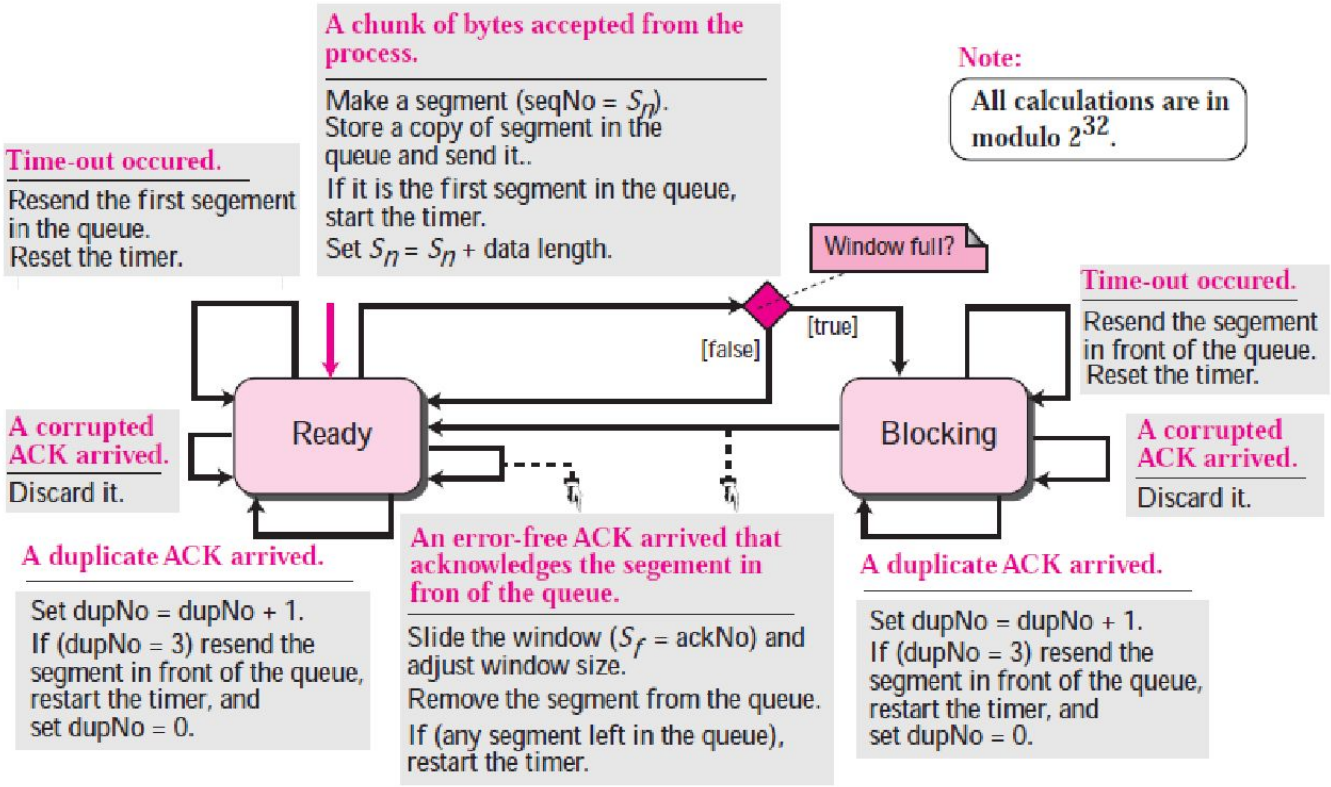
Transmission Control Protocol (TCP)

TCP Error Control Contd…

e) FSM for Data Transfer in TCP

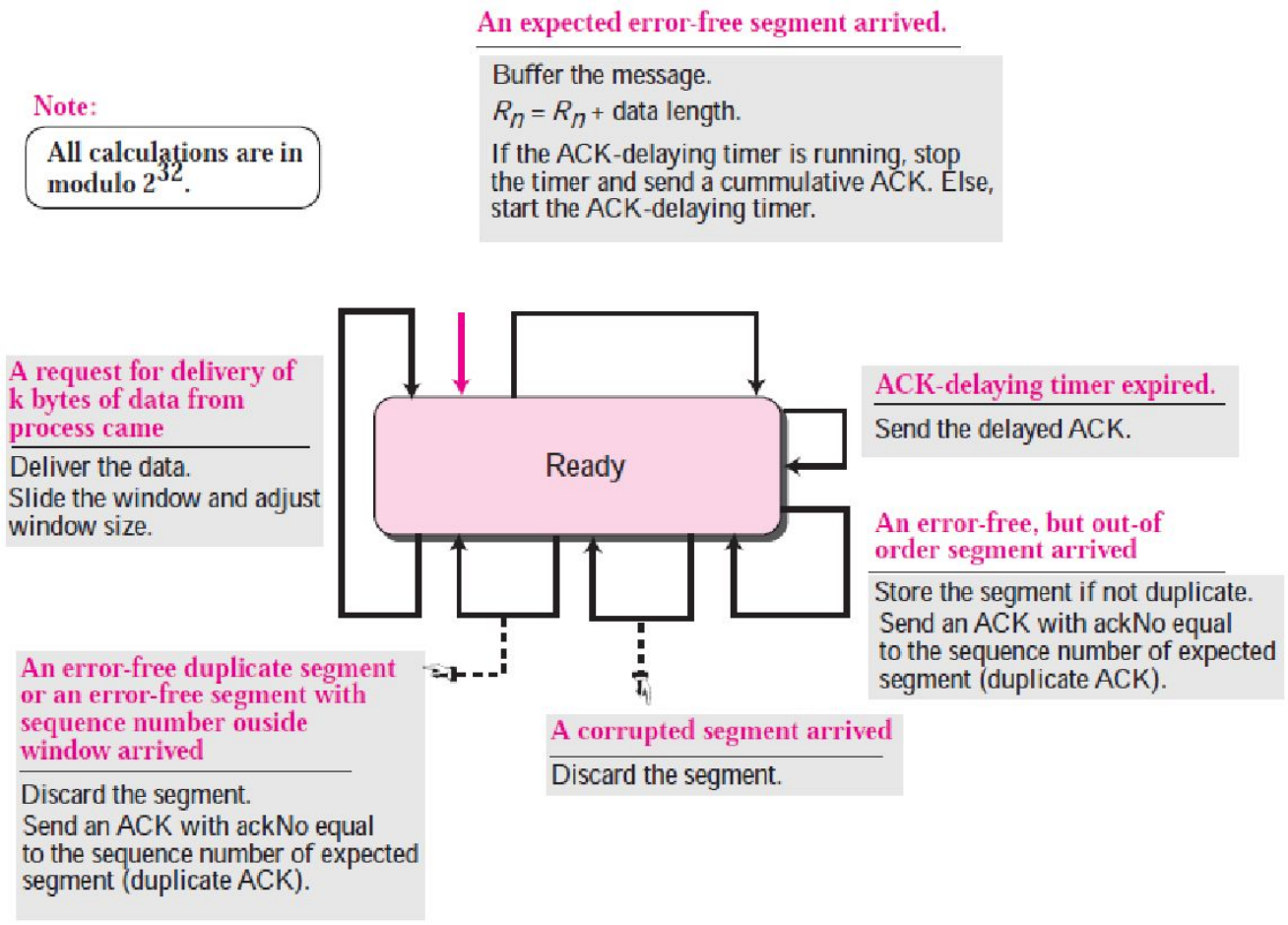
- FSM – Finite State Machine
- Similar to Selective repeat and Go Back-N protocol
- ***Sender-side & Receiver Side FSM***
  - ***Assumption :*** Unidirectional communication
  - ***Ignored Parameters:*** Selective ACK and Congestion Control
  - Nagle's algorithm / Windows shutdown not included in FSM
  - ***Advantage:*** Fast transmission policy using 3 duplicate ACK segments
  - ***Bi-directional FSM :*** Complex and more practical

Simplified FSM for TCP Sender Side



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Simplified FSM for TCP Receiver Side



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

TCP Congestion Control

Transmission Control Protocol (TCP)

TCP Congestion Control

- Congestion window and congestion policy handles TCP congestion

a) Congestion Window

- Client window size (rwnd) decided by the available buffer space of Server
- Ignored entity in deciding window size : Network Congestion
- Sender window size determined by,
  - rwnd (receiver advertised window size) &
  - cwnd (Congestion window size)

Actual window size = min (rwnd, cwnd)

Transmission Control Protocol (TCP)

TCP Congestion Control

Contd...

TERMINOLOGIES & ABBREVIATIONS USED

- ***rwnd –*** Sender Window Size
- ***cwnd –*** Congestion Window Size
- ***ssthresh –*** Slow Start Threshold
- ***MSS –*** Maximum Segment Size
- ***ACK –*** Acknowledgement
- ***RTT –*** Round Trip Time
- ***RTO –*** Retransmission Time-out

Transmission Control Protocol (TCP)

TCP Congestion Control

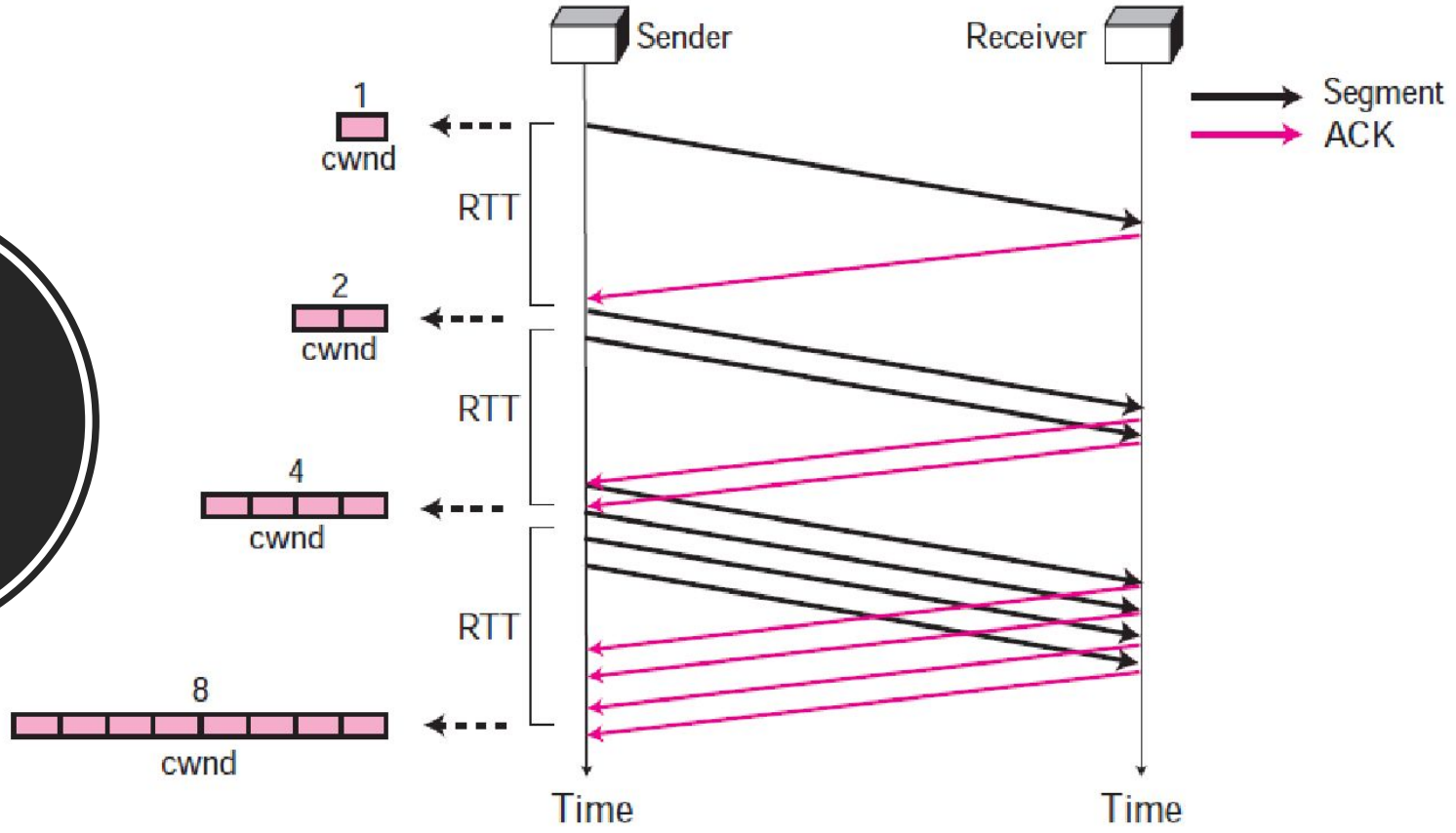
b) Cont...Congestion Policy

- Three phases : Slow Start, Congestion avoidance & Congestion detection

i. ***Slow Start (Exponential Increase)***

- ***Assumption:*** rwnd(Sender Window Size) > cwnd(Congestion Window Size)
- cwnd initialized to one Maximum window size (MSS)
- On arrival of each ACK, cwnd increases by 1
- Algorithm starts slowly & grows exponentially
- Delayed ACK policy is ignored
- ***Note:*** Consider each segment is individually acknowledged

Slow Start, Exponential Increase



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Congestion Control

b) Congestion Policy

Contd...

i. Slow Start (Exponential Increase) contd...

- Initial value of cwnd = 1 MSS

No of MSS sent	No of Segments Acknowledged	RTT	cwnd in MSS
Nil	Nil	-	1
1	1	1	$1 \times 2 = 2 \Rightarrow 2^1$
2	2	2	$2 \times 2 = 4 \Rightarrow 2^2$
4	4	3	$4 \times 2 = 8 \Rightarrow 2^3$
8	8	4	$8 \times 2 = 16 \Rightarrow 2^4$

Transmission Control Protocol (TCP)

TCP Congestion Control

b) Congestion Policy Cont'd...

i. Slow Start (Exponential Increase) contd...

- For Delayed Acknowledgements
  - If multiple segments are acknowledged accumulatively, cwnd increases by 1
  - ***Example:*** if ACK = 4 then cwnd = 5
  - Growth is exponential but not to the power of 2
  - Slow start stops with a threshold value ***ssthresh***
    - It stops when ***window size = = ssthresh***

Transmission Control Protocol (TCP)

TCP Congestion Control

b) Congestion Policy

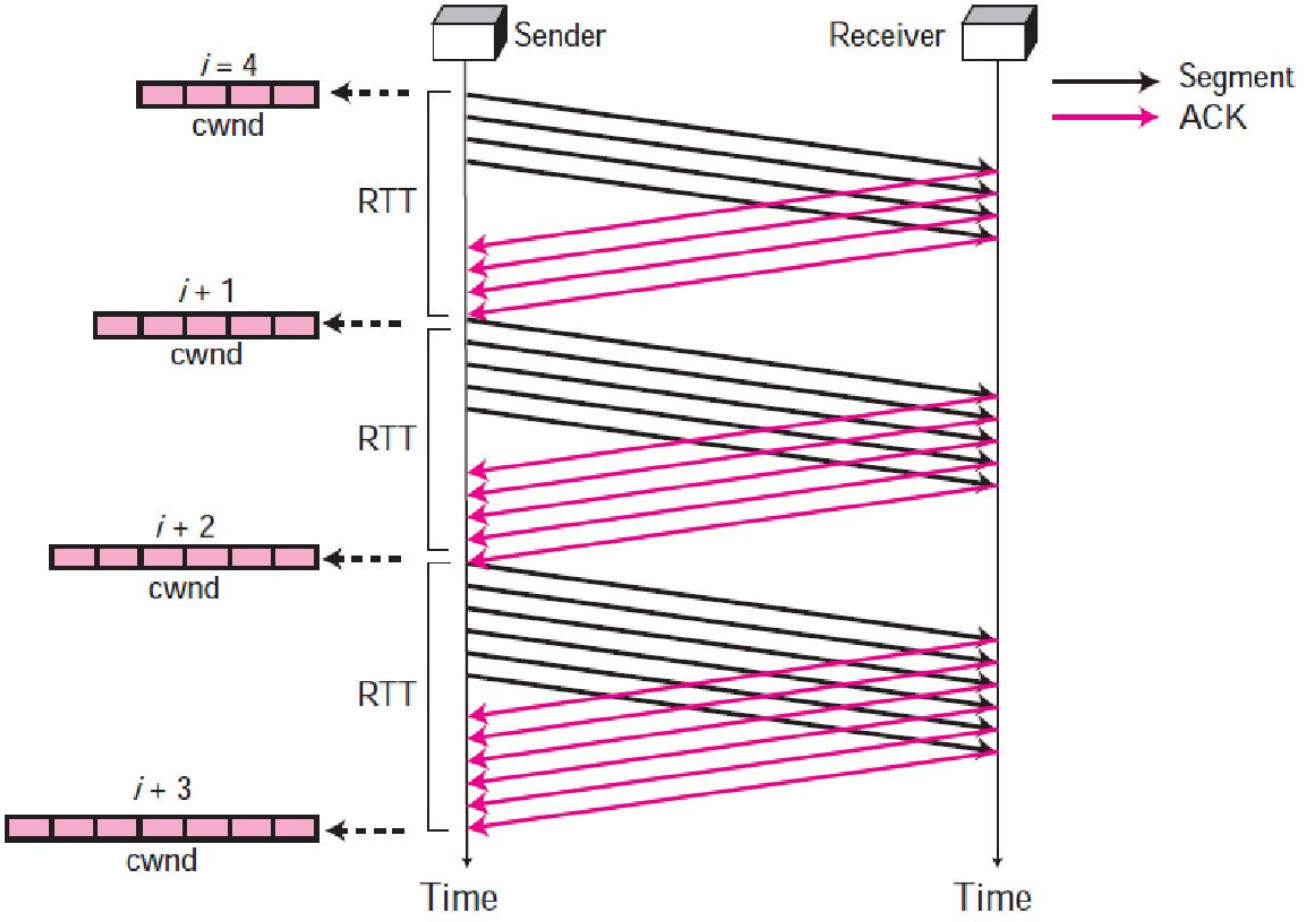
Contd..

ii. Congestion Avoidance : Additive Increase

- Slow start increases congestion window size (cwnd) exponentially
- Congestion avoidance increases cwnd additively
- Additive phase begins when slow start reaches ssthresh i.e. cwnd = I
- Increase in cwnd is based on RTT & not on number of ACK's

RTT	cwnd in MSS
1	i
2	i + 1

Congestive avoidance, Additive Increase



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Congestion Control

b) Cont...Congestion Policy

iii. Congestion Detection : Multiplicative Decrease

- Size of Cwnd must be decreased in case of congestion
- Retransmission occurs during missing segments / lost segments
- Retransmission helps identify whether congestion has occurred or not
- Retransmission occurs when
  - There is RTO Time-out
  - On receipt of three duplicate ACK's
- Note: In both cases, ssthresh is reduced by half (Multiplicative decrease)

Transmission Control Protocol (TCP)

TCP Congestion Control

b) Congestion Policy Contd...

iii. Congestion Detection : Multiplicative Decrease Contd...

a) Time-out increases possibility of congestion. TCP reacts as follows:

- Ssthresh set to half the value of rwnd
- Cwnd initialized to 1
- Slow start phase is initiated again

b) Three duplicate ACK's indicates a weaker possibility of Congestion. Also called as fast transmission & fast recovery. TCP reacts as follows:

- Ssthresh set to half the value of rwnd

Transmission Control Protocol (TCP)

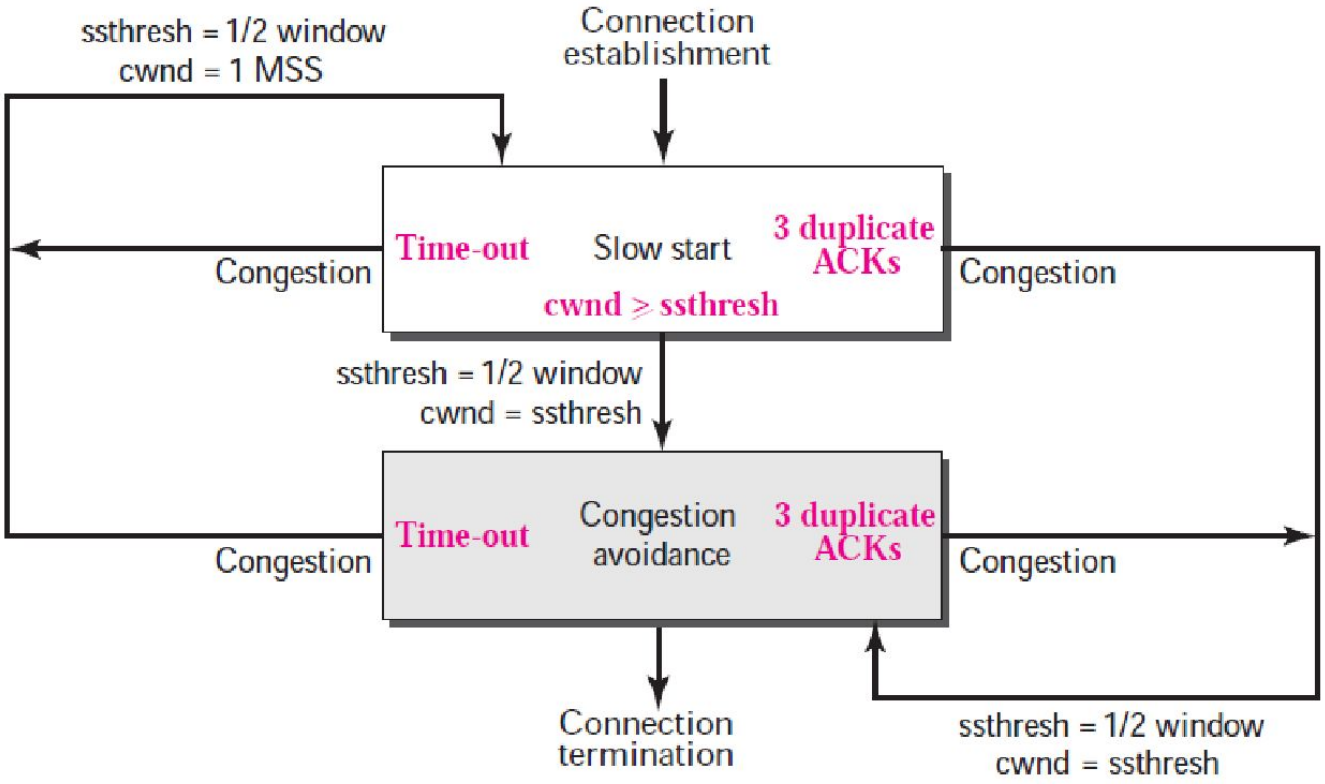
TCP Congestion Control

b) Congestion Policy Contd...

iii. Congestion Detection : Multiplicative Decrease Contd...

- Cwnd = ssthresh
- Congestion avoidance phase is initiated again

TCP Congestion Policy : A Summary



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

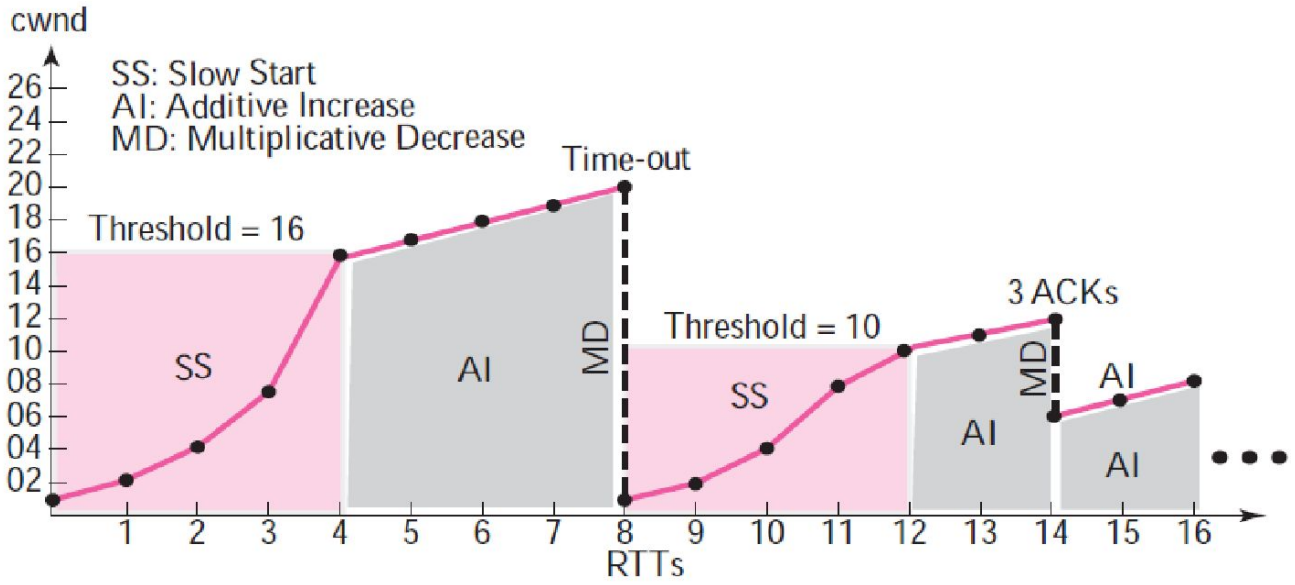
Transmission Control Protocol (TCP)

TCP Congestion Control

Contd... ***Summarization with Example***

- Assumptions
  - Maximum window Size (MSS) = 32
  - Threshold (ssthresh) = 16
- TCP moves to slow start
  - rwnd starts from 1 and and grows exponentially till it reaches ssthresh (16)
- Additive increase increases rwnd from 16 to 20 (one by one)
  - When rwnd = 20, time-out occurs
- Multiplicative Decrease: ssthresh reduced to 10 (half the window size)

Congestion Example



Source: Behrouz A. Forouzan, "TCP IP Protocol Suite " 4th edition, 2010, McGraw-Hill ISBN: 0073376043

Transmission Control Protocol (TCP)

TCP Congestion Control Contd...

Summarization with Example Contd...

- New ssthresh = 10
- TCP moves to Slow start again
  - rwnd starts from 1 and and grows exponentially till it reaches new ssthresh (10)
- Additive increase increases rwnd from 10 to 12 (one by one)
- 2 duplicate ACK's are received by the sender
- Multiplicative Decrease: ssthresh reduced to 6 (half the window size)

Multicasting and Multicast Routing Protocols

Objectives

Upon completion you will be able to:

- ***Differentiate between a unicast, multicast, and broadcast message***
- ***Know the many applications of multicasting***
- ***Understand multicast link state routing and MOSPF***
- ***Understand multicast link state routing and DVMRP***
- ***Understand the Core-Based Tree Protocol***
- ***Understand the Protocol Independent Multicast Protocols***
- ***Understand the MBONE concept***

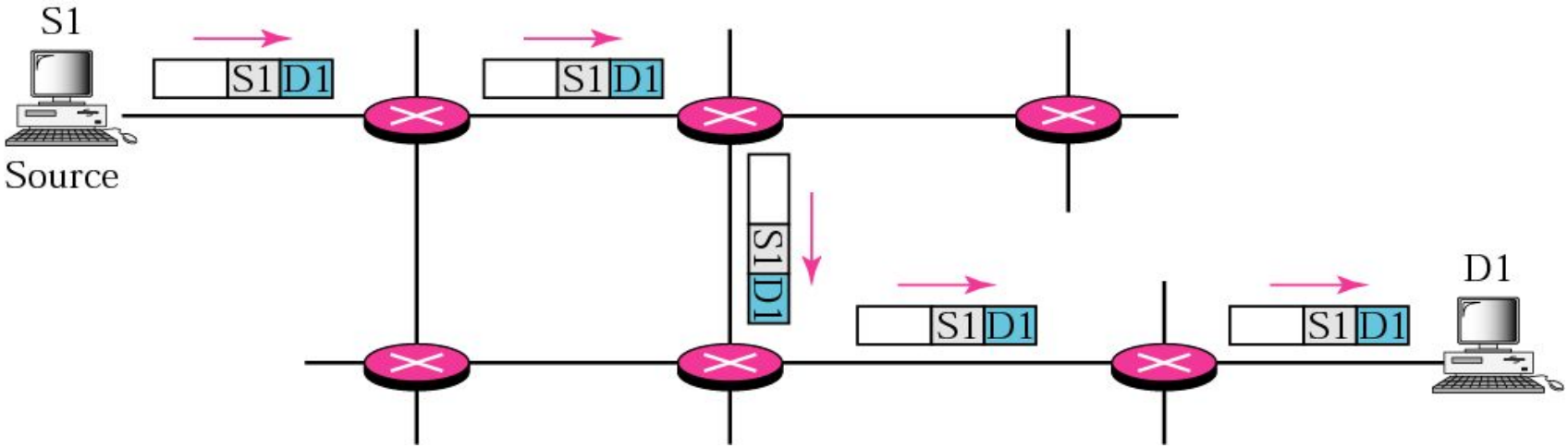
UNICAST, MULTICAST, AND BROADCAST

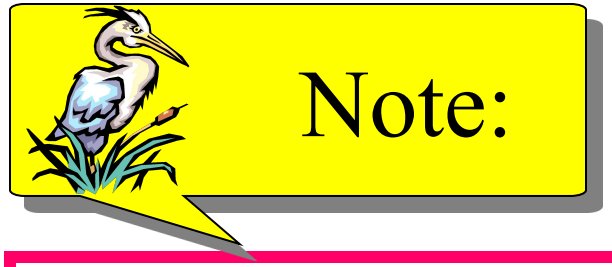
A message can be unicast, multicast, or broadcast. Let us clarify these terms as they relate to the Internet.

The topics discussed in this section include:

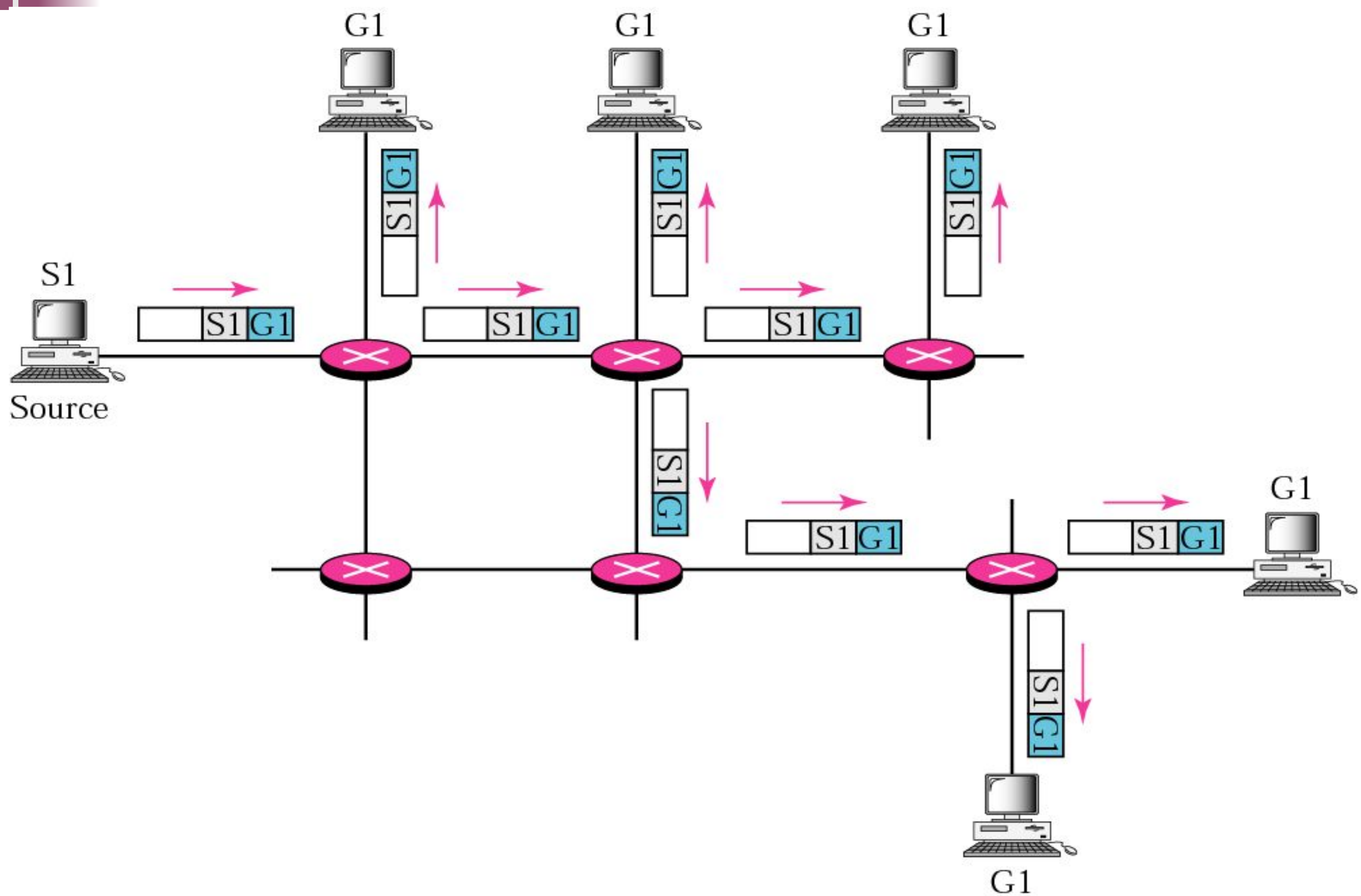
Unicasting
Multicasting
Broadcasting
Multicasting versus Multiple Unicasting

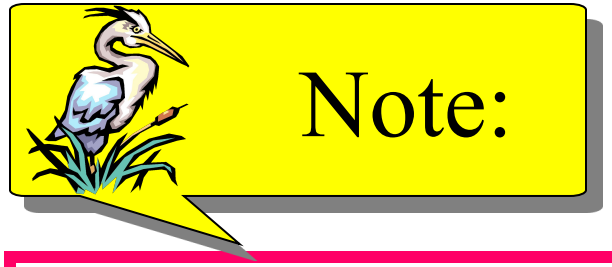
Unicasting





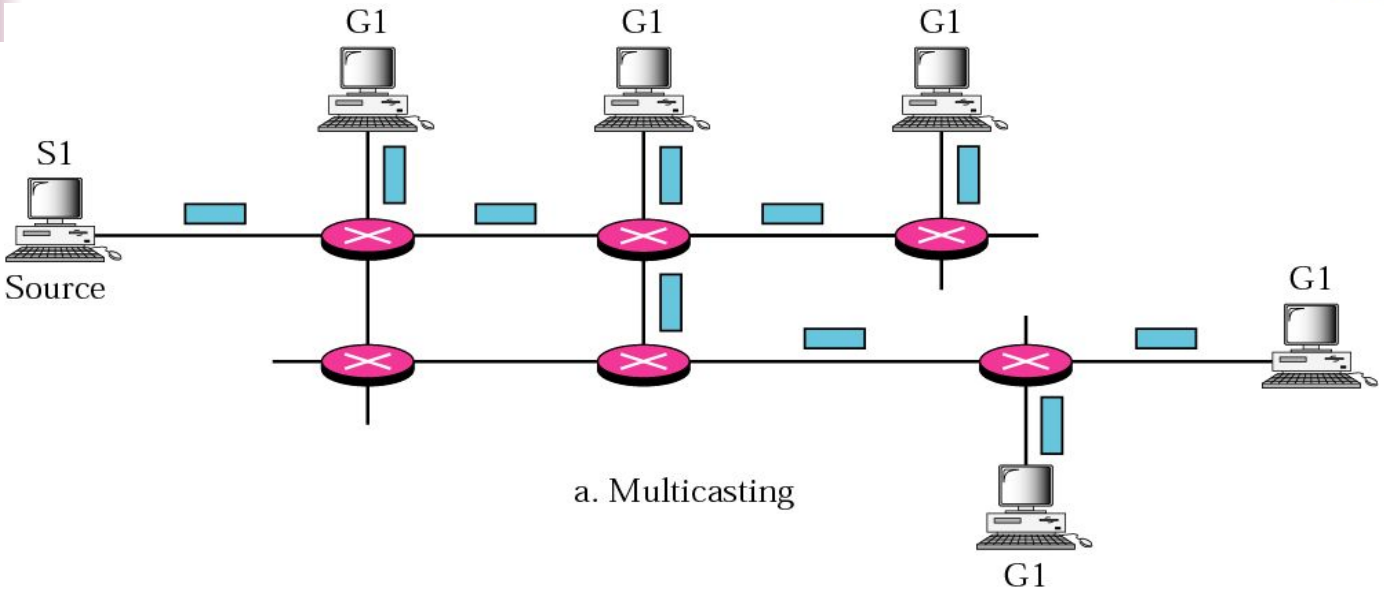
In unicasting, the router forwards the received packet through only one of its interfaces.


G1
G1
G1
S1
Source
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
S1 G1
G1
G1

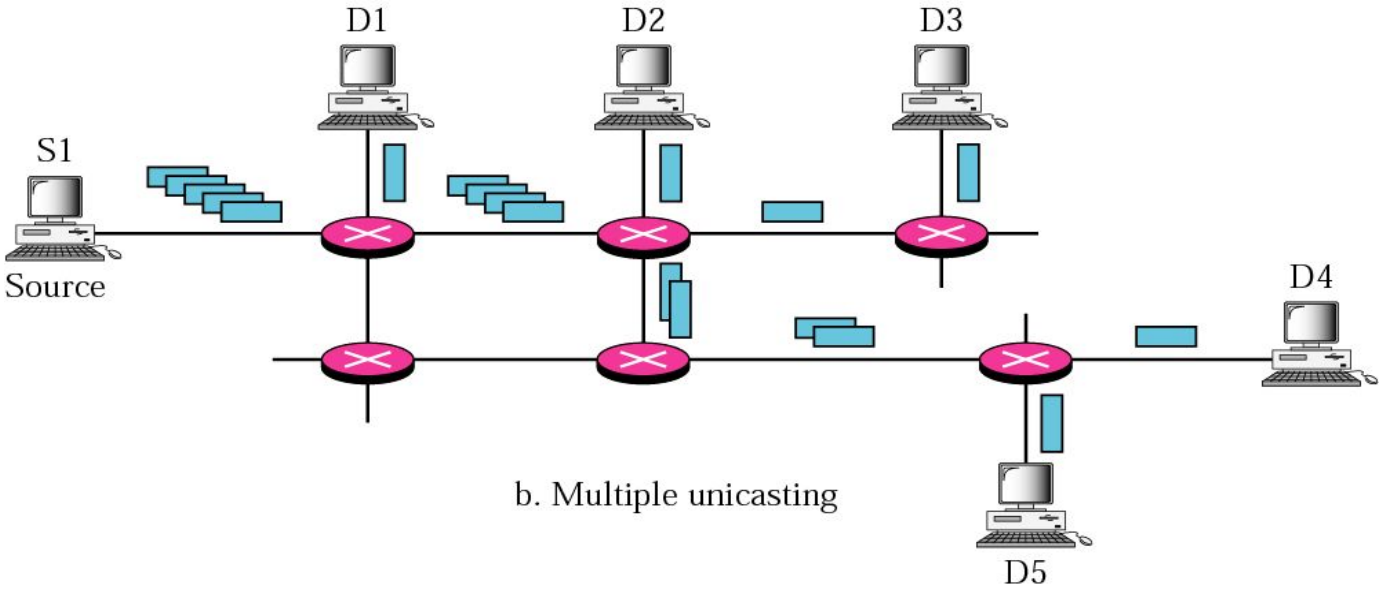


In multicasting, the router may forward the received packet through several of its interfaces.

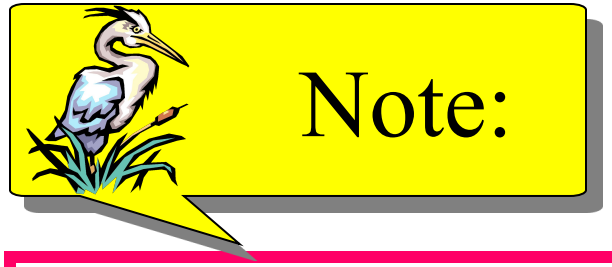
Multicasting versus multiple unicasting



a. Multicasting



b. Multiple unicasting



Emulation of multicasting through multiple unicasting is not efficient and may create long delays, particularly with a large group.

MULTICAST APPLICATIONS

Multicasting has many applications today such as access to distributed databases, information dissemination, teleconferencing, and distance learning.

The topics discussed in this section include:

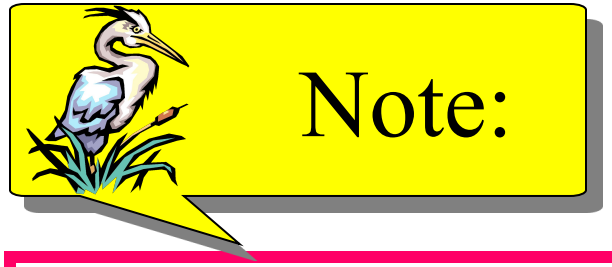
Access to Distributed Databases
Information Dissemination
Dissemination of News
Teleconferencing
Distance Learning

MULTICAST ROUTING

In this section, we first discuss the idea of optimal routing, common in all multicast protocols. We then give an overview of multicast routing protocols.

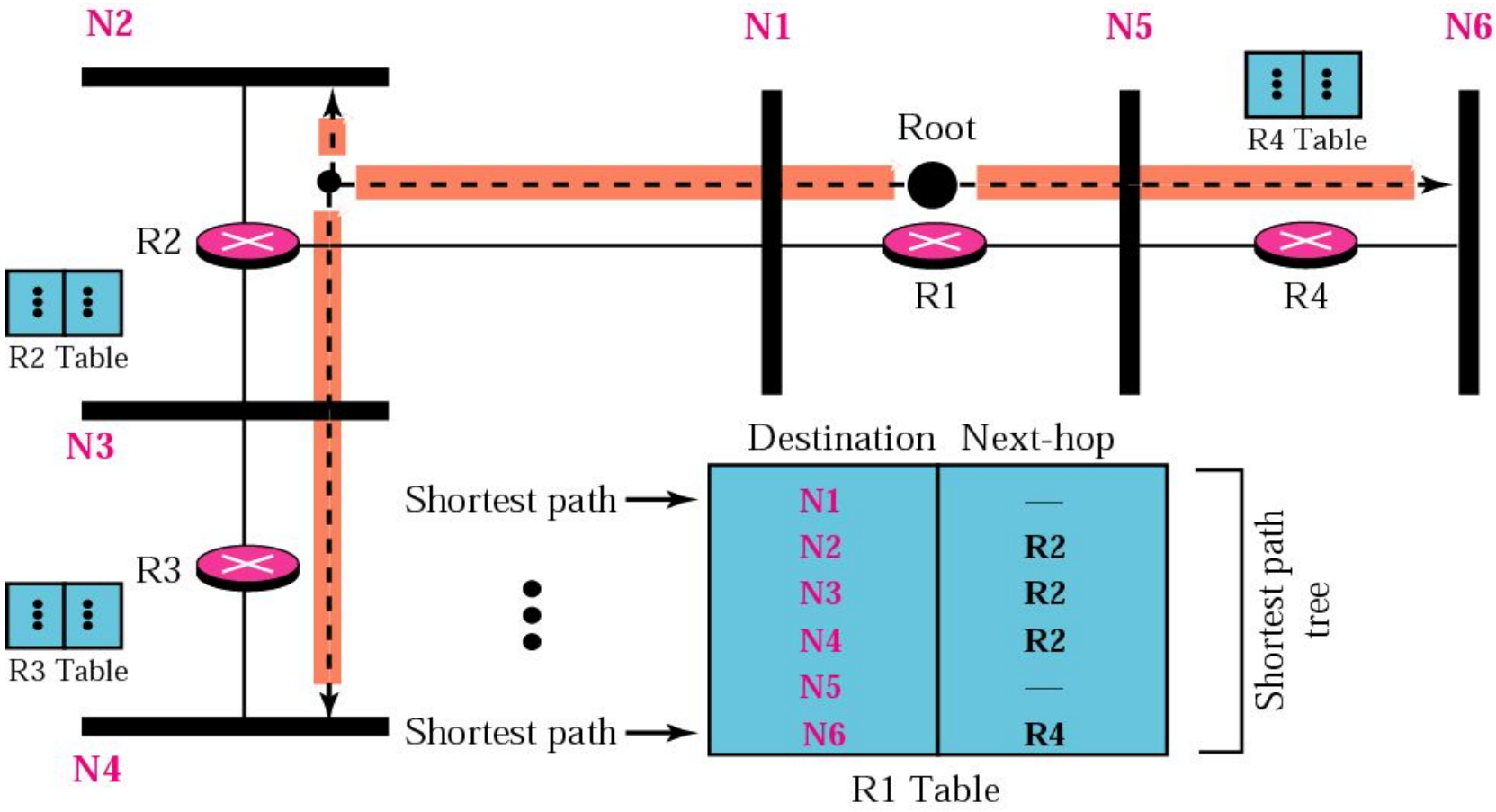
The topics discussed in this section include:

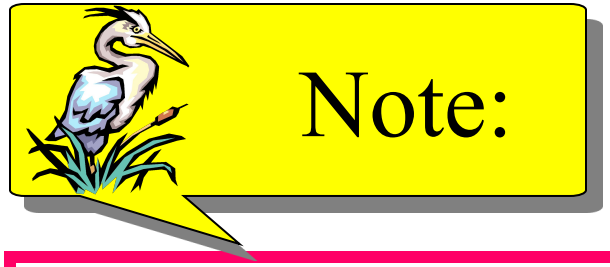
Optimal Routing: Shortest Path Trees
Routing Protocols



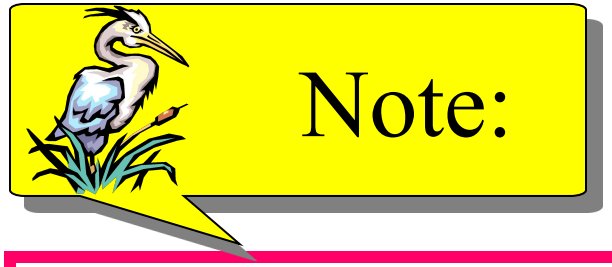
In unicast routing, each router in the domain has a table that defines a shortest path tree to possible destinations.

Shortest path tree in unicast routing



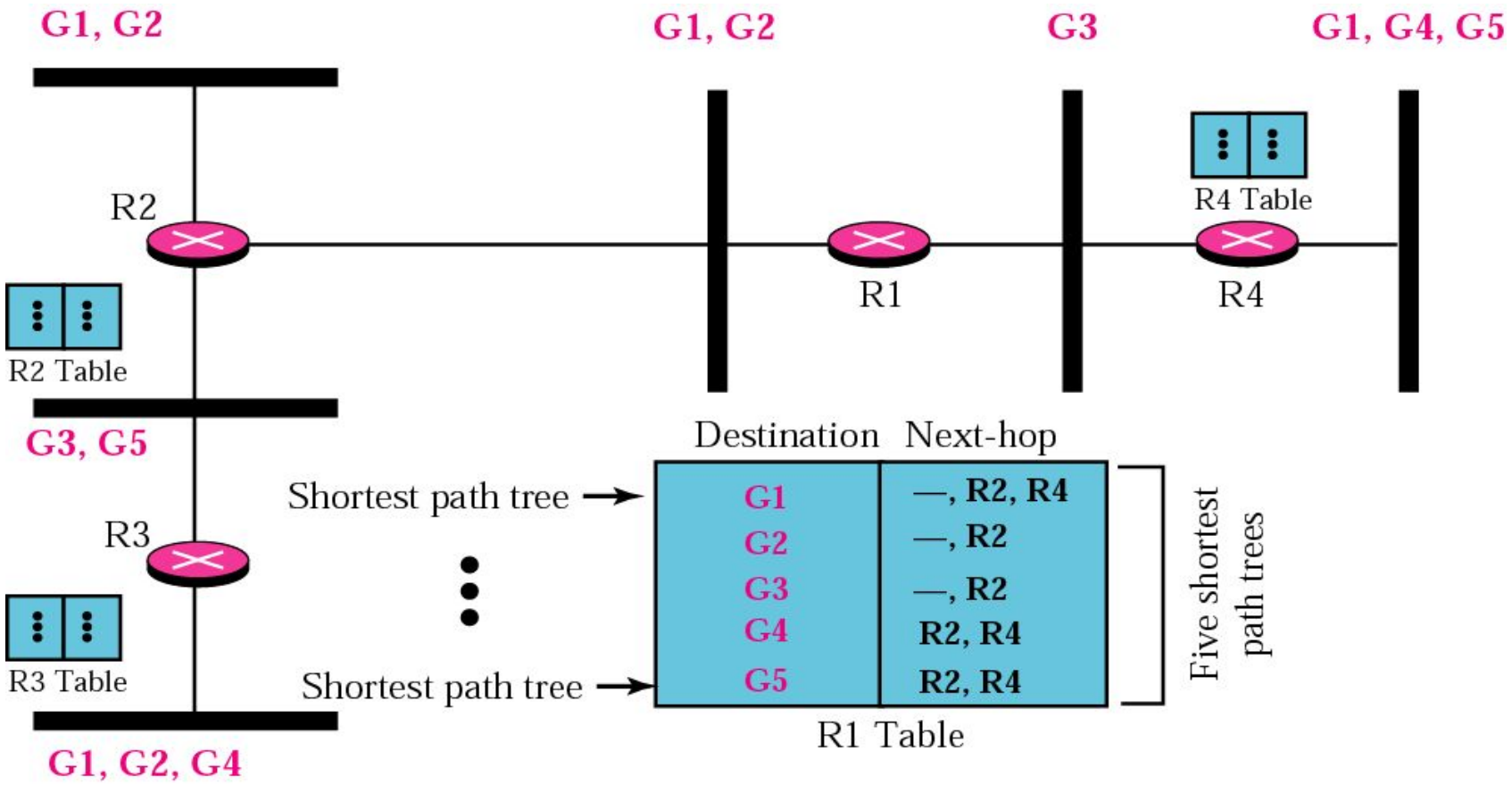


In multicast routing, each involved router needs to construct a shortest path tree for each group.

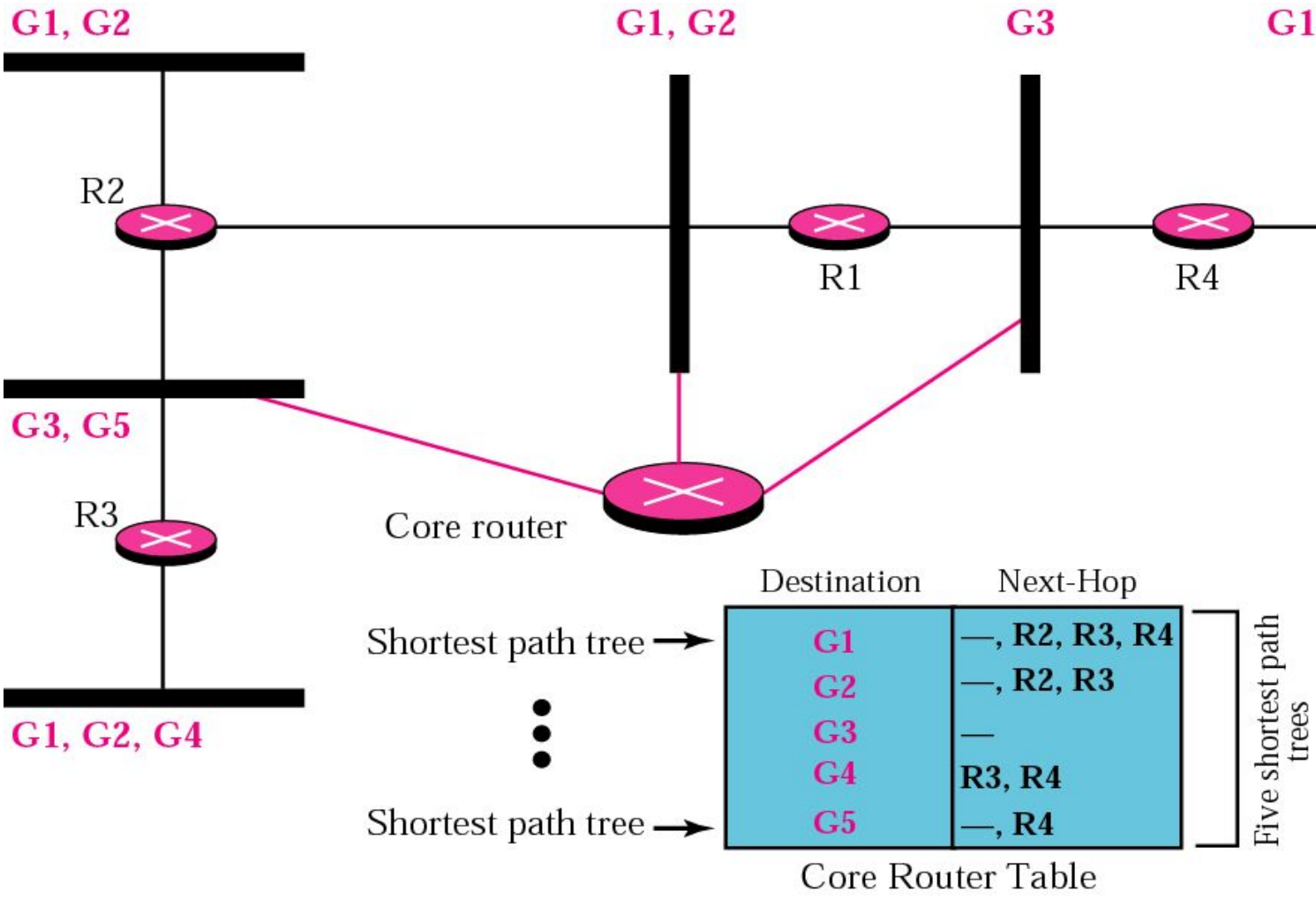


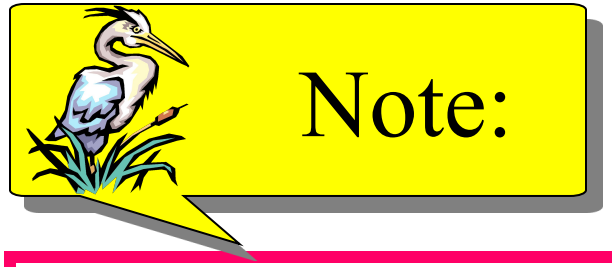
In the source-based tree approach, each router needs to have one shortest path tree for each group.

Source-based tree approach



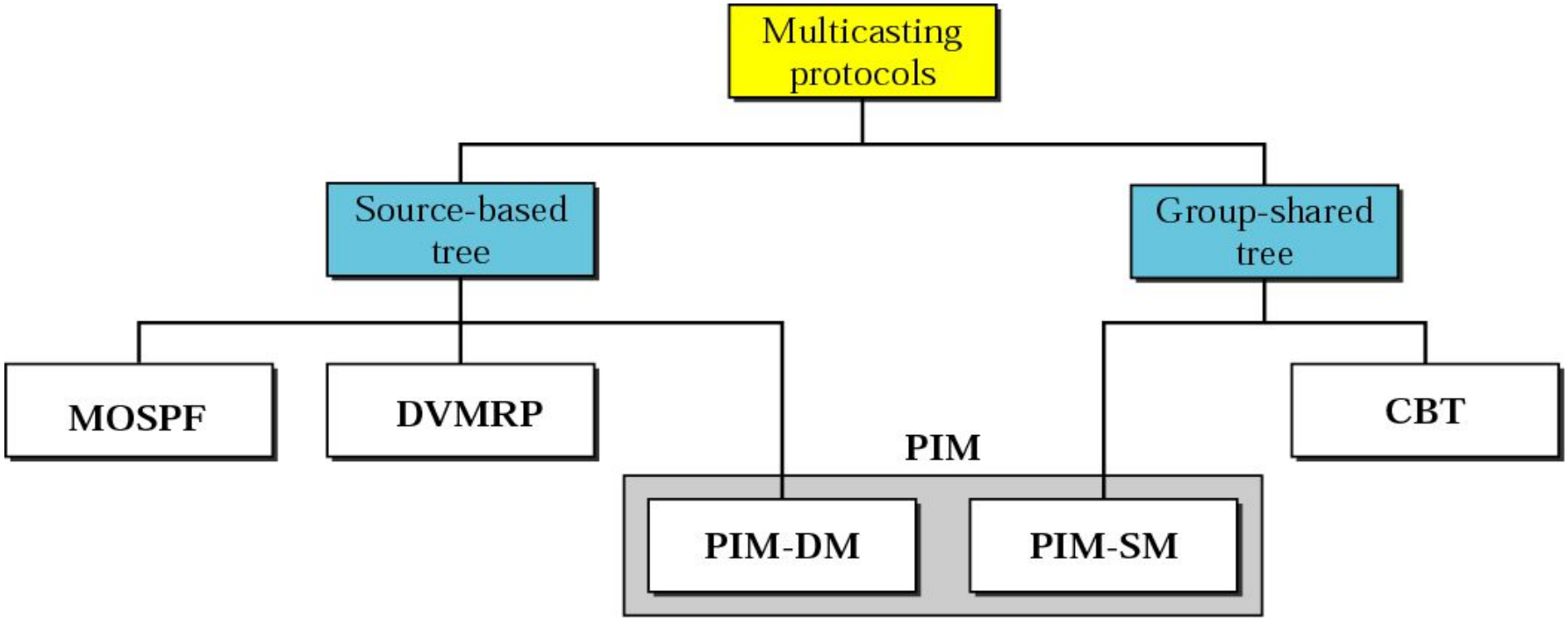
Group-shared tree approach





In the group-shared tree approach, only the core router, which has a shortest path tree for each group, is involved in multicasting.

Taxonomy of common multicast protocols

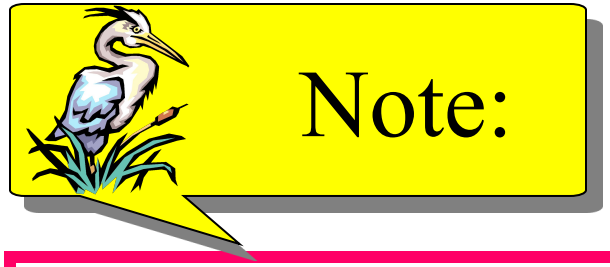


MULTICAST LINK STATE ROUTING: MOSPF

In this section, we briefly discuss multicast link state routing and its implementation in the Internet, MOSPF.

The topics discussed in this section include:

Multicast Link State Routing
MOSPF



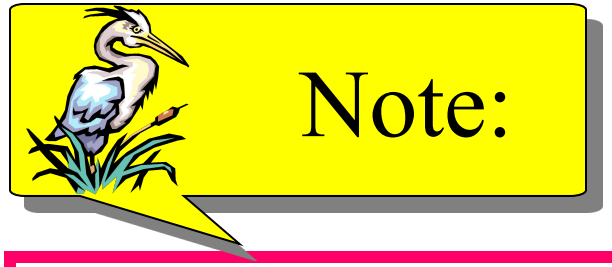
Multicast link state routing uses the source-based tree approach.

MULTICAST DISTANCE VECTOR: DVMRP

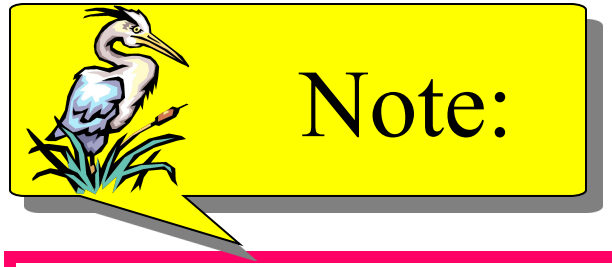
In this section, we briefly discuss multicast distance vector routing and its implementation in the Internet, DVMRP.

The topics discussed in this section include:

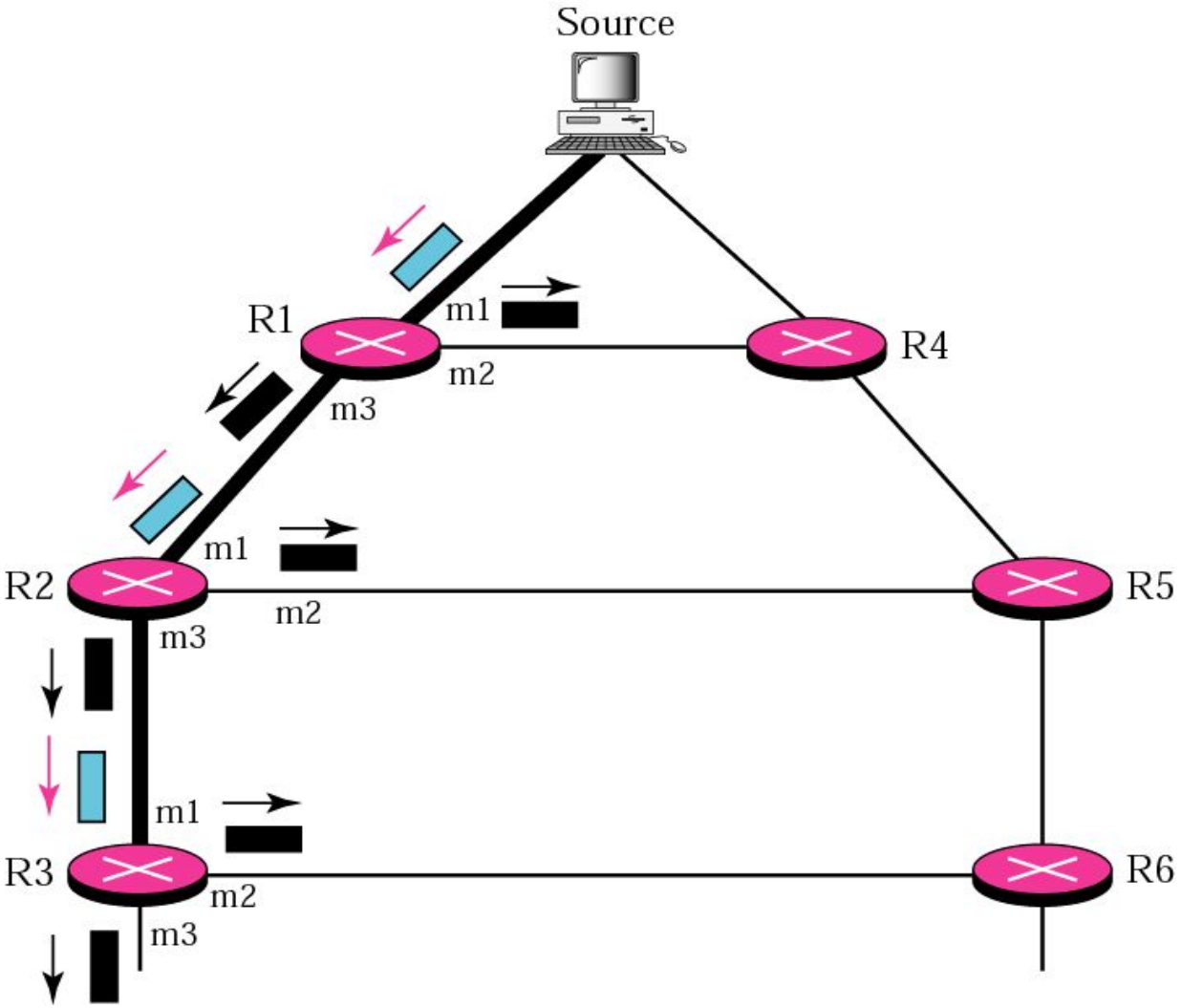
Multicast Distance Vector Routing
DVMRP



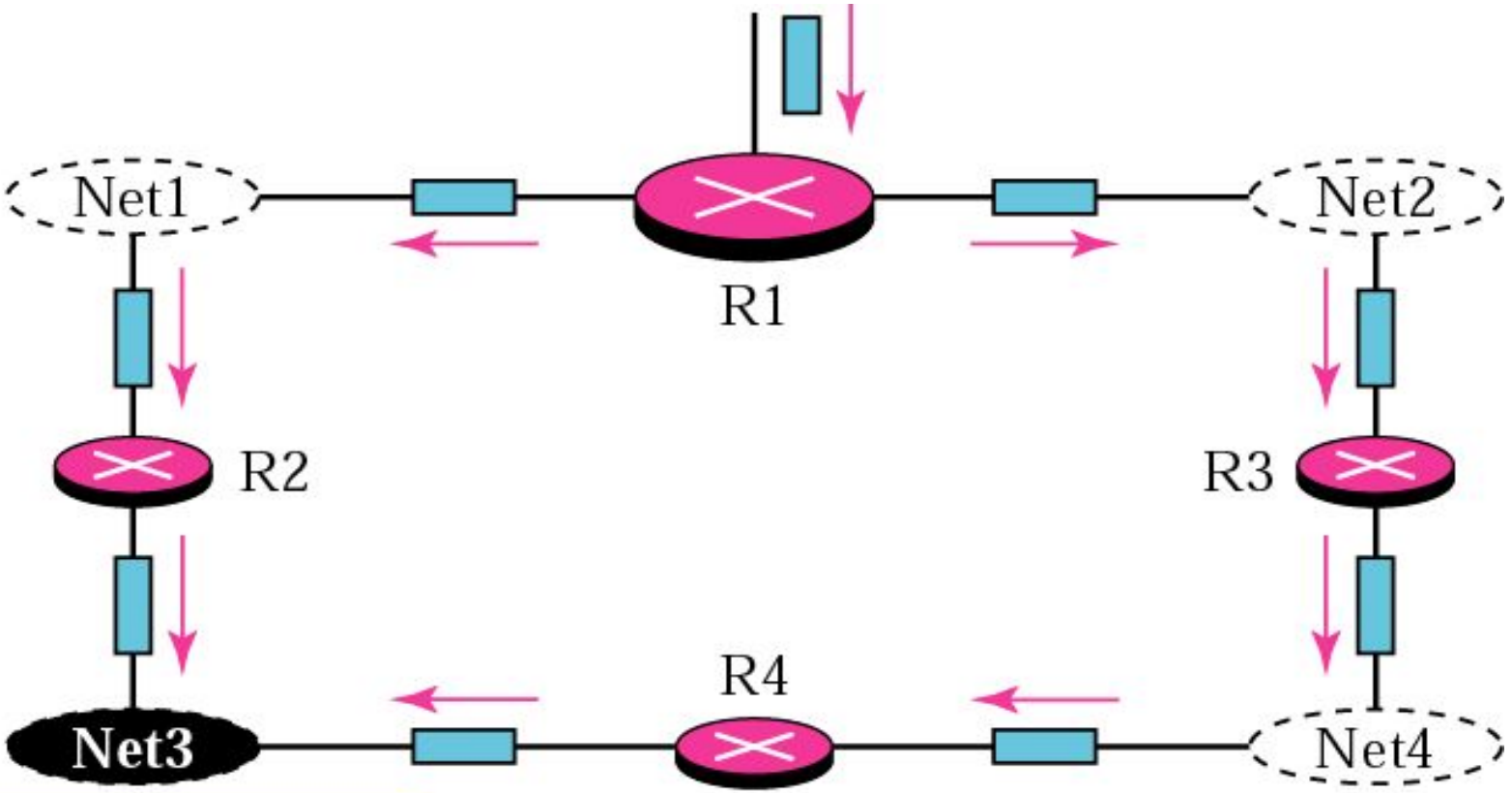
Flooding broadcasts packets, but creates loops in the systems.



RPF eliminates the loop in the flooding process.

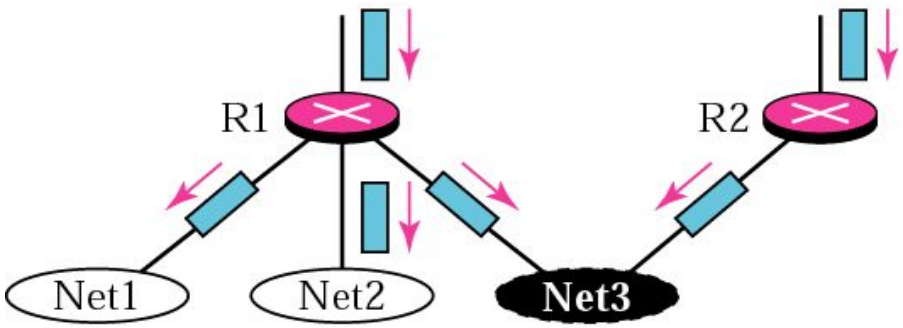

Source
R1
R2
R3
R4
R5
R6
m1
m2
m3
m1
m2
m3
m1
m2
m3

Problem with RPF

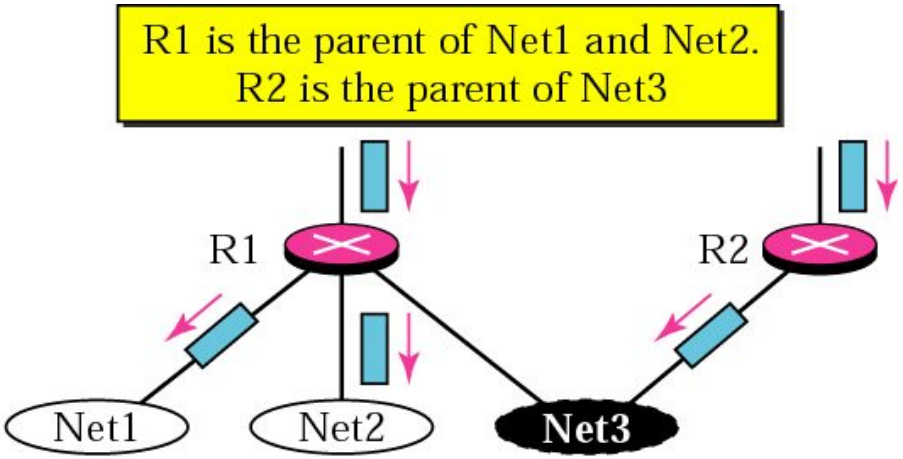


Net3 receives two copies of the packet

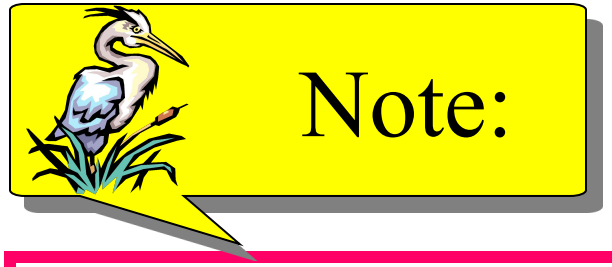
RPF versus RPB



a. RPF



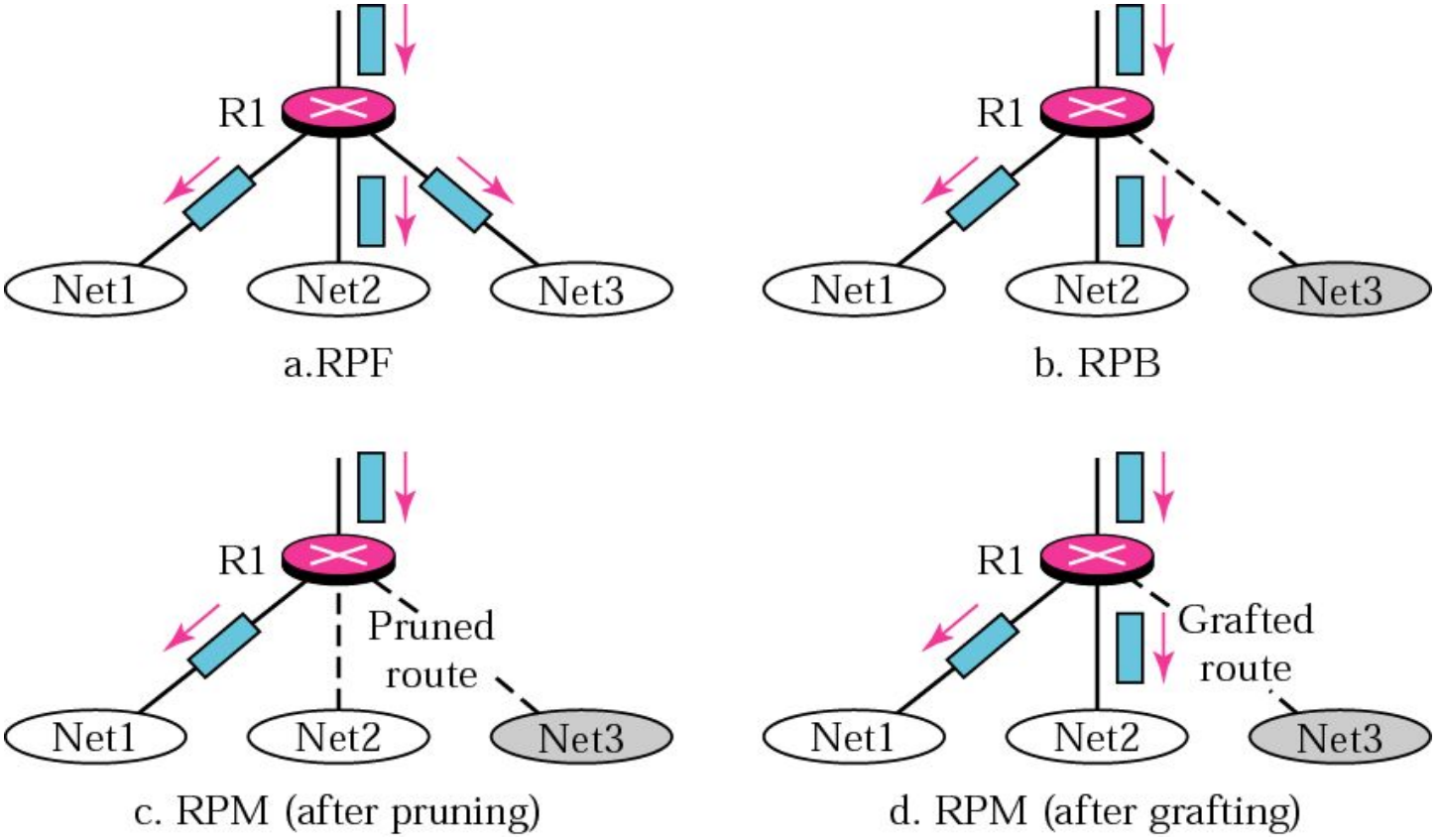
b. RPB

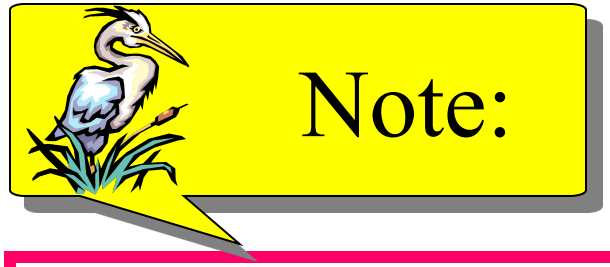


RPB creates a shortest path broadcast tree from the source to each destination.

It guarantees that each destination receives one and only one copy of the packet.

RPF, RPB, and RPM





RPM adds pruning and grafting to RPB to create a multicast shortest path tree that supports dynamic membership changes.

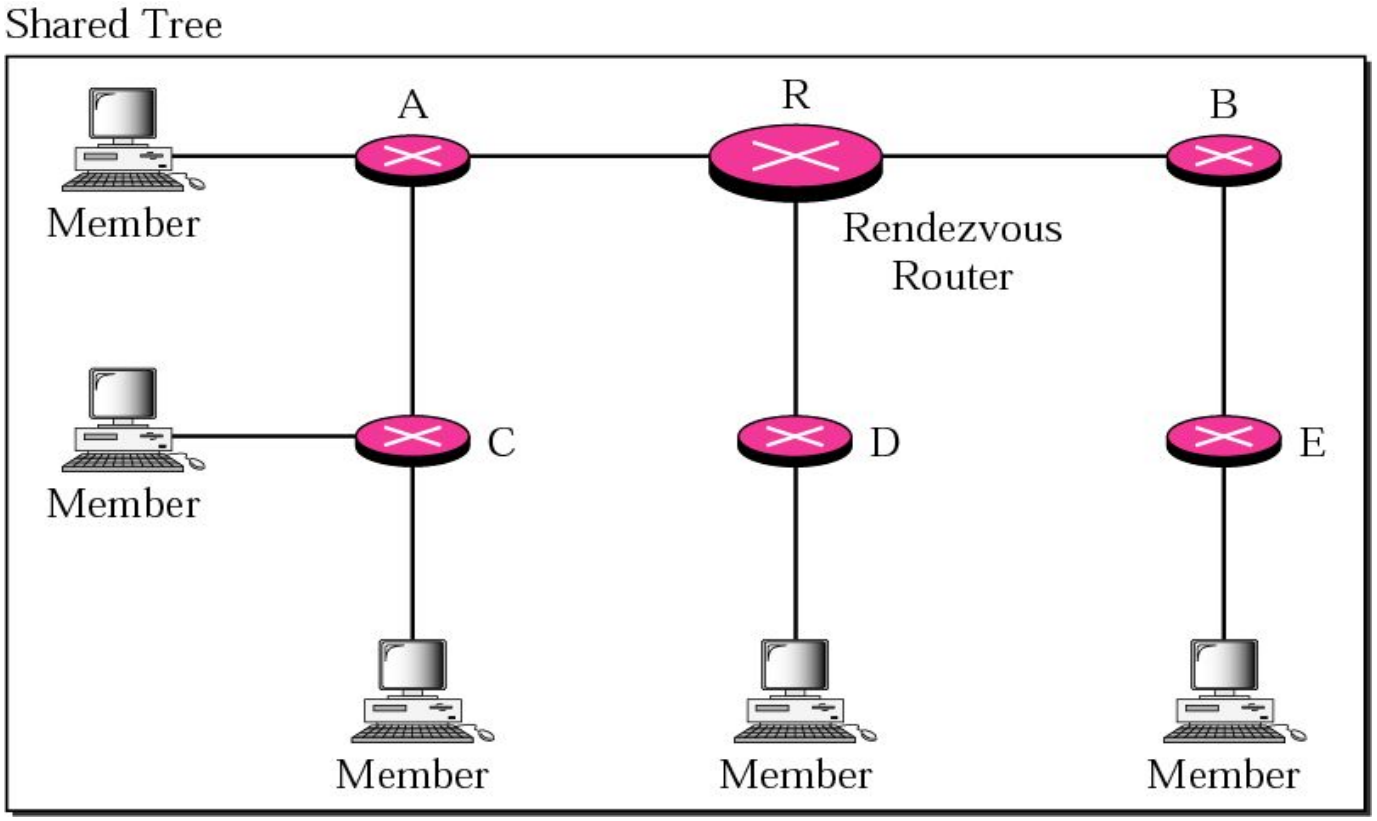
CBT

The Core-Based Tree (CBT) protocol is a group-shared protocol that uses a core as the root of the tree. The autonomous system is divided into regions and a core (center router or rendezvous router) is chosen for each region.

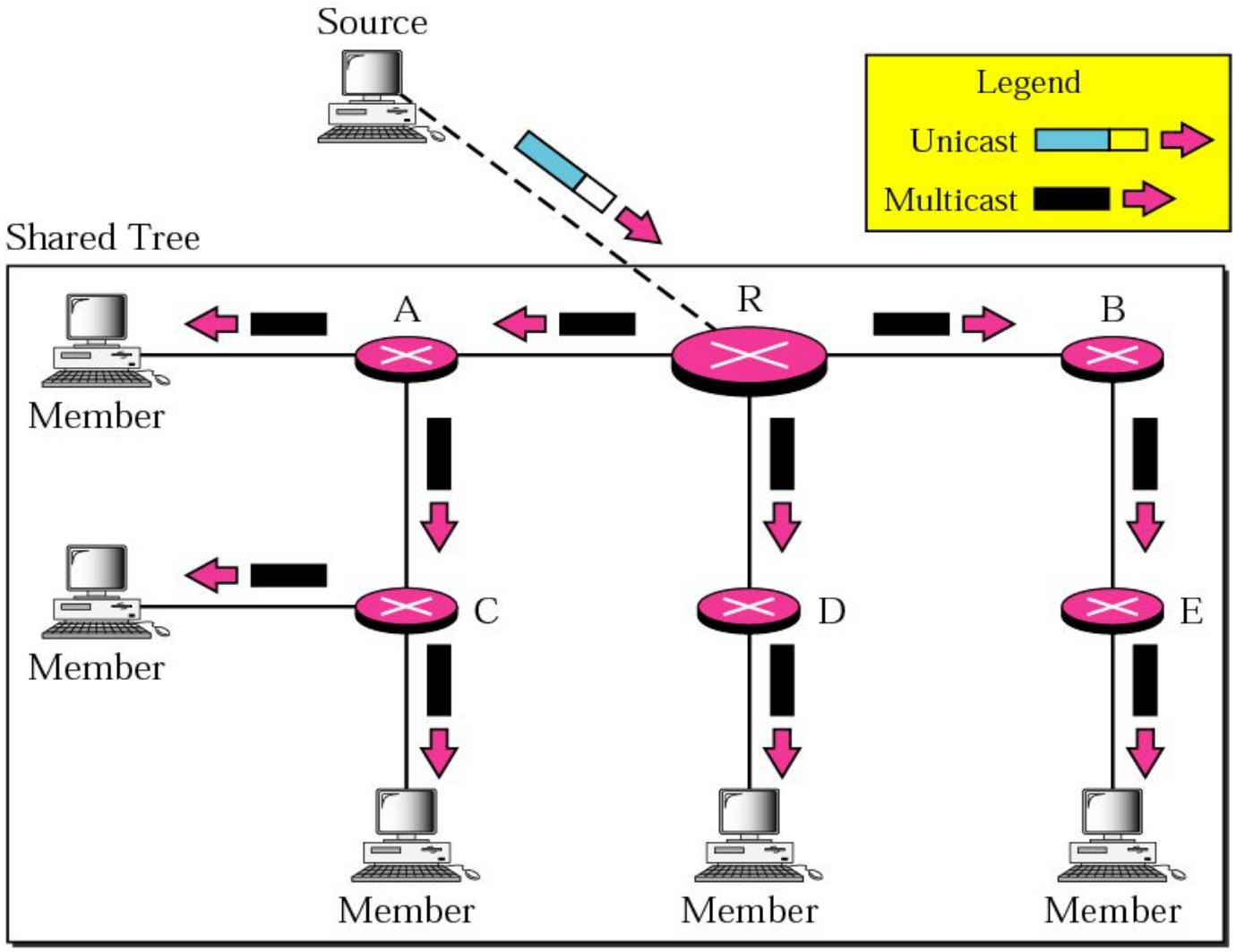
The topics discussed in this section include:

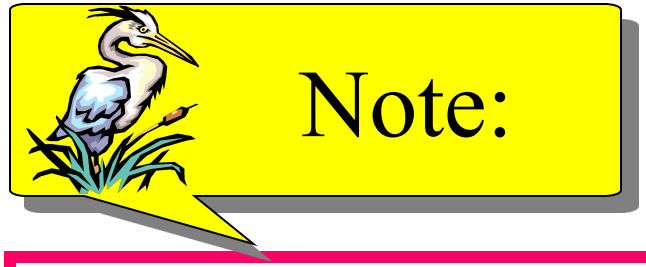
Formation of the Tree
Sending Multicast Packets
Selecting the Rendezvous Router

Group-shared tree with rendezvous router



Sending a multicast packet to the rendezvous router





Note:

In CBT, the source sends the multicast packet (encapsulated in a unicast packet) to the core router.

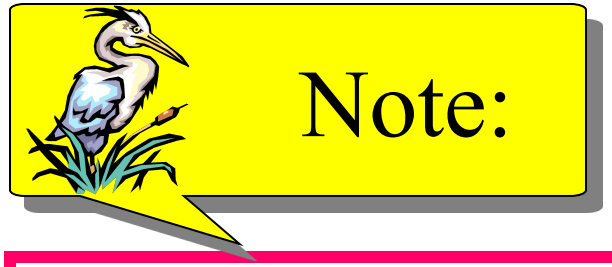
The core router decapsulates the packet and forwards it to all interested interfaces.

PIM

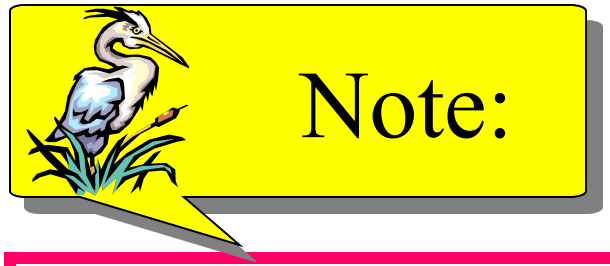
Protocol Independent Multicast (PIM) is the name given to two independent multicast routing protocols: Protocol Independent Multicast, Dense Mode (PIM-DM) and Protocol Independent Multicast, Sparse Mode (PIM-SM).

The topics discussed in this section include:

PIM-DM
PIM-SM

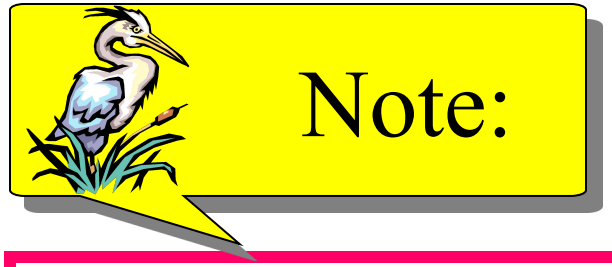


PIM-DM is used in a dense multicast environment, such as a LAN.

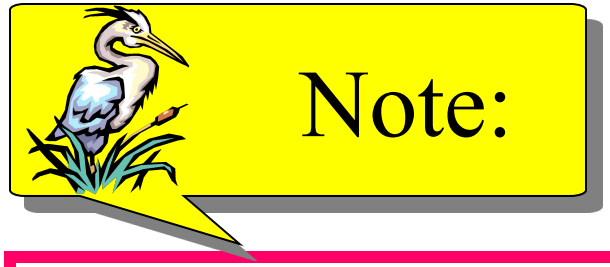


PIM-DM uses RPF and pruning/grafting strategies to handle multicasting.

However, it is independent from the underlying unicast protocol.



PIM-SM is used in a sparse multicast environment such as a WAN.

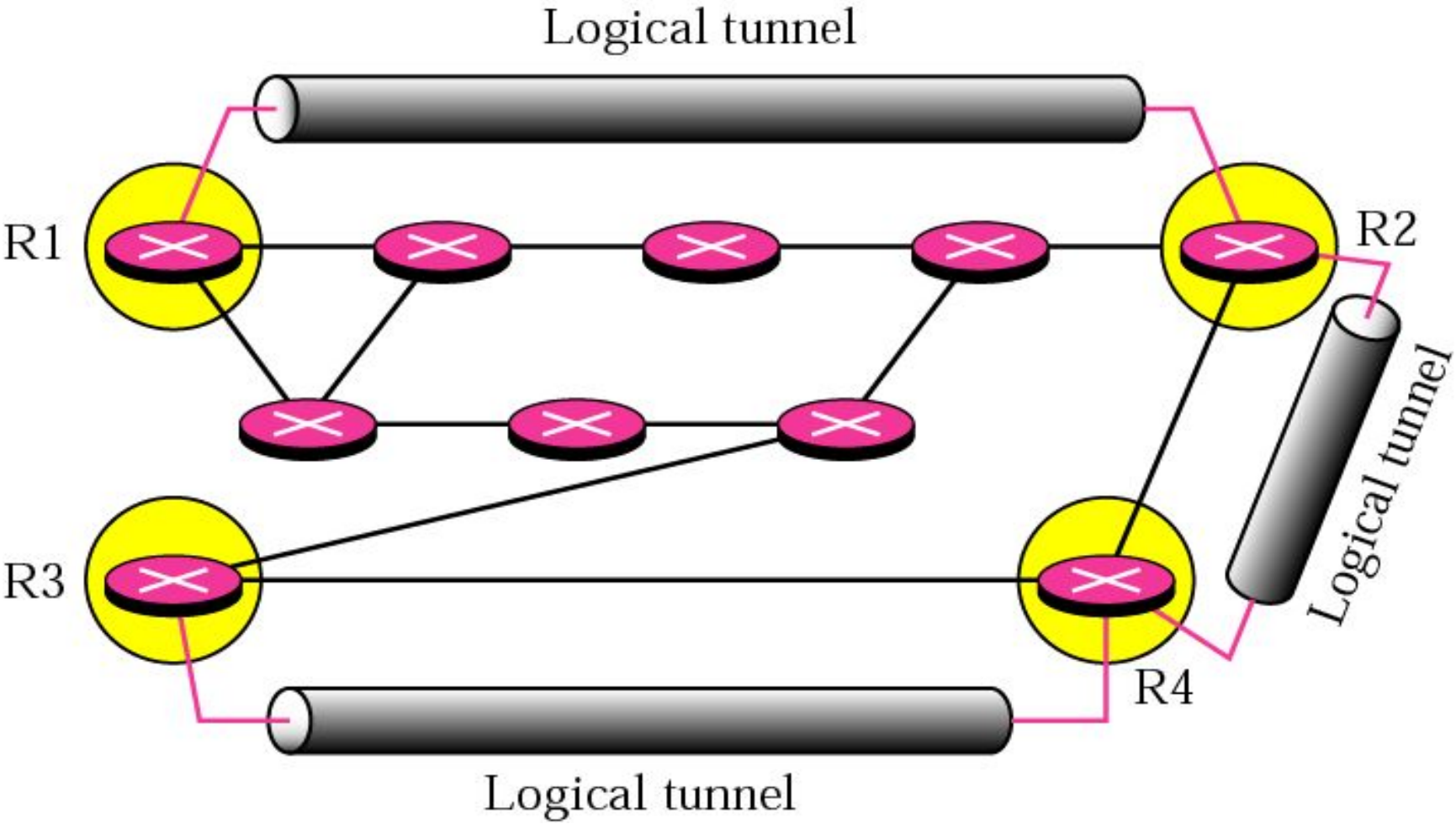


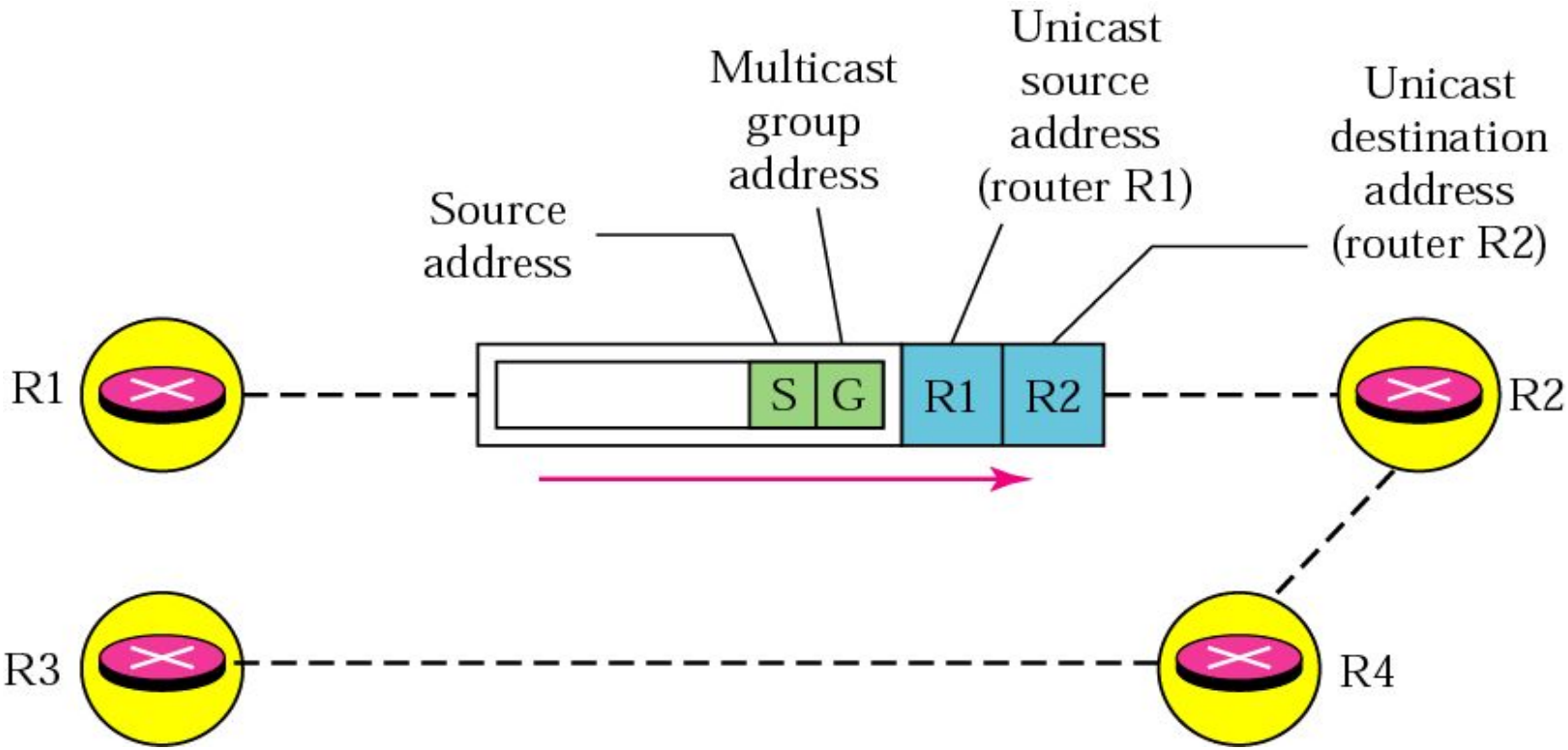
PIM-SM is similar to CBT but uses a simpler procedure.

MBONE

A multicast router may not find another multicast router in the neighborhood to forward the multicast packet. A solution for this problem is tunneling. We make a multicast backbone (MBONE) out of these isolated routers using the concept of tunneling.

Logical tunneling




Multicast
group
address
Unicast
source
address
(router R1)
Unicast
destination
address
(router R2)
Source
address
R1
S
G
R1
R2
R2
R3
R4