NETWORK LAYER

UNIT – 3

NANDINI S B

ASSISTANT PROFESSOR

Syllabus

Unit 3

Network Layer: Internet Protocol Version 4: IPv4 Addressing, Main and Auxiliary protocols, **Routing Algorithms:** Distance-Vector(DV) Routing Link-State Routing Unicast Routing Protocols: Internet Structure, Routing Information Protocol, Open Shortest Path First, Border Gateway Protocol version 4.

The Internet Protocol (IP): IPv4

- The Internet Protocol (IP): IPv4
  - IP Version 4 (IPv4) - almost depleted
  - IP Version 6 (IPv6)

IPv4 Addressing

- Internet address or IP address - Identifier used in IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet.

- An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet.
  - Unique – Each address defines one, and only one connection to the internet.
  - Universal - Addressing system accepted by any host that wants to be connected to the Internet

Address Space:

- An address space is the total number of addresses used by the protocol.
- If a protocol uses b bits to define an address, the address space is 2^b.
- IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than 4 billion).
- If there were no restriction, more than 4 billion devices could be connected to the internet.

Notation :

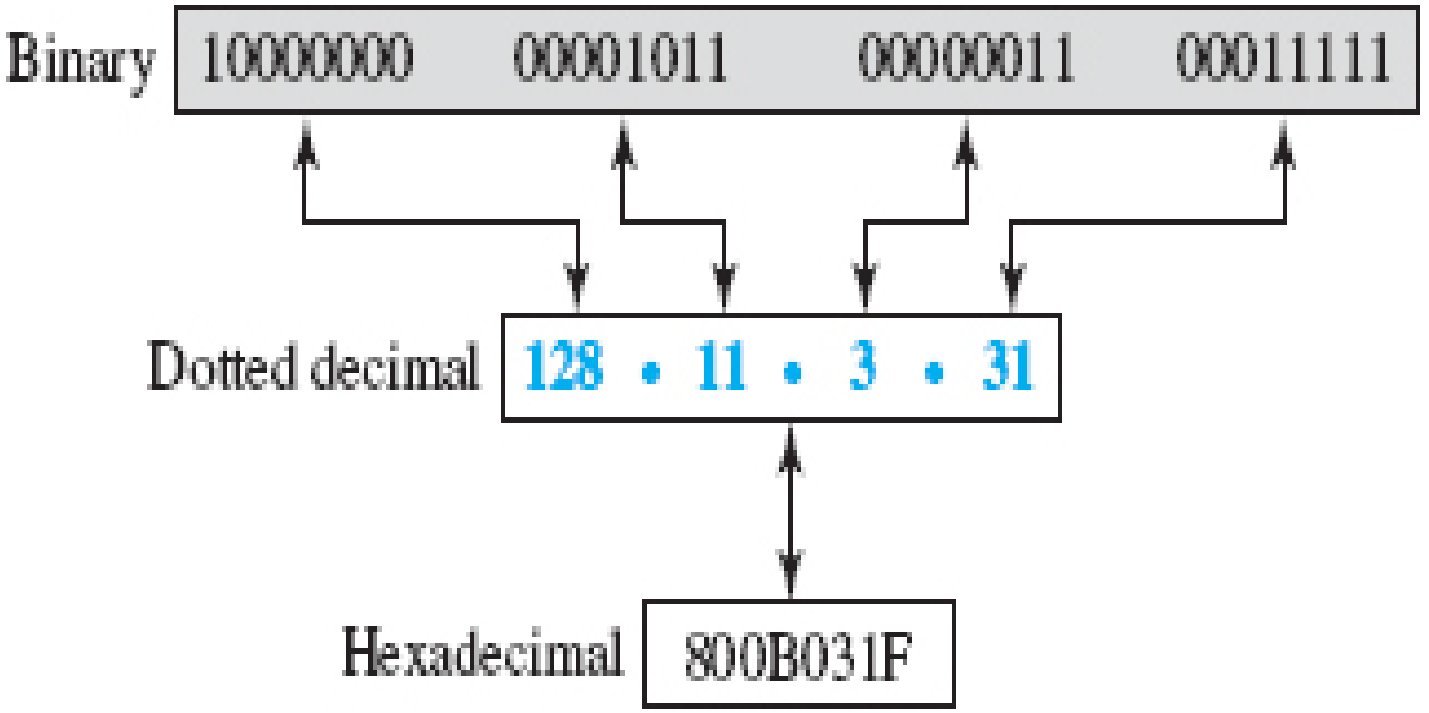


Figure : Three different notations in IPv4 addressing

- There are three common notations to show an IPv4 address:
  - binary notation (base 2),
  - dotted-decimal notation (base 256), and
  - hexadecimal notation (base 16).
- Binary notation → An IPv4 address is displayed as 32 bits.
- Dotted-decimal notation → To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes. each number in the dotted-decimal notation is between 0 and 255.
- Hexadecimal notation → Each hexadecimal digit is equivalent to 4 bits. This means that a 32-bit address has eight hexadecimal digits. This notation is often used in network programming.

Hierarchy in Addressing :

- A 32-bit IPv4 address is also hierarchical but is divided only into two parts.
  - The first part of the address, called the prefix, defines the network.
  - The second part of the address, called the suffix, defines the node (connection of a device to the Internet).
- *Prefix-defines the network (n-bits)*
  - fixed length -classful addressing - obsolete
  - variable length-classless addressing-

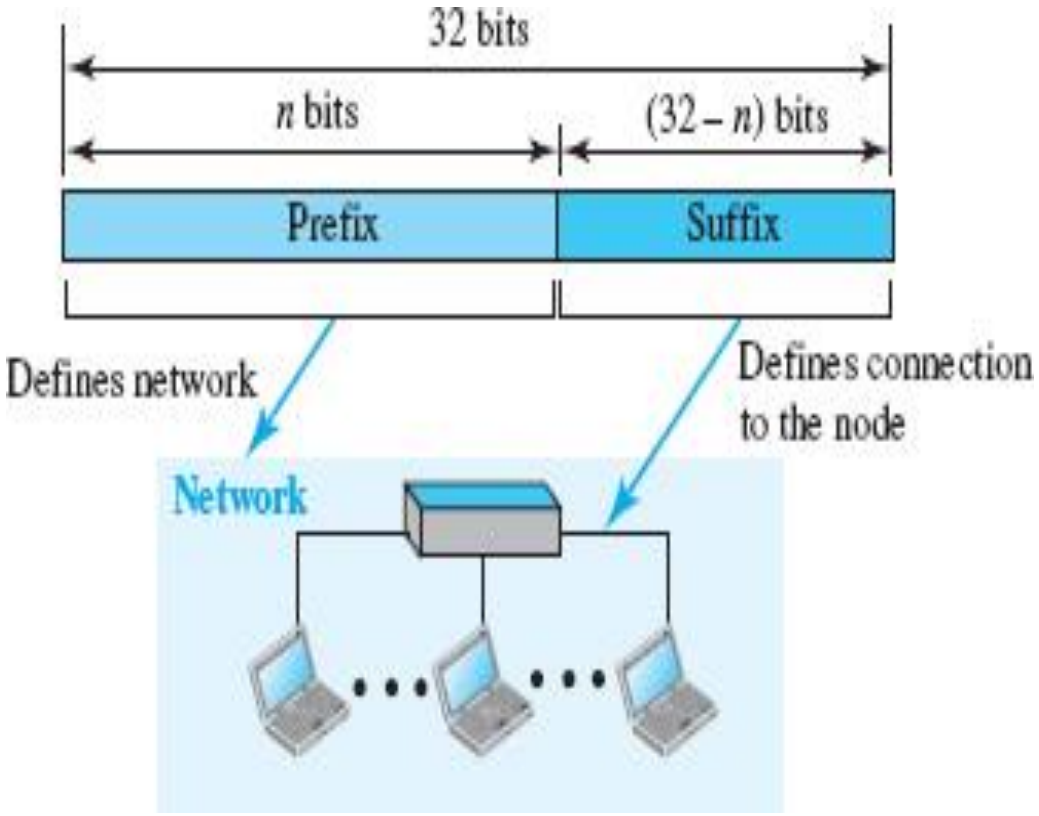
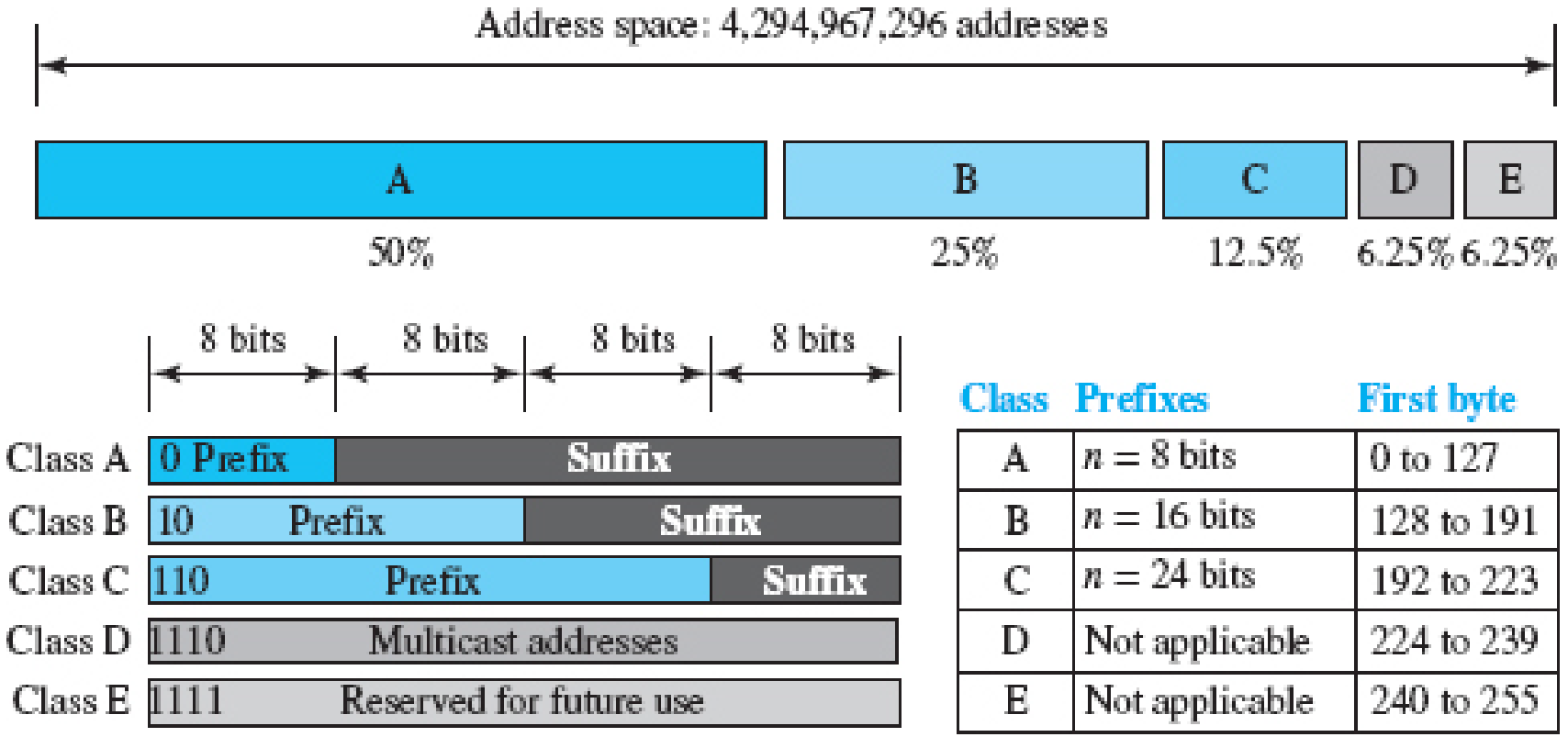


Figure : Hierarchy in addressing

Classful Addressing :



Class	Prefixes	First byte
A	$n = 8$ bits	0 to 127
B	$n = 16$ bits	128 to 191
C	$n = 24$ bits	192 to 223
D	Not applicable	224 to 239
E	Not applicable	240 to 255

Figure : Occupation of the address space in classful addressing

- When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one (n = 8, n = 16, and n = 24).
- The whole address space was divided into five classes (classes A, B, C, D, and E). This scheme is referred to as classful addressing.
- class A → the network length is 8 bits, but because the first bit, which is 0, defines the class, we can have only 7 bits as the network identifier. This means there are only $2^7 = 128$ networks in the world that can have a class A address.
- class B → the network length is 16 bits, but because the first 2 bits, which are $(10)_2$, define the class, we can have only 14 bits as the network identifier. This means there are only $2^{14} = 16{,}384$ networks in the world that can have a class B address.

- Class C → the network length is 24 bits, but because 3 bits define the class, we can have only 21 bits as the network identifier. This means there are 2 21 = 2,097,152 networks in the world that can have a class C address.
- Class D is not divided into prefix and suffix. It is used for multicast addresses.
- All addresses that start with 1111 in binary belong to class E. As in class D, class E is not divided into prefix and suffix and is used as reserve.

Address Depletion:

- The addresses were not distributed properly so the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses being available for organizations and individuals that needed to have an Internet connection.
- Eg. class A -assigned to only 128 organizations in the world, but each organization would need to have one single network (seen by the rest of the world) with 16,777,216 nodes (computers in this single network).
  - Because there were only a few organizations that are this large, most of the addresses in this class were wasted (unused).
- Class B addresses were designed for midsize organizations, but many of the addresses in this class also remained unused.

- Class C addresses have a completely different design flaw. The number of addresses that can be used in each network (256) was so small that most companies were not comfortable using a block in this address.
- Class E addresses were almost never used, wasting the whole class.

Solution:

larger address space requires the length of IP addresses be increased -means the format of the IP packets needs to be changed (IPv6) - long-range solution

Short term solution – *classless addressing (*no classes)

Classless Addressing :

- In classless addressing, variable-length blocks are used that belong to no classes.
- We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and so on.
  - 2^0 , 2^1 , 2^2 , …, 2^{32} addresses -- One of the restrictions is that the number of addresses in a block needs to be a power of 2.
- The prefix in an address defines the block (network);
- the suffix defines the node (device).
- An organization can be granted one block of addresses.

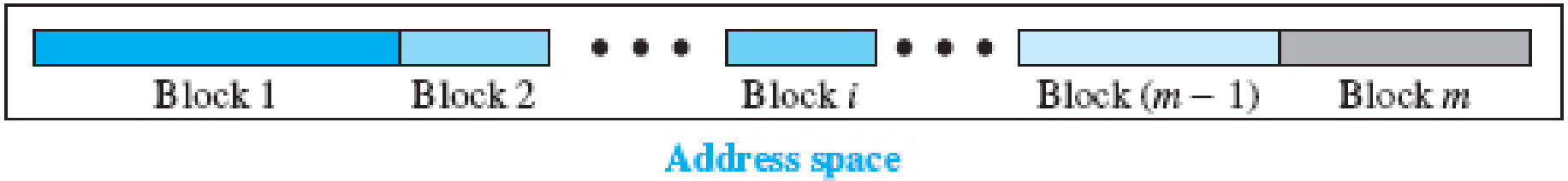


Figure : Variable-length blocks in classless addressing

Prefix Length: Slash Notation :

- The prefix length is not inherent in the address, we need to separately give the length of the prefix.
- In this case, the prefix length, n, is added to the address, separated by a slash.
- The notation is informally referred to as slash notation and formally as Classless Interdomain routing (CIDR, pronounced cider) strategy.

Examples:
12.24.76.8/8
23.14.67.92/12
220.8.24.255/25

Figure : Slash notation (CIDR)

Extracting Information from an Address :

- Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs:
  - number of addresses,
  - first address in the block, and
  - last address.
- If the value of prefix length, n, is given, we can easily find these three pieces of information.
  1. To find the first address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 0s.
  2. 2. To find the last address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 1s.

Example 1

A classless address is given as 167.199.170.82/27. We can find the desired three pieces of information as follows. The number of addresses in the network is $2^{32-n} = 2^5 = 32$ addresses. The first address can be found by keeping the first 27 bits and changing the rest of the bits to 0s.

Address: 167.199.170.82/27	10100111 11000111 10101010 01010010
First address: 167.199.170.64/27	10100111 11000111 10101010 01000000

The last address can be found by keeping the first 27 bits and changing the rest of the bits to 1s.

Address: 167.199.170.82/27	10100111 11000111 10101010 01011111
Last address: 167.199.170.95/27	10100111 11000111 10101010 01011111

Address Mask

- Another way to find the first and last addresses in the block is to use the address mask.
- The address mask is a 32-bit number in which the *n leftmost* bits are set to 1s and the rest of the bits (32 – *n) are set to 0s.*
- *A computer* can easily find the address mask because it is the complement of ($2^{32-n} - 1$).

The reason for defining a mask in this way is that it can be used by a computer program to extract the information in a block, using the three bitwise operations NOT, AND, and OR.

1. The number of addresses in the block *N = NOT (Mask) + 1.*

2. The first address in the block = (Any address in the block) AND (Mask).

3. The last address in the block = (Any address in the block) OR [(NOT (Mask)].

Address Mask

Repeat Example 1 using the mask. The mask in dotted-decimal notation is 256.256.256.224. The AND, OR, and NOT operations can be applied to individual bytes using calculators and applets at the book website.

Number of addresses in the block: *N =* ***NOT (mask) + 1 = 0.0.0.31 + 1 =*** 32 addresses

First address: First = (address) **AND (mask) =** 167.199.170. 82

Last address: Last = (address) **OR (NOT mask) =** 167.199.170. 255

Example

We repeat Example 1 using the mask.

The mask in dotted-decimal notation is 256.256.256.224.

The AND, OR, and NOT operations can be applied to individual bytes using calculators and applets at the book website.

Number of addresses in the block: N = NOT (mask) + 1 = 0.0.0.31 + 1 = 32 addresses

First address: First = (address) AND (mask) = 167.199.170. 82

Last address: Last = (address) OR (NOT mask) = 167.199.170. 255

Example

In classless addressing, an address cannot per se(itself) define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks.

Some of them are shown here with the value of the prefix associated with that block.

Prefix length:16 → Block: 230.8.0.0 to 230.8.255.255

Prefix length:20 → Block: 230.8.16.0 to 230.8.31.255

Prefix length:26 → Block: 230.8.24.0 to 230.8.24.63

Prefix length:27 → Block: 230.8.24.32 to 230.8.24.63

Prefix length:29 → Block: 230.8.24.56 to 230.8.24.63

Prefix length:31 → Block: 230.8.24.56 to 230.8.24.57

Network Address

- The preceding examples show that, given any address, we can find all information about the block.
- The first address, the **network address, is** particularly important because it is used in routing a packet to its destination network.

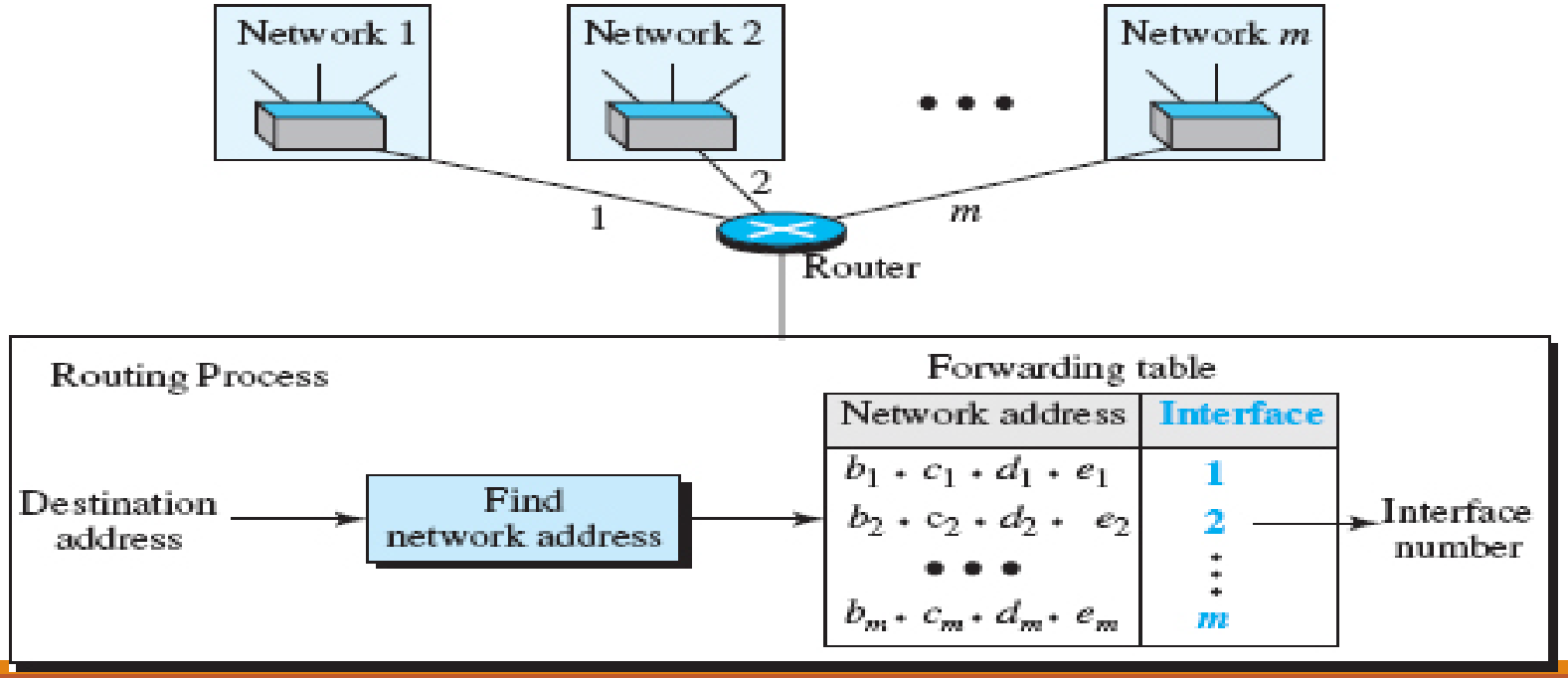


Figure : Network address

- For the moment, let us assume that an internet is made up of *m* networks and a router with *m interfaces.*
  - *When a packet arrives at the router* from any source host, the router needs to know to which network the packet should be sent and from which interface the packet should be sent out.
  - When the packet arrives at the network, it reaches its destination host using link layer addressing
- After the network address has been found, the router consults its forwarding table to find the corresponding interface from which the packet should be sent out.

The network address is actually the identifier of the network; each network is identified by its network address.

Block Allocation

- The next issue in classless addressing is block allocation. How are the blocks allocated?
- The ultimate responsibility of block allocation is given to a global authority called the **Internet Corporation for Assigned Names and Numbers (ICANN)**.
- However, ICANN does not normally allocate addresses to individual Internet users. It assigns a large block of addresses to an ISP.
- For the proper operation of the CIDR, two restrictions need to be applied to the allocated block.

1. The number of requested addresses, N, needs to be a power of 2.

2. The requested block needs to be allocated where there are a contiguous number of available addresses in the address space.

First address = (prefix in decimal) × 2^{32-n} = (prefix in decimal) × N

Example 4

An ISP has requested a block of 1000 addresses. Because 1000 is not a power of 2, 1024 addresses are granted.

The prefix length is calculated as $n = 32 - \log_2 1024 = 22$.

An available block, 18.14.12.0/22, is granted to the ISP.

It can be seen that the first address in decimal is 302,910,464, which is divisible by 1024.

Subnetting :

- More levels of hierarchy can be created using subnetting.
- An organization (or an ISP) that is granted a range of addresses may divide the range into several subranges and assign each subrange to a subnetwork (or subnet).
- Note that nothing stops the organization from creating more levels.
- A subnetwork can be divided into several sub-subnetworks. A sub-subnetwork can be divided into several sub-sub-subnetworks, and so on.

Designing Subnets

- The subnetworks in a network should be carefully designed to enable the routing of packets.
- We assume the total number of addresses granted to the organization is N, the prefix length is n, the assigned number of addresses to each subnetwork is N_{sub} , and the prefix length for each subnetwork is n_{sub} .
- Steps need to be carefully followed to guarantee the proper operation of the subnetworks.
  - The number of addresses in each subnetwork should be a power of 2.
  - The prefix length for each subnetwork should be found using the following formula:

$$n_{sub} = 32 - \log_2 N_{sub}$$

  - The starting address in each subnetwork should be divisible by the number of addresses in that subnetwork. This can be achieved if we first assign addresses to larger subnetworks.

Finding Information about Each Subnetwork

- After designing the subnetworks, the information about each subnetwork, such as first and last address, can be found using the process we described to find the information about each network in the Internet.

- Example:

An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have three sub blocks of addresses to use in its three subnets:

1. One subblock of 10 addresses
2. One subblock of 60 addresses, and
3. One subblock of 120 addresses. Design the subblocks.

Solution:

There are $2^{32-24} = 256$ addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24.

To satisfy the third requirement, we assign addresses to subblocks, starting with the largest and ending with the smallest one.

1. The number of addresses in the largest subblock, which requires 120 addresses, is not a power of 2. we allocate 128 addresses. The subnet mask for this subnet can be found as n1 = 32 – $\log_2 128 = 25$.

The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.

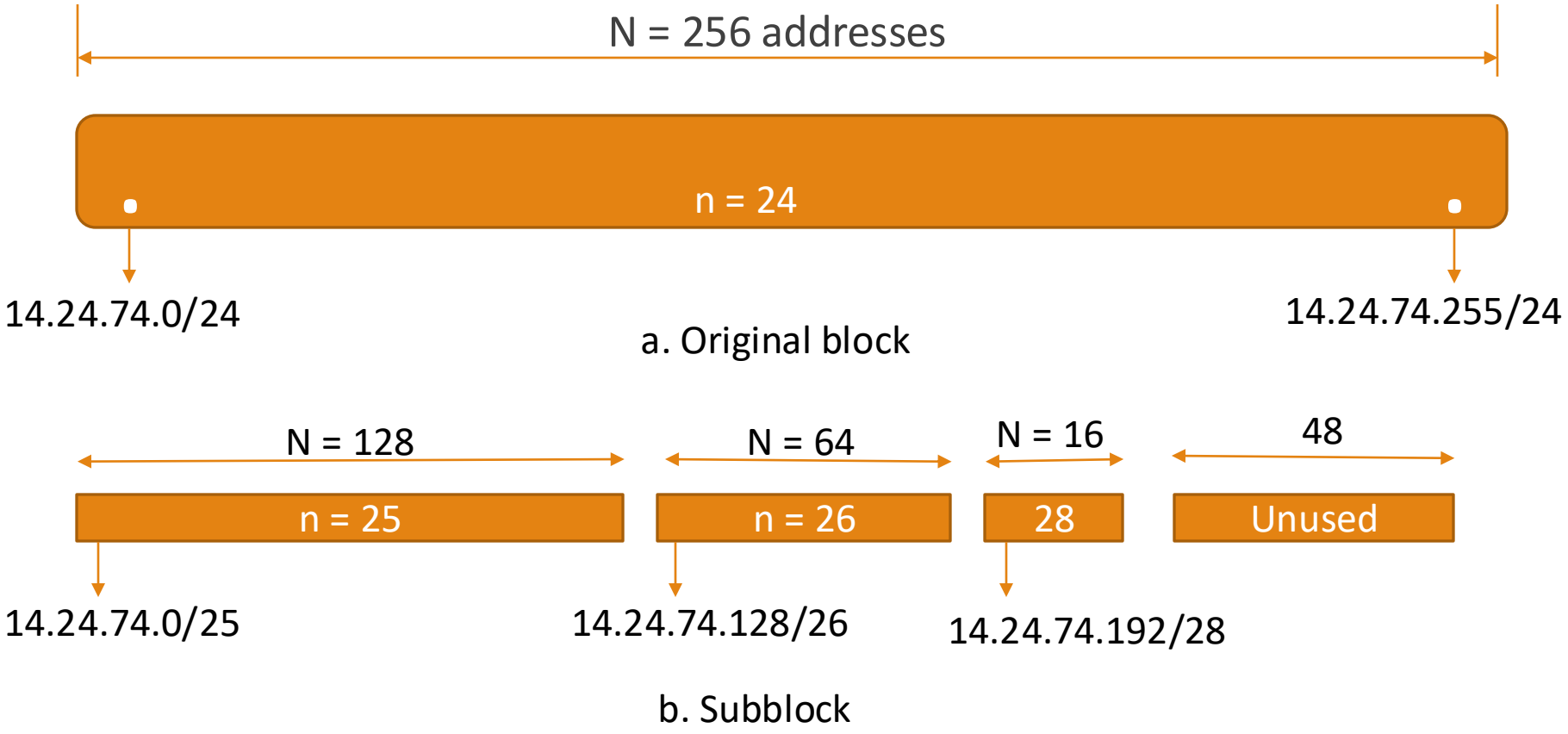
2. The number of addresses in the second largest subblock, which requires 60 addresses, is not a power of 2 either. we allocate 64 addresses. The subnet mask for this subnet can be found as $n2=32 - \log_2 64 = 26$.

The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.

3. The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either. we allocate 16 addresses. The subnet mask for this subnet can be found as $n3=32 - \log_2 16 = 28$.

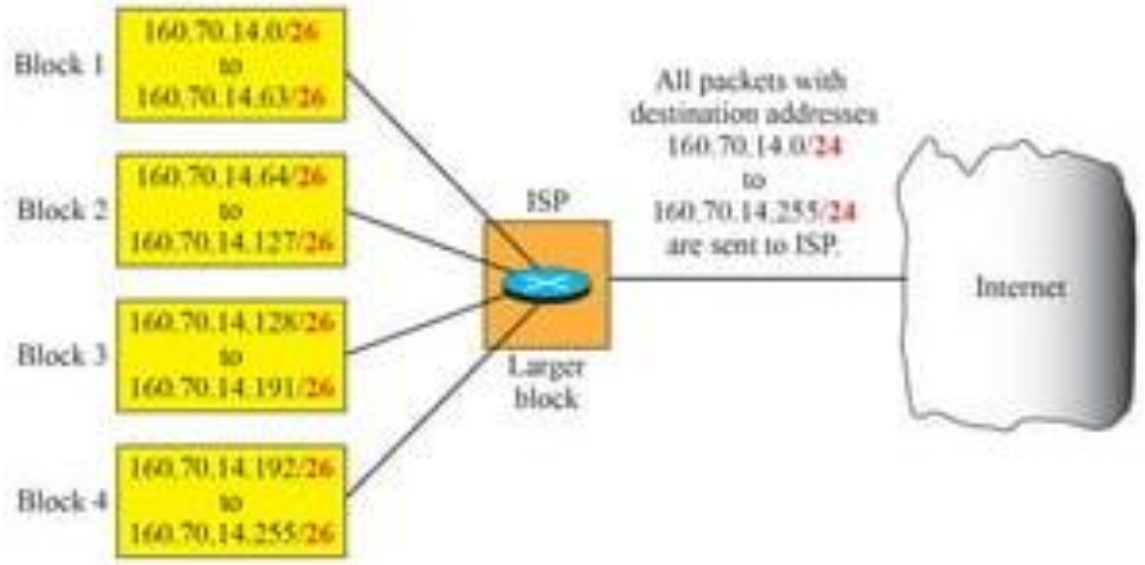
The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

- If we add all addresses in the subblock = 208 addresses.
- 48 addresses are left in reserve.


N = 256 addresses
n = 24
14.24.74.0/24
14.24.74.255/24
a. Original block
N = 128
N = 64
N = 16
48
n = 25
n = 26
28
Unused
14.24.74.0/25
14.24.74.128/26
14.24.74.192/28
b. Subblock

Address Aggregation

- One of the advantages of the CIDR strategy is address aggregation (sometimes called address summarization or route summarization). When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block.



Example of address aggregation

Main and Auxiliary Protocols

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

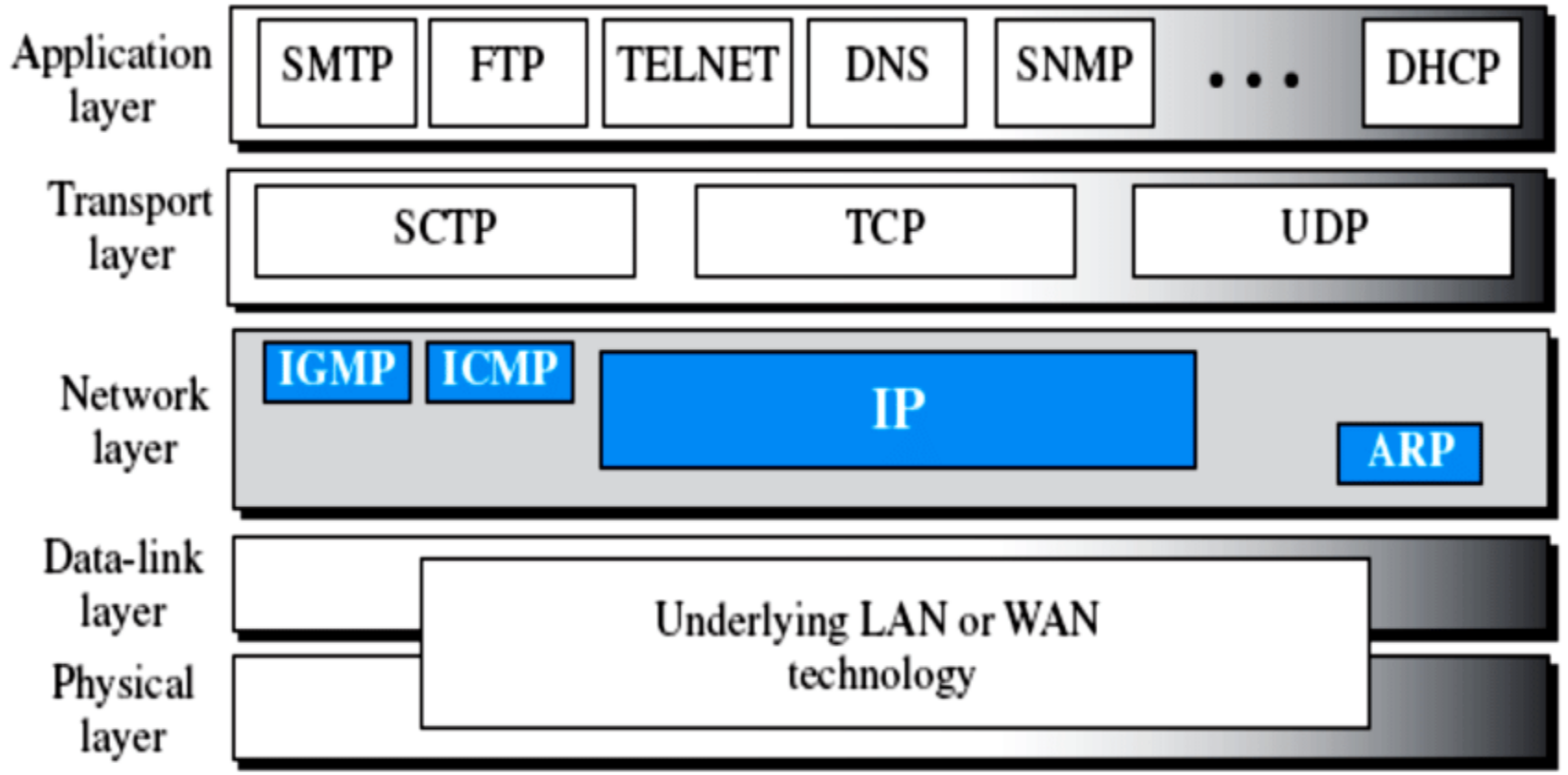
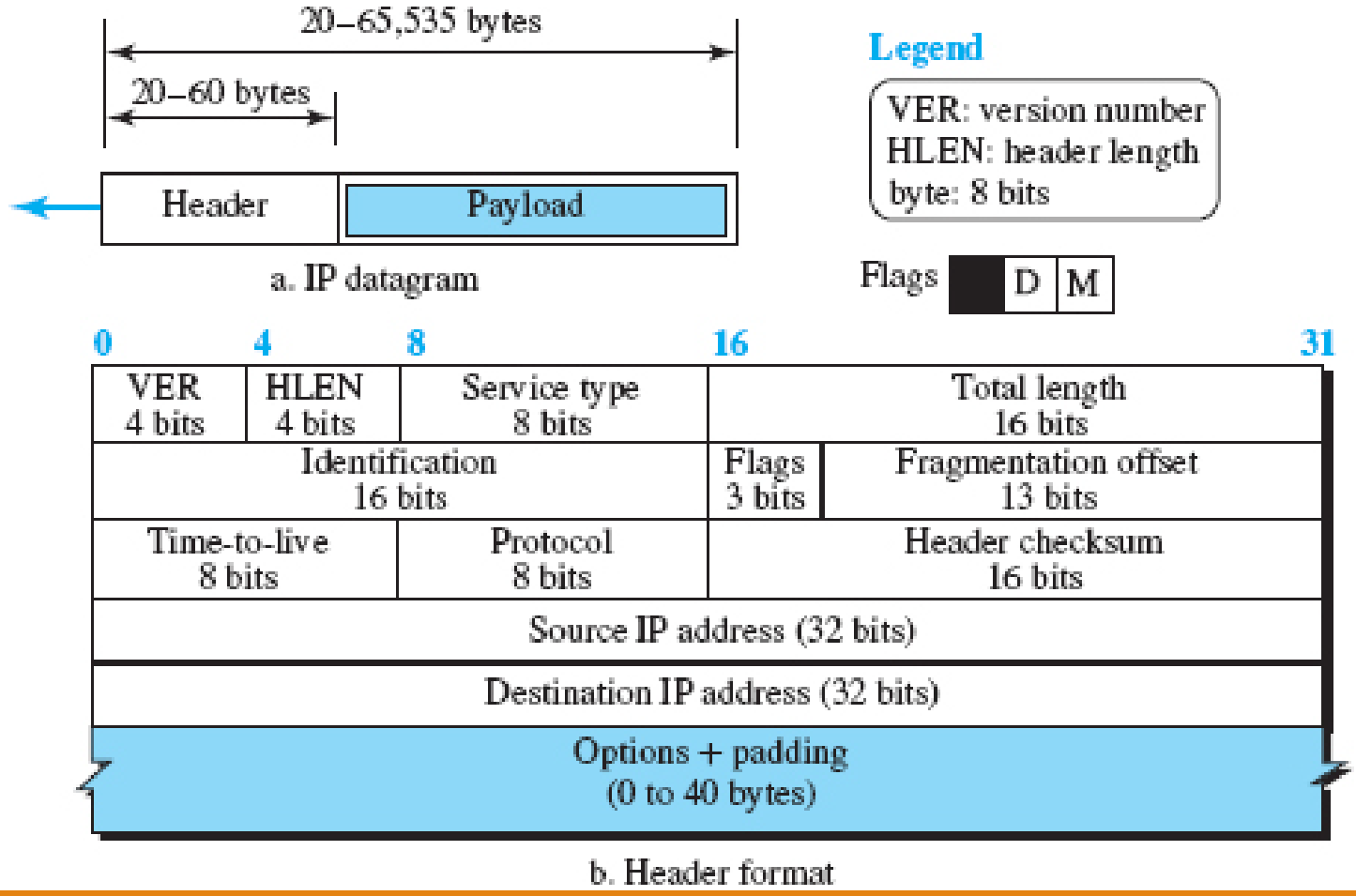


Figure : Position of IP and other network-layer protocols in TCP/IP protocol suite

- IPv4 is an unreliable datagram protocol.
- If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP.
- IPv4 is also a connectionless protocol that uses the datagram approach.

Datagram Format



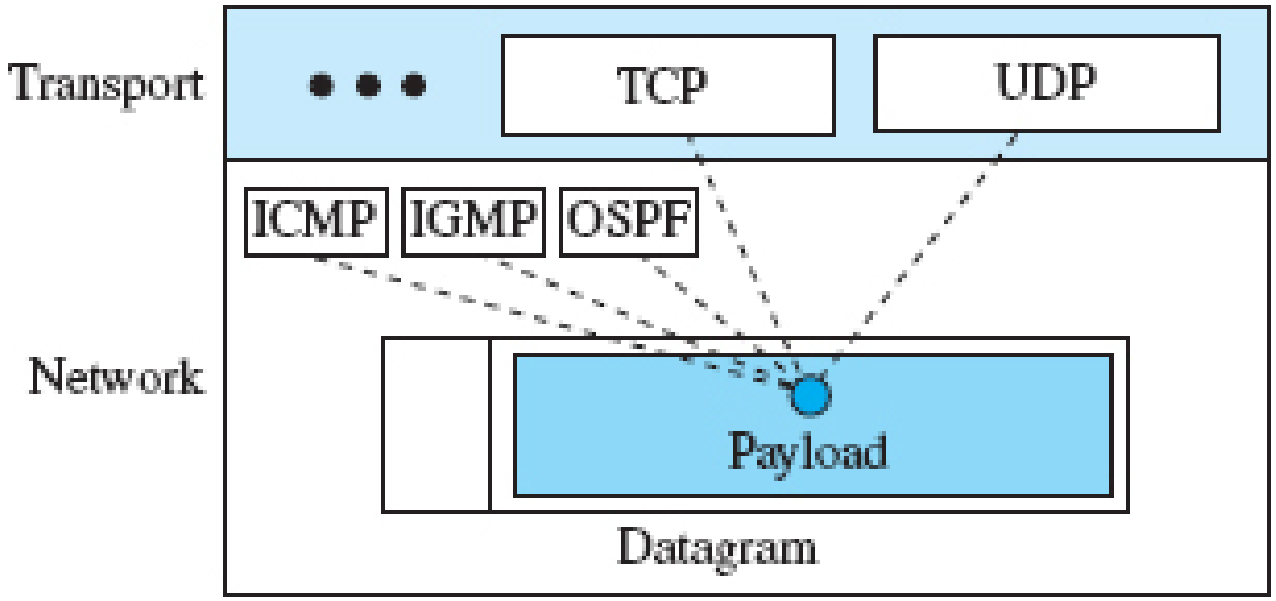
A brief description of each field is in order:

- Version number. The 4-bit version number (VER) field defines the version of the IPv4 protocol.
- Header length. The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header.
- Service type. This field was referred to as type of service (TOS), which defined how the datagram should be handled.
- Total length. This 16-bit field defines the total length (header plus data) of the IP datagram in bytes.
  - This field helps the receiving device know when the packet has completely arrived. To find the length of the data coming from the upper layer, subtract the header length from the total length. The header length can be found by multiplying the value in the HLEN field by 4.

$$\text{Length of data} = \text{total length} - (\text{HLEN}) \times 4$$

- Identification, flags, and fragmentation offset. These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry. are used for packet fragmentation and reassembly.
- Time-to-live. The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram.
- Protocol. In TCP/IP, the data section of a packet, called the payload, carries the whole packet from another protocol.
  - In other words, this field provides multiplexing at the source and demultiplexing at the destination

Figure: Multiplexing and demultiplexing using the value of the protocol field



Some protocol values

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

- Header checksum. IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission.
- Source and destination addresses. These 32-bit source and destination address fields define the IP address of the source and destination, respectively.
- Options. A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging.
- Payload. Payload, or data, is the main reason for creating a datagram. Payload is the packet coming from other protocols that use the service of IP

Example:

An IPv4 packet has arrived with the first 8 bits as $(01000010)_2$. The receiver discards the packet. Why?

Solution:

There is an error in this packet. The 4 leftmost bits $(0100)_2$ show the version, which is correct. The next 4 bits $(0010)_2$ show an invalid 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:

In an IPv4 packet, the value of HLEN is $(1000)_2$. 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:

In an IPv4 packet, the value of HLEN is 5, 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 $(0028)_{16}$ or 40 bytes, which means the packet is carrying 20 bytes of data (40 − 20).

Example:

An IPv4 packet has arrived with the first few hexadecimal digits as shown.

$$(45000028000100000102 \ldots)_{16}$$

How many hops can this packet travel before being dropped? To which upper-layer protocol do the data belong?

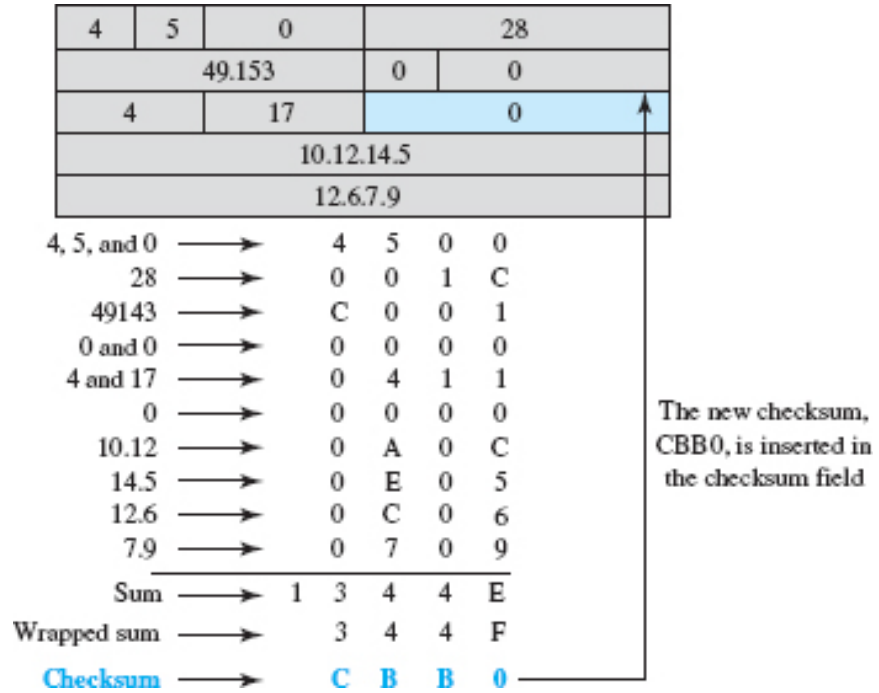
Solution:

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 $(01)_{16}$. This means the packet can travel only one hop. The protocol field is the next byte $(02)_{16}$, which means that the upper-layer protocol is IGMP.

Example:

Figure shows an example of a checksum calculation for an IPv4 header without options. The header is divided into 16-bit sections. All the sections are added, and the sum is complemented after wrapping the leftmost digit. The result is inserted in the checksum field.

Note that the calculation of wrapped sum and checksum can also be done as follows in hexadecimal:
Wrapped Sum = Sum mod FFFF
Checksum = FFFF − Wrapped Sum



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.
- For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU):

- One of the features of each format is the maximum size of the payload that can be encapsulated.
- The value of the maximum transfer unit (MTU) differs from one physical network protocol to another.

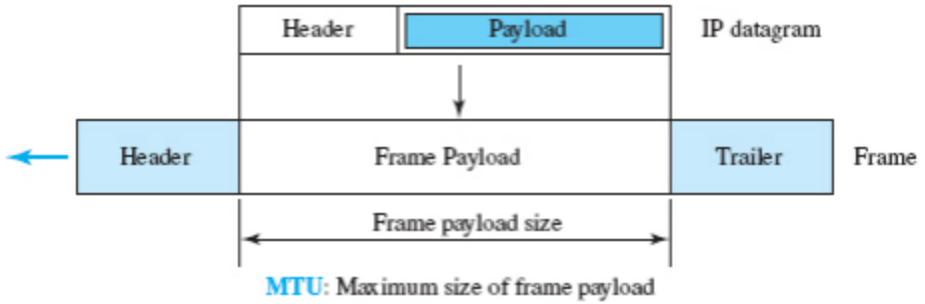


Figure : Maximum transfer unit (MTU)

- We must divide the datagram to make it possible for it to pass through these networks. This is called fragmentation.
- When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed.
- The host or router that fragments a datagram must change the values of three fields: flags, fragmentation offset, and total length.
- The rest of the fields must be copied.
- The value of the checksum must be recalculated regardless of fragmentation.

Fields Related to Fragmentation :

- Three fields in an IP datagram are related to fragmentation: identification, flags, and fragmentation offset.
- The 16-bit identification field identifies a datagram originating from the source host.
- The identification number helps the destination in reassembling the datagram.
- The 3-bit flags field defines three flags. The leftmost bit is reserved (not used). The second bit (D bit) is called the do not fragment bit. The third bit (M bit) is called the more fragment bit.
- The 13-bit fragmentation offset field shows the relative position of this fragment with respect to the whole datagram.

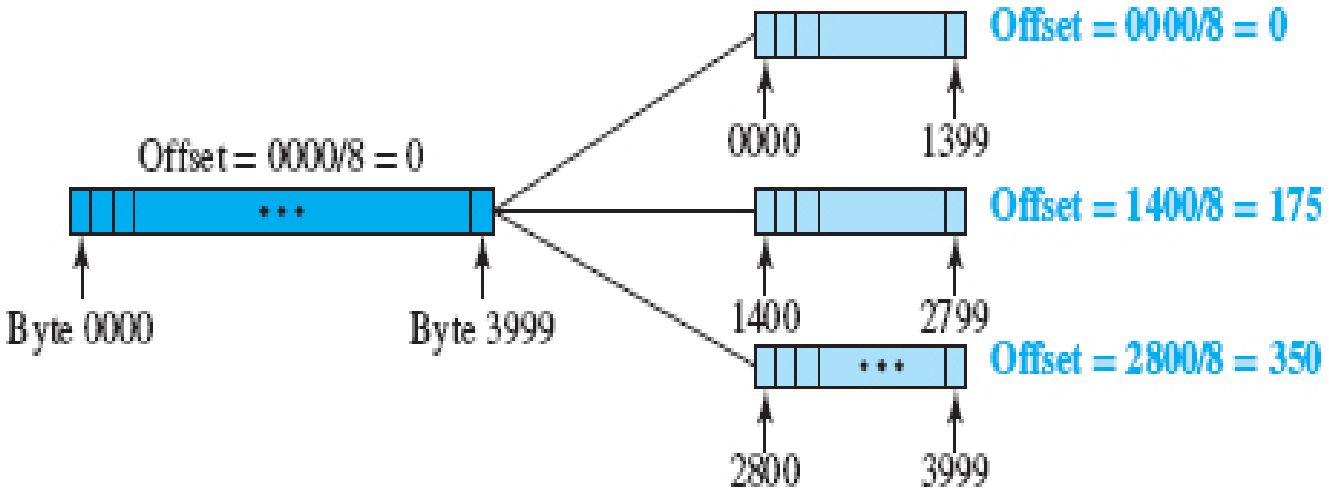
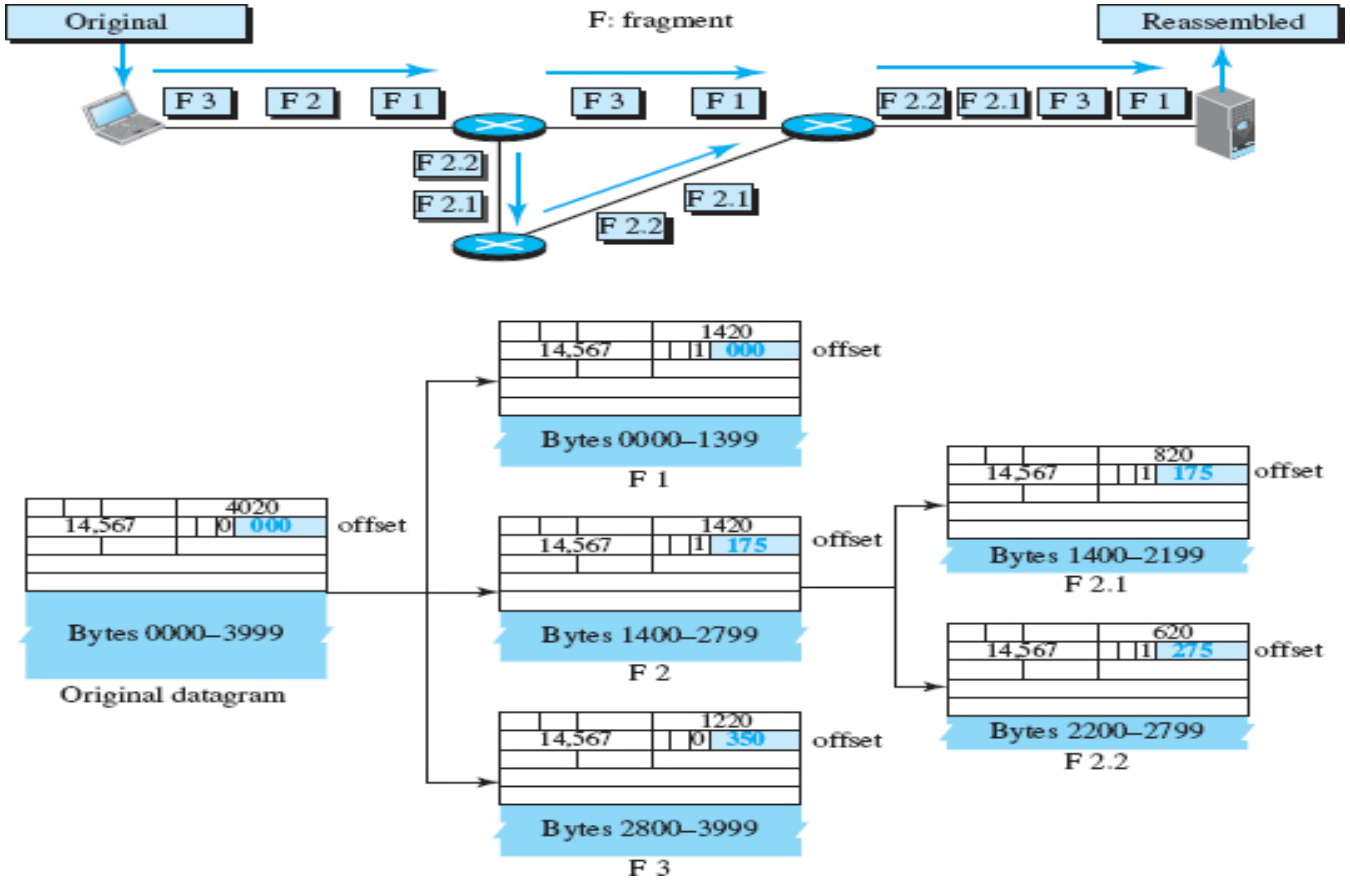


Figure : Fragmentation example

Figure : Detailed fragmentation example



- The final destination host can reassemble the original datagram from the fragments received using the following strategy:

a. The first fragment has an offset field value of zero.

b. Divide the length of the first fragment by 8. The second fragment has an offset value equal to that result.

c. Divide the total length of the first and second fragment by 8. The third fragment has an offset value equal to that result.

d. Continue the process. The last fragment has its M bit set to 0.

e. Continue the process. The last fragment has an M bit value of 0.

Example :

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 :

A packet has arrived with an M bit value of 1 and a fragmentation offset value of 0. 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 :

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 :

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 are the numbers 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.

Routing Algorithms

Distance-Vector Routing

- Goal : find the best route.
- In distance-vector routing, a router continuously tells all its neighbors about what it knows about the whole internet.
- Two important topics:
  - the Bellman Ford equation and
  - the concept of distance vectors.

Bellman-Ford Equation

- The heart of distance-vector routing is the famous Bellman-Ford equation.

- This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, …).
  - when the costs between the source and the intermediary nodes and the least costs between the intermediary nodes and the destination are given.

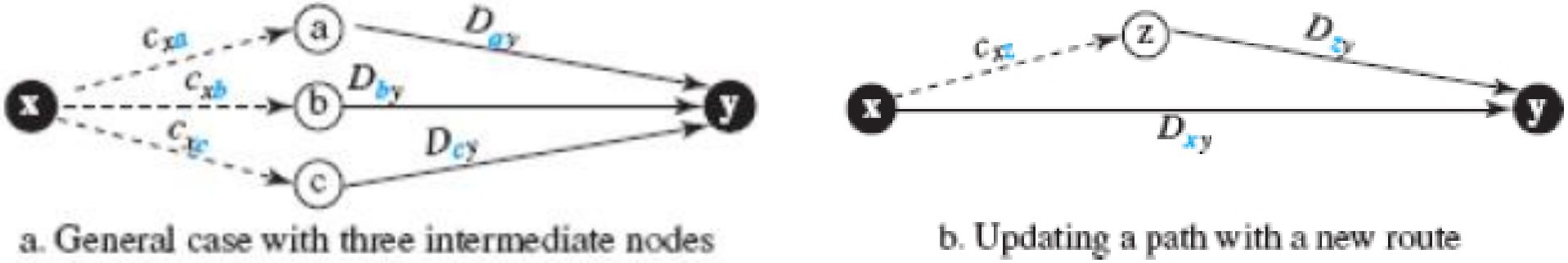
- The following shows the general case in which Dij is the shortest distance and cij is the cost between nodes i and j.

$$D_{xy} = min \{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

- In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as z, if the latter is shorter. In this case, the equation becomes simpler:

$$D_{xy} = min\{D_{xy}, (c_{xz} + D_{zy})\}$$

- Figure shows the idea graphically for both cases.



a. General case with three intermediate nodes

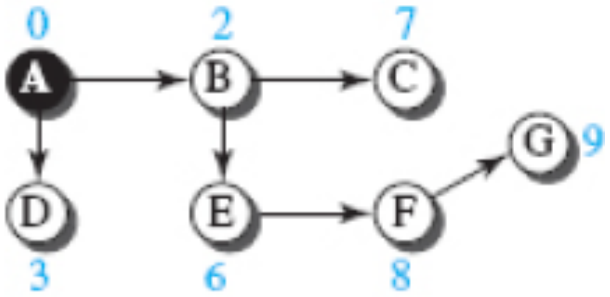
b. Updating a path with a new route

we can think of (a → y), (b → y), and (c → y) as previously established least-cost paths and (x → y) as the new least-cost path.

- We can say that the Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths.

Distance Vectors :

- A least-cost tree is a combination of least-cost paths from the root of the tree to all destinations.
- Distance-vector routing unglues these paths and creates a distance vector, a one-dimensional array to represent the tree.



Tree for a node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

Distance vector for a node A

Figure : The distance vector corresponding to a tree

Note:

- The name of the distance vector defines the root,
- The indexes define the destinations, and
- The value of each cell defines the least cost from the root to the destination.

Figure : First distance vector for an internet

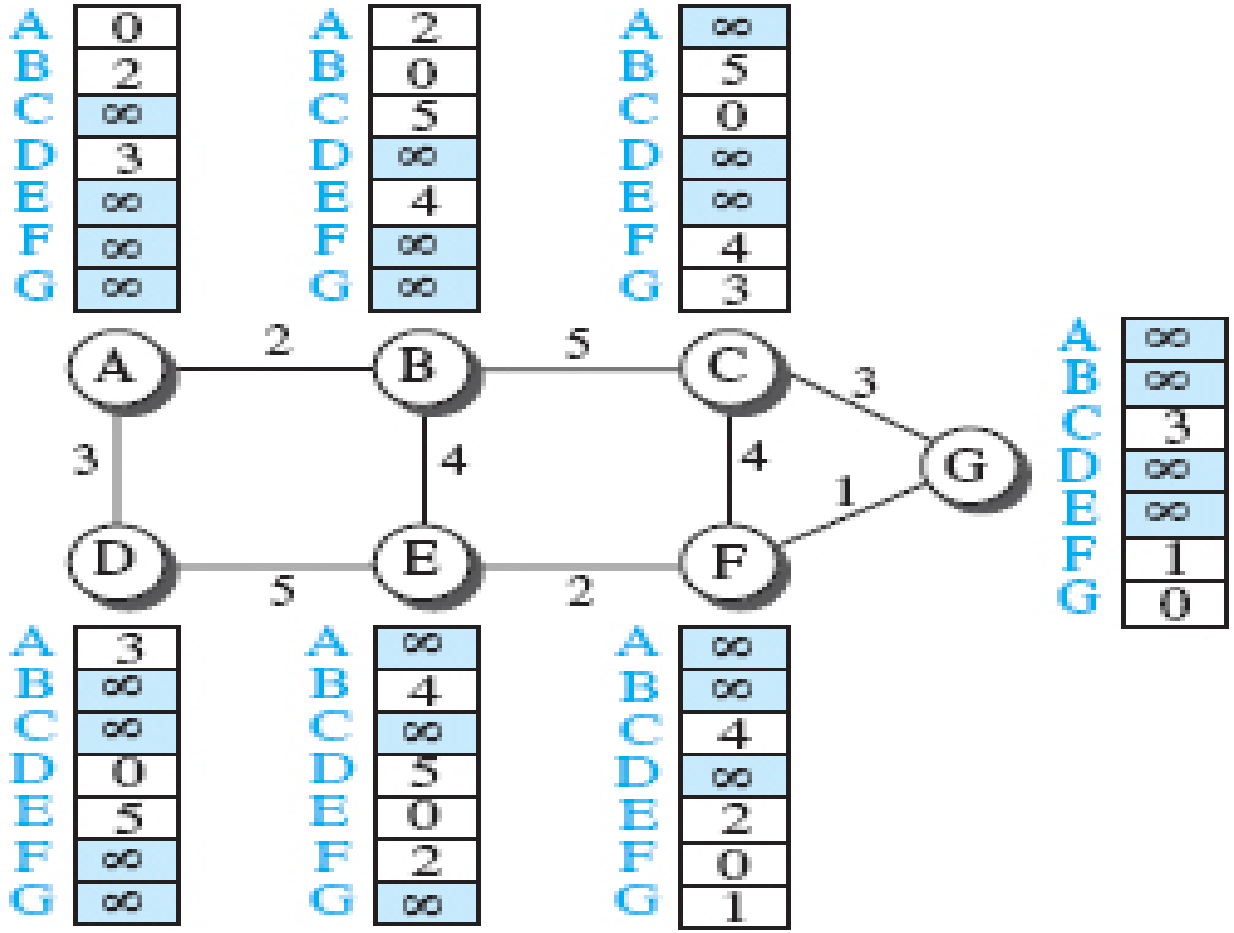
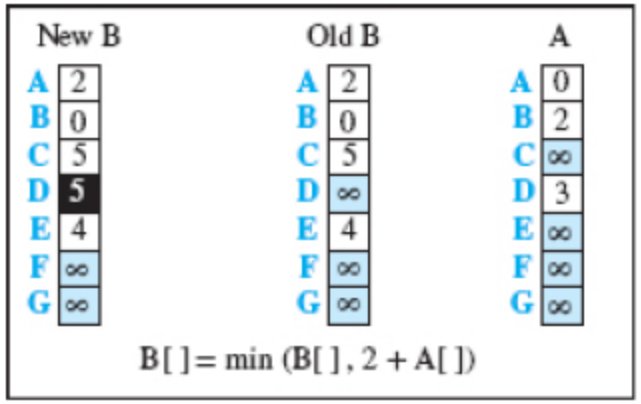
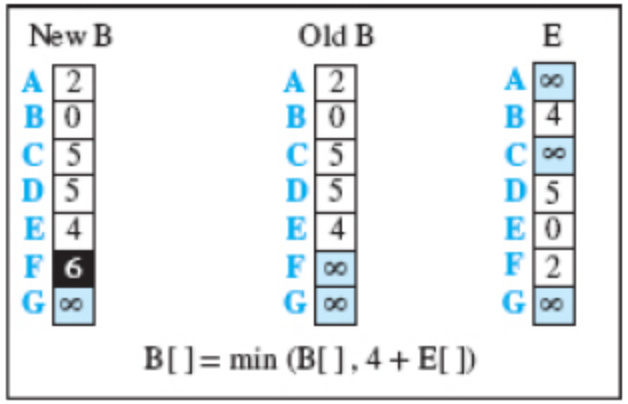
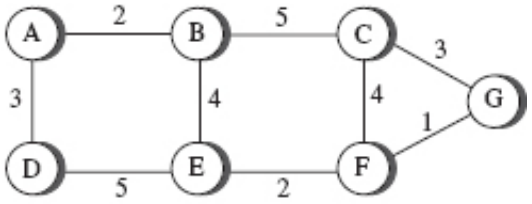


Figure : Updating distance vectors







b. The weighted graph

- In the first event, node A has sent its vector to node B. Node B updates its vector using the cost $c_{BA} = 2$.
- In the second event, node E has sent its vector to node B. Node B updates its vector using the cost $c_{EA} = 4$.

Distance-Vector Routing Algorithm :

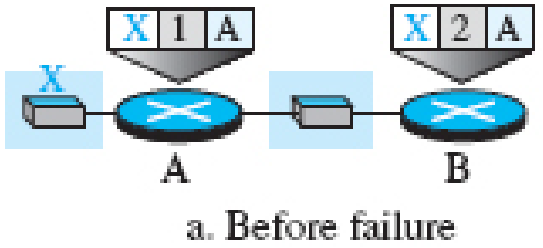
```
1  Distance_Vector_Routing ( )
2  {
3     // Initialize (create initial vectors for the node)
4     D[myself ] = 0
5     for (y = 1 to N)
6     {
7        if (y is a neighbor)
8           D[y] = c[myself][y]
9        else
10          D[y] = ∞
11    }
12    send vector {D[1], D[2], ..., D[N]} to all neighbors
13    // Update (improve the vector with the vector received from a neighbor)
14    repeat (forever)
15    {
16       wait (for a vector D_w from a neighbor w or any change in the link)
17       for (y = 1 to N)
18       {
19          D[y] = min [D[y], (c[myself ][w] + D_w[y])]  // Bellman- Ford equation
20       }
21       if (any change in the vector)
22          send vector {D[1], D[2], ..., D[N]} to all neighbors
23    }
24 }
```

Count to Infinity:

- For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some time.
- *It sometimes takes several updates before the cost for a* broken link is recorded as infinity by all routers.
- Example of count to infinity - ***Two-Node Loop***

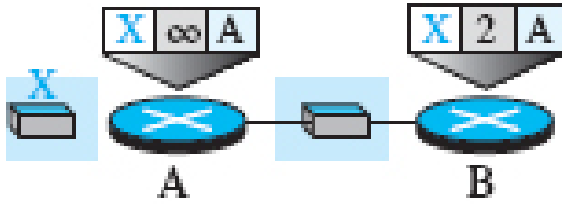
Count to infinity - *Two-Node Loop*

Both nodes A and B know how to reach node X



a. Before failure

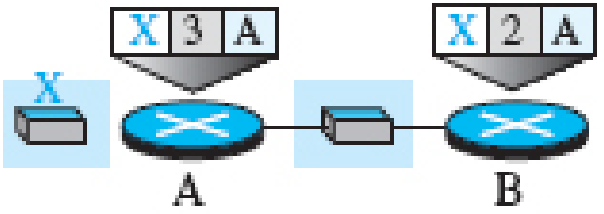
Link between A and X fails. Node A changes its table.



b. After link failure

If A can send its table to B immediately, everything is fine.

- However, the system becomes unstable if B sends its forwarding table to A before receiving A's forwarding table.
- Node A receives the update and, assuming that B has found a way to reach X, immediately updates its forwarding table.



c. After A is updated by B

Count to infinity - *Two-Node Loop*

d. After B is updated by A

e. Final result

Now A sends its new update to B.

- Now B thinks that something has been changed around A and updates its forwarding table.
- The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached.

During this time the system is not stable.

- Node A thinks that the route to X is via B; node B thinks that the route to X is via A.
- If A receives a packet destined for X, the packet goes to B and then comes back to A.
- Similarly, if B receives a packet destined for X, it goes to A and comes back to B.
- Packets bounce between A and B, creating a two-node loop problem.

Solution to instability - *split horizon*

Instead of flooding the table through each interface, each node sends only part of its table through each interface.

If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A

- the information has come from A (A already knows).
  - Taking information from node A, modifying it, and sending it back to node A is what creates the confusion.
- Node A keeps the value of infinity as the distance to X - Later, when node A sends its forwarding table to B, node B also corrects its forwarding table.

The system becomes stable after the first update: Both nodes A and B know that X is not reachable.

Poisoned Reverse

Split horizon problem:

Normally, the corresponding protocol uses a timer

If there is no news about a route, the node deletes the route from its table

- When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the source of information was A) or because B has not received any news about X recently

Solution: poisoned reverse strategy

- B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: “Do not use this value; what I know about this route comes from you.”

The two-node instability can be avoided using split horizon combined with poisoned reverse.

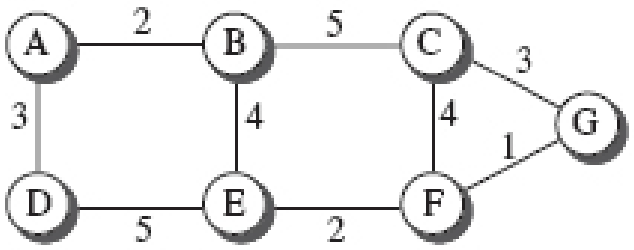
If the instability is between three nodes , it cannot be guaranteed

Link-State Routing

- A routing algorithm that directly follows link state for creating least cost trees and forwarding tables is link-state (LS) routing.
- In this algorithm the cost associated with an edge defines the state of the link.
- Links with lower costs are preferred to links with higher costs.
- if the cost of a link is infinity, it means that the link does not exist or has been broken.

Link-State Database (LSDB) :

- The collection of states for all links is called the link-state database (LSDB).
- There is only one LSDB for the whole internet, each node needs to have a duplicate of it to be able to create the least-cost tree.
- The LSDB can be represented as a two-dimensional array (matrix) in which the value of each cell defines the cost of the corresponding link.



a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Figure : Example of a link-state database

1. “How can each node create this LSDB that contains information about the whole internet?”

- done by a process called Flooding.
- Each node can send some greeting messages to all its immediate neighbors (those nodes to which it is connected directly) to collect two pieces of information for each neighboring node:
  - Identity of the node and
  - Cost of the link.
- The combination of these two pieces of information is called the *LS packet (LSP).*
- *LSP is sent out of each* interface

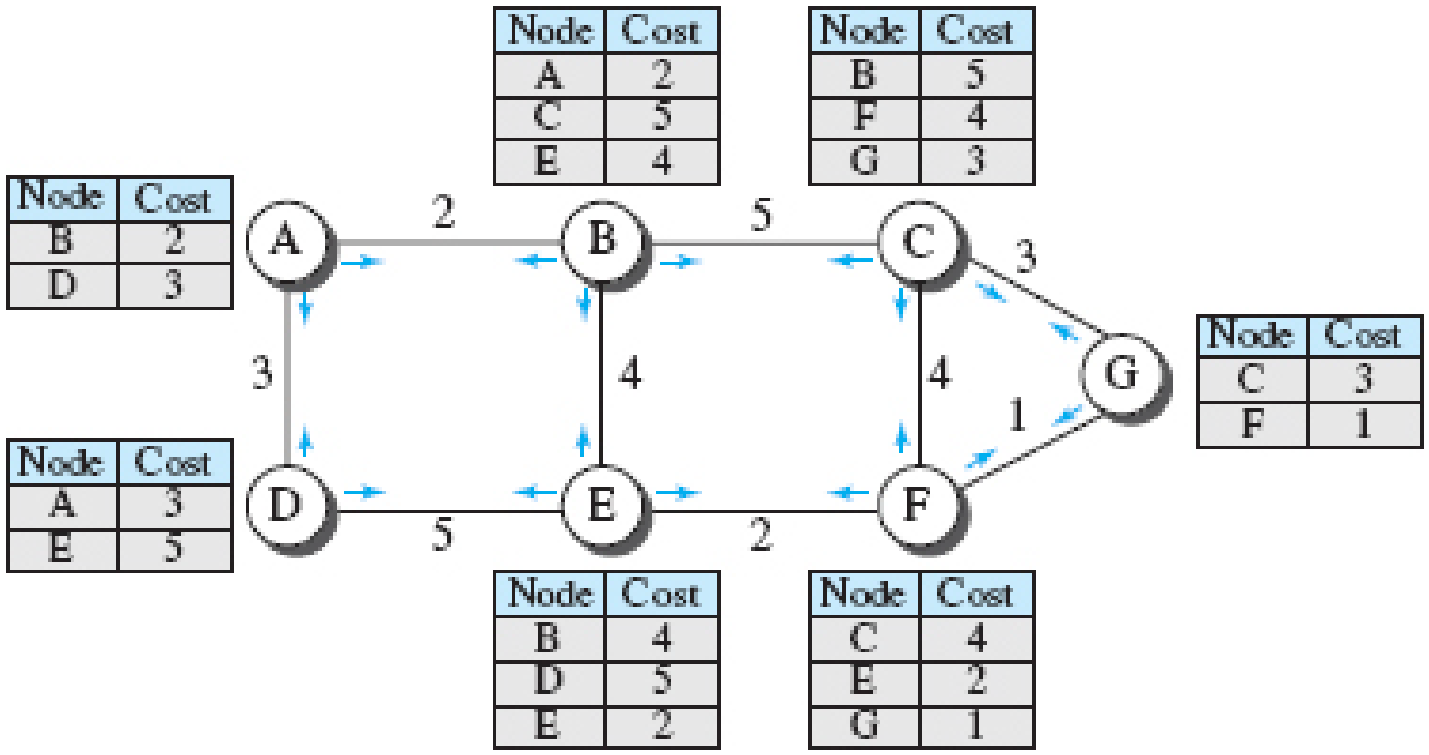


Figure : LSPs created and sent out by each node to build the LSDB

Formation of Least-Cost Trees :

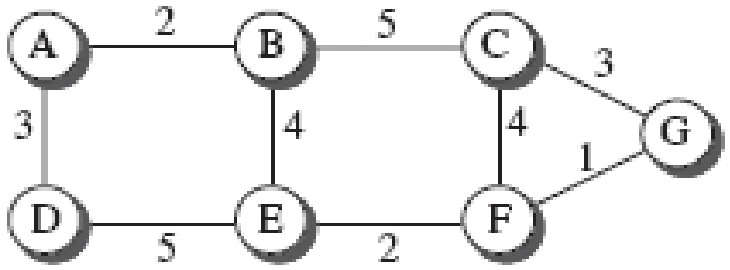
- To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous Dijkstra's algorithm. This iterative algorithm uses the following steps:

1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.
2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.
3. The node repeats step 2 until all nodes are added to the tree.

Dijkstra's algorithm

```
1   Dijkstra's Algorithm ( )
2   {
3     // Initialization
4     Tree = {root}          // Tree is made only of the root
5     for (y = 1 to N)       // N is the number of nodes
6     {
7         if (y is the root)
8             D [y] = 0      // D [y] is shortest distance from root to node y
9         else if (y is a neighbor)
10            D [y] = c[root][y]      // c [x] [y] is cost between nodes x and y in LSDB
11        else
12            D [y] = ∞
13    }
14    // Calculation
15    repeat
16    {
17        find a node w, with D [w ] minimum among all nodes not in the Tree
18        Tree = Tree ∪ {w}      // Add w to tree
19        // Update distances for all neighbor of w
20        for (every node x, which is neighbor of w and not in the Tree)
21        {
22            D[x] = min{D[x], (D[w] + c[w][x])}
23        }
24    } until (all nodes included in the Tree)
25  }
```

Formation of Least-Cost Trees

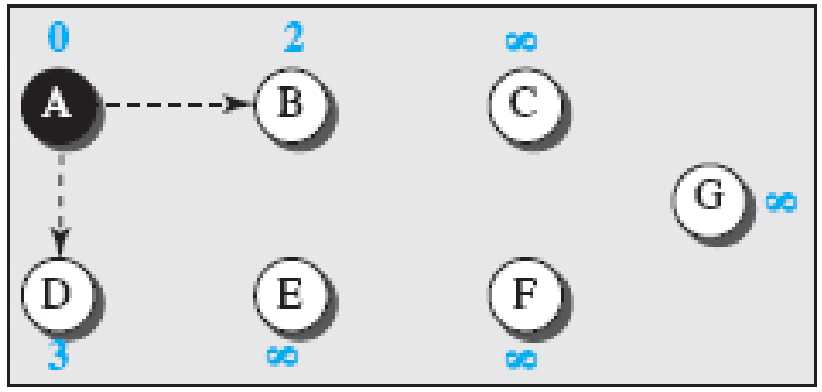


a. The weighted graph

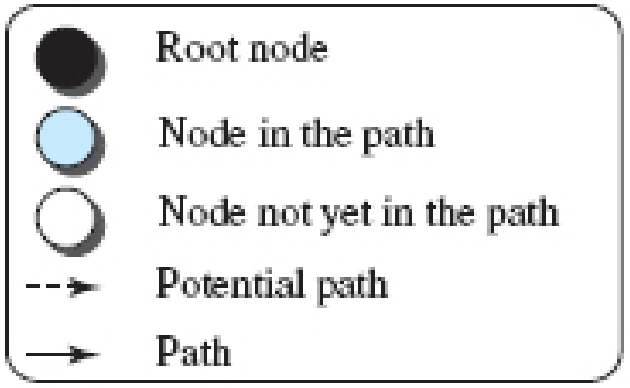
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

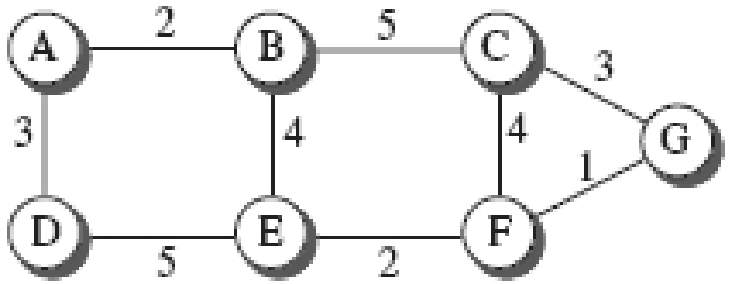
Initialization



Legend



Formation of Least-Cost Trees

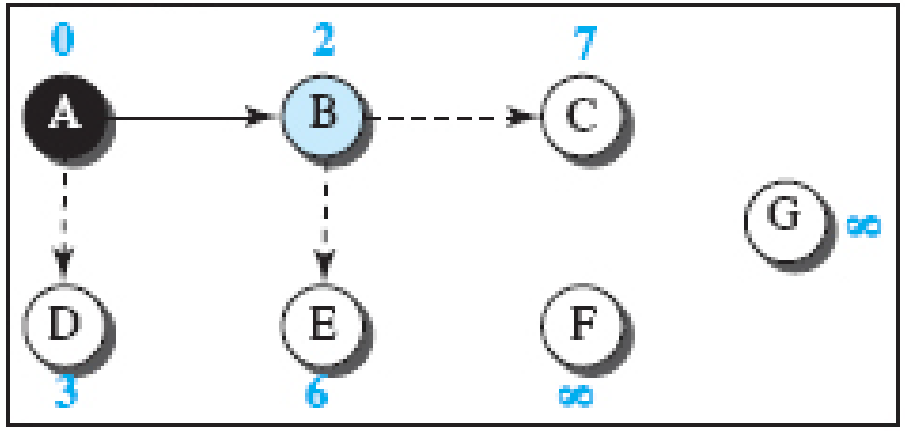


a. The weighted graph

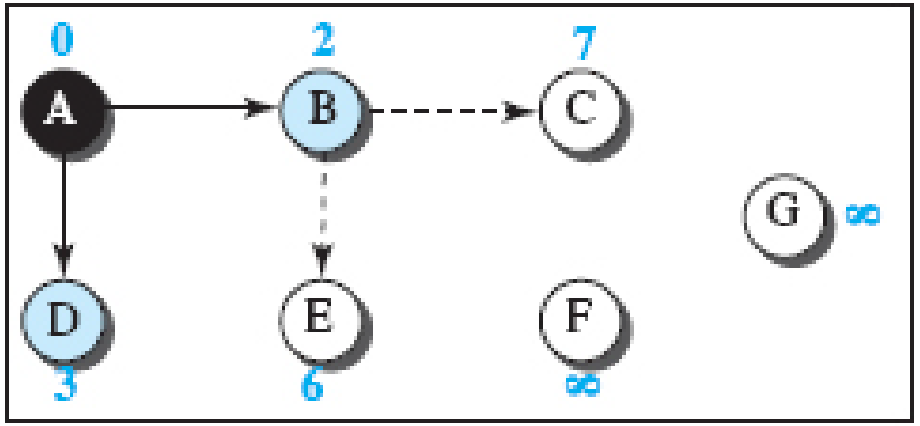
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

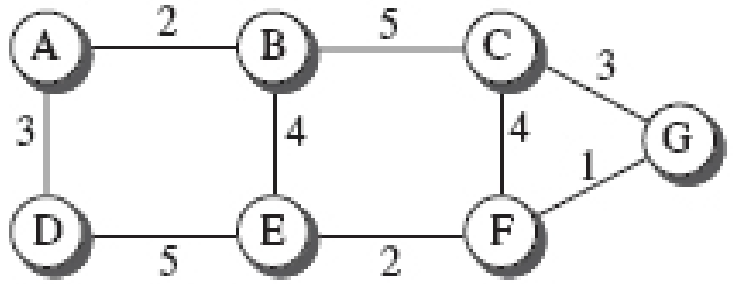
Iteration 1



Iteration 2



Formation of Least-Cost Trees

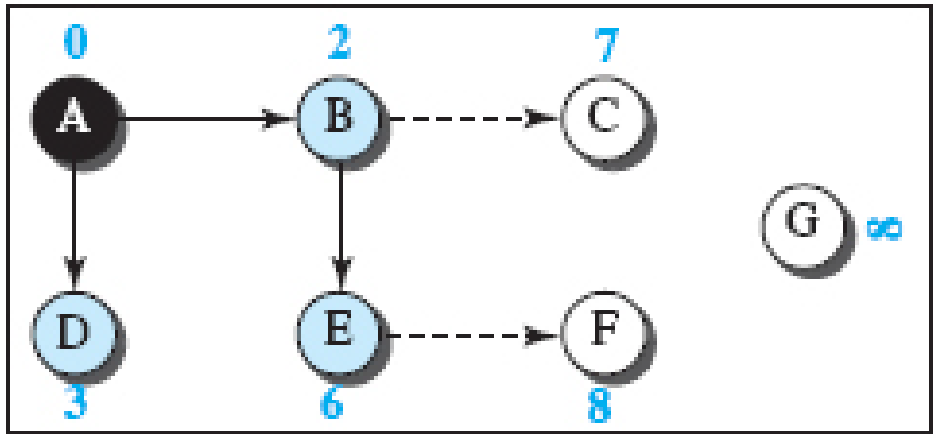


a. The weighted graph

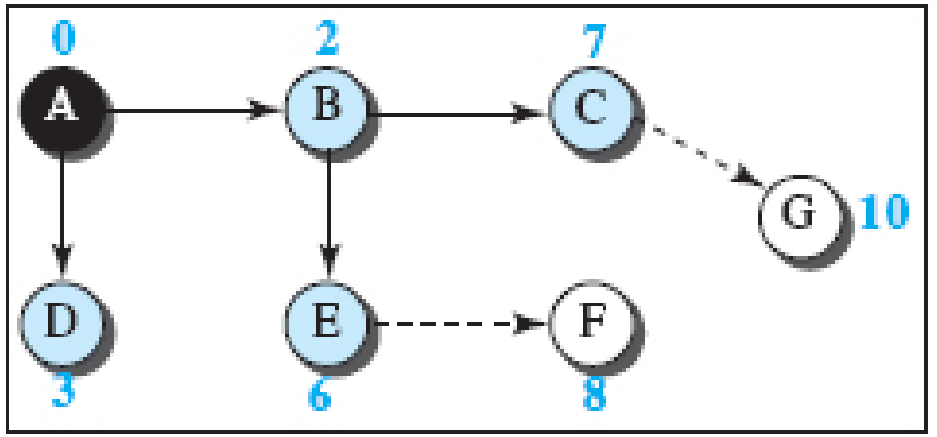
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

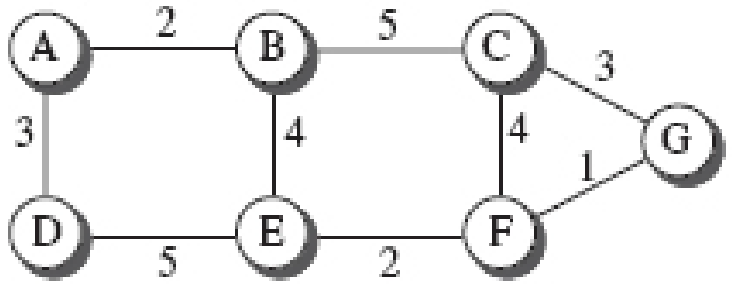
Iteration 3



Iteration 4



Formation of Least-Cost Trees

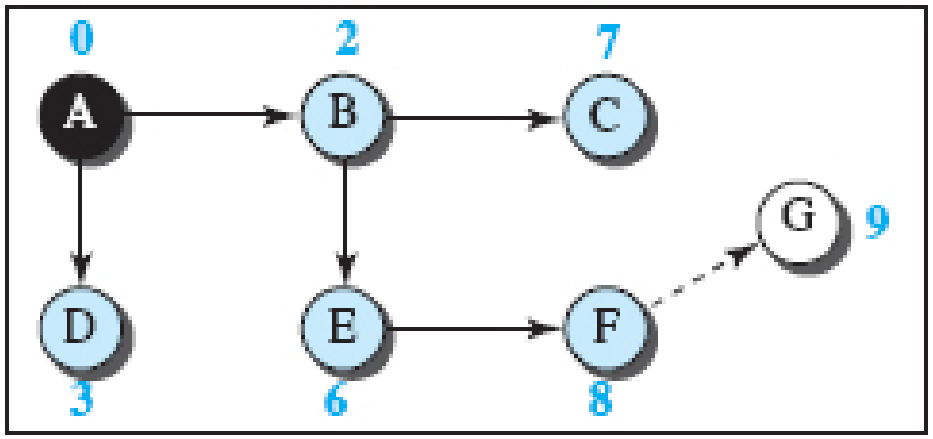


a. The weighted graph

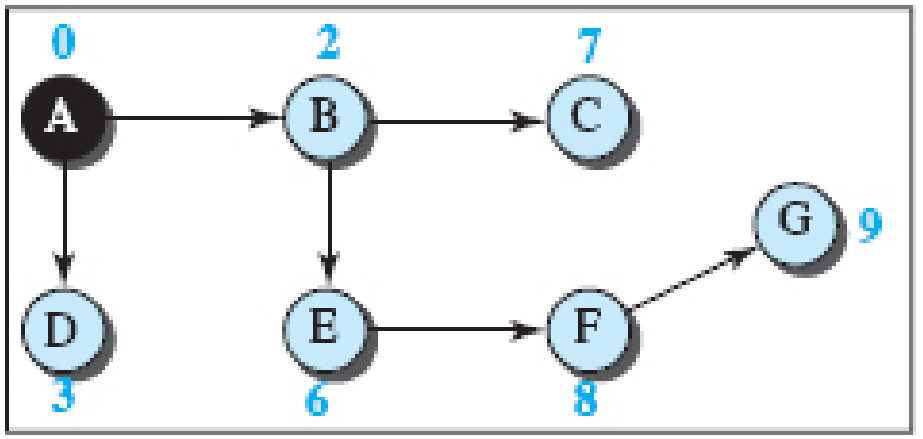
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Iteration 5



Iteration 6



Unicast routing protocols

A protocol is more than an algorithm

Protocol needs to define

- its domain of operation
- the messages exchanged
- Communication between routers
- interaction with protocols in other domains

Three common protocols used in the Internet:

- Routing Information Protocol (RIP), based on the distance-vector algorithm
- Open Shortest Path First (OSPF), based on the link-state algorithm
- Border Gateway Protocol (BGP), based on the path-vector algorithm

Internet Structure :

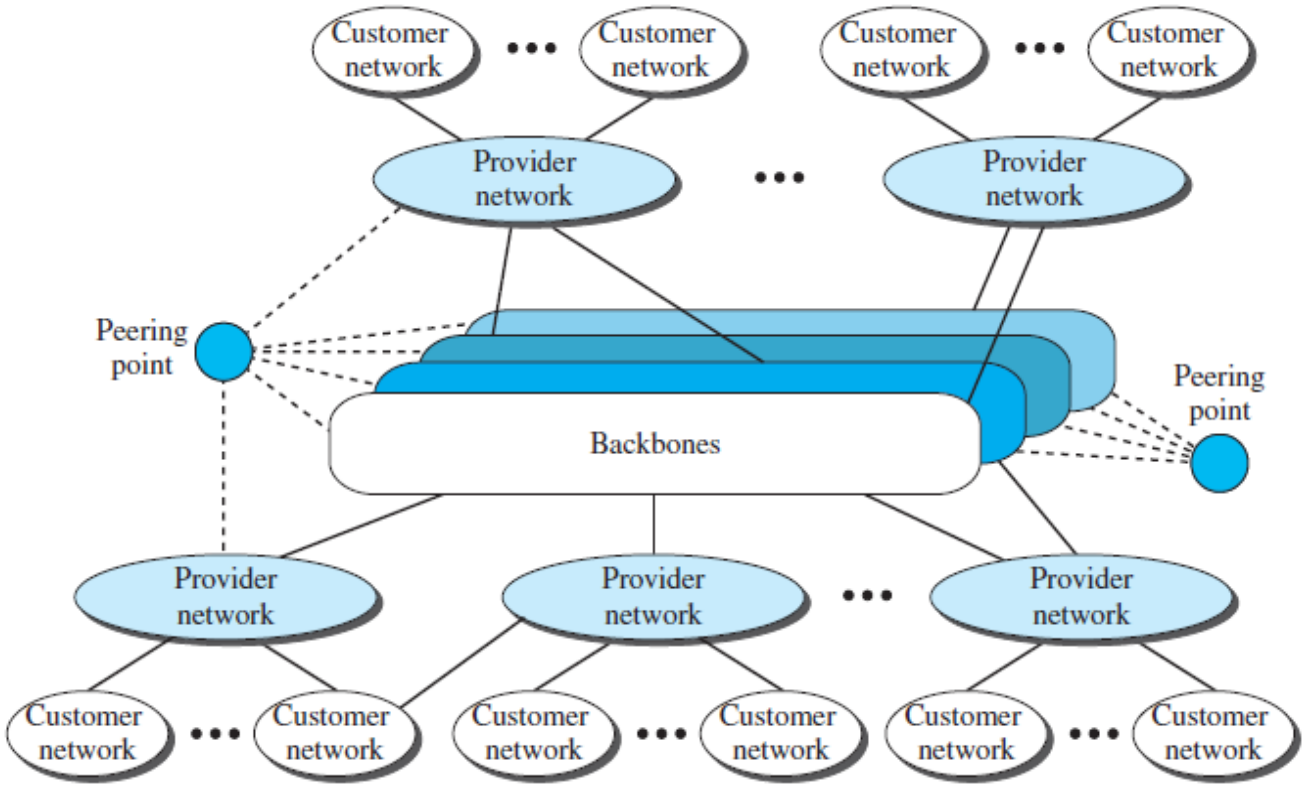


Figure : Internet structure

Hierarchical Routing :

- The Internet today is made up of a huge number of networks and routers that connect them.
- Internet cannot be done using one single protocol for two reasons:
  - A scalability problem : The size of the forwarding tables becomes huge.
  - An administrative issue : The administrator needs to have control in its system.
- Hierarchical routing means considering each ISP as an **autonomous system (AS).**
- The routing protocol run in each AS is referred to as intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP).
- The global routing protocol is referred to as inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP).
- Presently, the two common intradomain routing protocols are RIP and OSPF; the only interdomain routing protocol is BGP. The situation may change when we move to IPv6.

Autonomous Systems :

- Each ISP is an autonomous system when it comes to managing networks and routers under its control
- Although we may have small, medium-size, and large ASs, each AS is given an autonomous number (ASN) by the Internet Corporation for Assigned Names and Numbers (ICANN)
- Each ASN is a 16-bit unsigned integer that uniquely defines an AS
- The autonomous systems, however, are not categorized according to their size; they are categorized according to the way they are connected to other Ass

Types of Autonomous Systems

- Stub AS- has only one connection to another AS
  - The data traffic can be either initiated or terminated in a stub AS
  - the data cannot pass through it
  - A good example of a stub AS is the costumer network, which is either the source or the sink of data.
- Multihomed AS - can have more than one connection to other ASs, but it does not allow data traffic to pass through it
  - A good example of such an AS is some of the costumer ASs that may use the services of more than one provider network, but their policy does not allow data to be passed through them.
- Transient AS - connected to more than one other AS and also allows the traffic to pass through
  - The provider networks and the backbone are good examples of transient ASs

Routing Information Protocol (RIP)

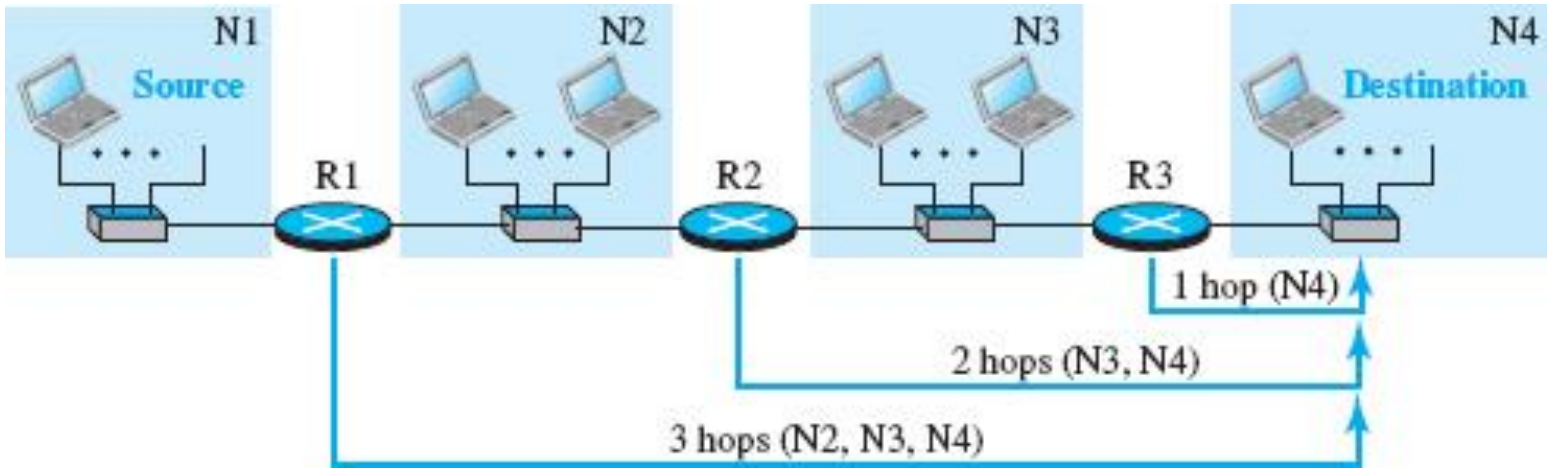
- Intradomain routing protocols based on the distance-vector routing algorithm
- RIP concept:
  - Hop count
  - Forwarding tables
  - Implementation – RIP messages, algorithm, timers
  - Performance

RIP – Hop count

- A router in an AS needs to know how to forward a packet to different networks (subnets) in an AS so RIP routers advertise the cost of reaching different networks instead of reaching other nodes in a theoretical graph
  - The cost is defined between a router and the network in which the destination host is located
- To make the implementation of the cost simpler (independent from performance factors of the routers and links, such as delay, and bandwidth), the cost is defined as the number of hops, which means the number of networks (subnets) a packet needs to travel through from the source router to the final destination host
  - The network in which the source host is connected is not counted in this calculation because the source host does not use a forwarding table
  - the packet is delivered to the default router.

RIP – Hop count

- Hop count advertised by three routers from a source host to a destination host



- In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity (no connection)
- For this reason, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops

RIP- Forwarding Tables

- Routers in an autonomous system need to keep forwarding tables to forward packets to their destination networks
- A forwarding table in RIP is a three column table
  - the address of the destination network
  - the address of the next router to which the packet should be forwarded
  - the cost (the number of hops) to reach the destination network
- the first and third columns together convey the same information as does a distance vector, but the cost shows the number of hops to the destination networks.

Forwarding table for R1

Destination network	Next router	Cost in hops
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Forwarding table for R2

Destination network	Next router	Cost in hops
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Forwarding table for R3

Destination network	Next router	Cost in hops
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

RIP- Forwarding Tables

Although a forwarding table in RIP defines only the next router in the second column, it gives the information about the whole least-cost tree

For example, R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 is R3; R3 defines that there is no next router for this path.

- The tree is then R1 → R2 → R3 → N4.

What the use of the third column?

- The third column is not needed for forwarding the packet, but it is needed for updating the forwarding table when there is a change in the route

RIP Implementation

uses the service of User Datagram Protocol (UDP) on the well-known port number 520

In BSD, RIP is a daemon process (a process running at the background), named *routed* (abbreviation for *route daemon and pronounced route-dee)*

- Although RIP is a routing protocol to help IP route its datagrams through the AS, the RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams.
  - RIP runs at the application layer, but creates forwarding tables for IP at the network later

RIP has gone through two versions: RIP-1 and RIP-2 (backward-compatible)

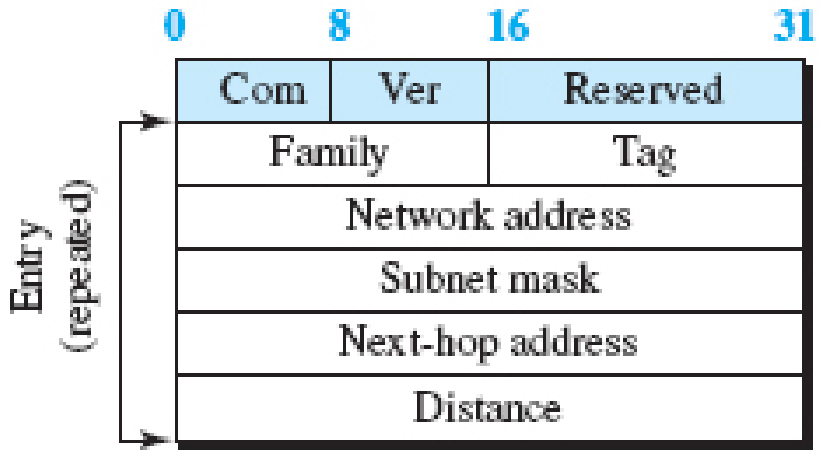
RIP-2 allows the use of more information in the RIP messages that were set to 0 in the first version.

RIP Messages

Two RIP processes, a client and a server need to exchange messages

Part of the message -call an entry

- Each entry carries the information related to one line in the forwarding table of the router that sends the message.



Fields

Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

RIP Messages

RIP has two types of messages: request and response

A request message

- Is sent by a router that has just come up or by a router that has some time-out entries
- can ask about specific entries or all entries

A response (or update) message

- Solicited - sent only in answer to a request message
  - contains information about the destination specified in the corresponding request message
- Unsolicited- sent periodically, every 30 s or when there is a change in the forwarding table.

RIP Algorithm

Same as the distance-vector routing algorithm

some changes to be made to enable a router to update its forwarding table:

- Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message.
- The receiver adds one hop to each cost and changes the next router field to the address of the sending router
  - We call each route in the modified forwarding table the *received route and each route in the old forwarding* table the *old route*
  - *The received router selects the old routes as the new* ones three cases
- The new forwarding table needs to be sorted according to the destination route (mostly using the longest prefix first).

RIP Algorithm - three cases

Cases where the received router does not selects the old routes as the new ones

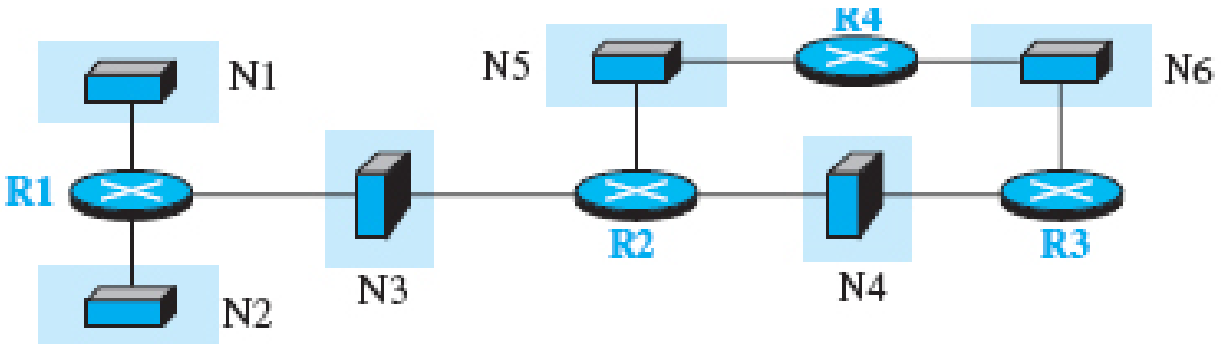
If the received route does not exist in the old forwarding table, it should be added to the route.

If the than cost of the received route is lower the cost of the old one, the received route should be selected as the new one.

If the cost of the received route is higher than the cost of the old one, but the value of the next router is the same in both routes, the received route should be selected as the new one.

- the route was actually advertised by the same router in the past, but now the situation has been changed
  - For example, suppose a neighbor has previously advertised a route to a destination with cost 3, but now there is no path between this neighbor and that destination.
    - The neighbor advertises this destination with cost value infinity (16 in RIP).
    - The receiving router must not ignore this value even though its old route has a lower cost to the same destination.

Realistic example of the operation of RIP



All forwarding tables after all routers have been booted

R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2

Des.	N. R.	Cost
N3	—	1
N4	—	1
N5	—	1

R3

Des.	N. R.	Cost
N4	—	1
N6	—	1

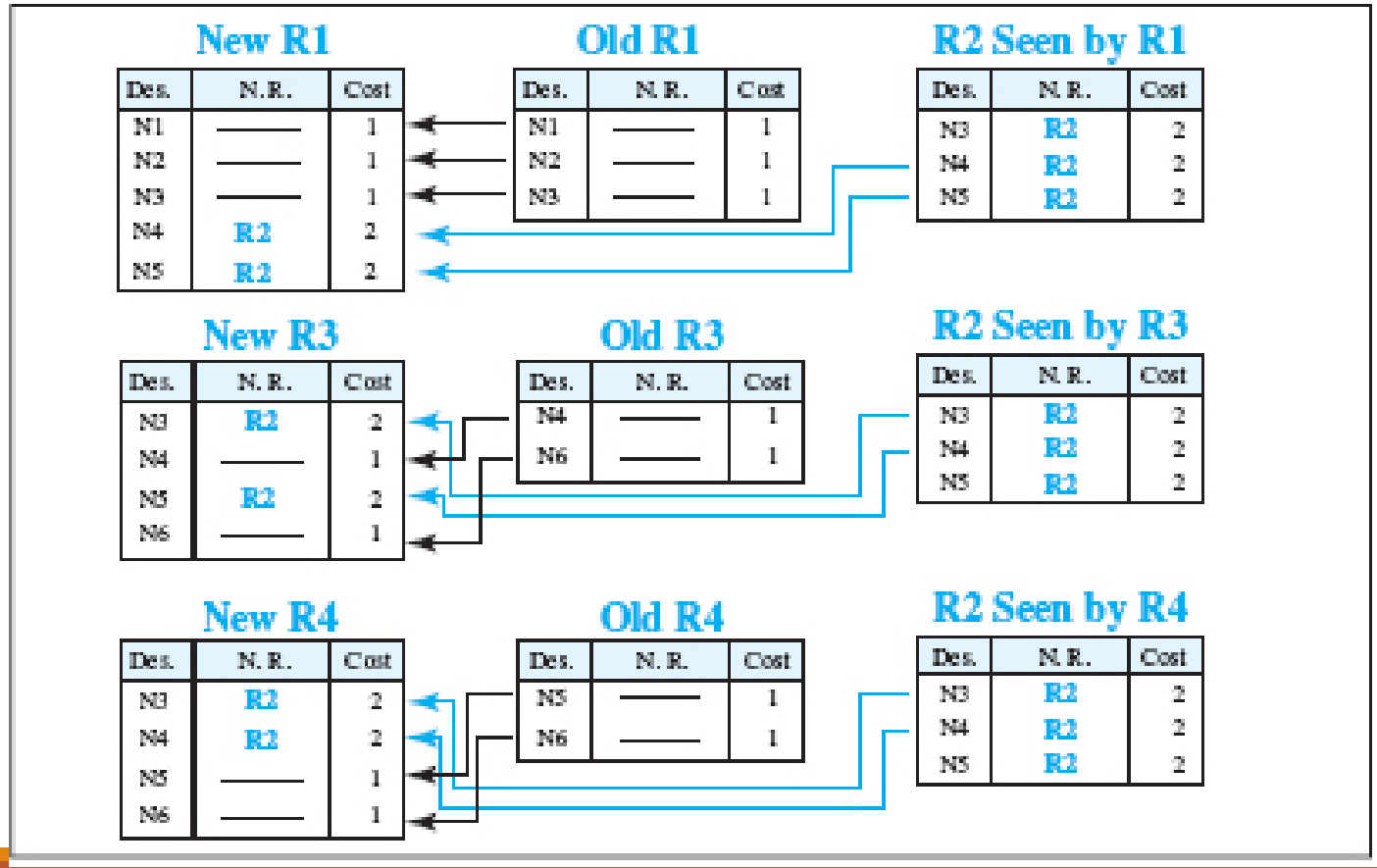
R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

Forwarding tables after all routers booted

Realistic example of the operation of RIP

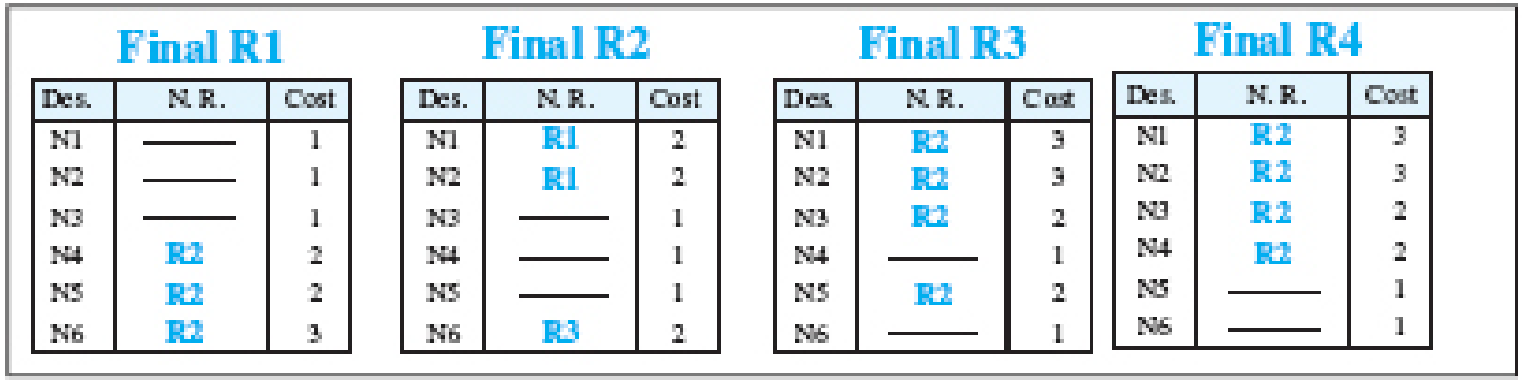
Then it shows changes in some tables when some update messages have been exchanged



Changes in the forwarding tables of R1, R3, and R4 after they receive a copy of R2's table

Realistic example of the operation of RIP

The stabilized forwarding tables when there is no more change



Final R1

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1
N4	R2	2
N5	R2	2
N6	R2	3

Final R2

Des.	N. R.	Cost
N1	R1	2
N2	R1	2
N3	———	1
N4	———	1
N5	———	1
N6	R3	2

Final R3

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	———	1
N5	R2	2
N6	———	1

Final R4

Des.	N. R.	Cost
N1	R2	3
N2	R2	3
N3	R2	2
N4	R2	2
N5	———	1
N6	———	1

Forwarding tables for all routers after they have been stablized

Timers in RIP

RIP uses three timers to support its operation

Periodic timer controls the advertising of regular update messages

- Each router has one periodic timer that is randomly set to a number between 25 and 35 s (to prevent all routers sending their messages at the same time and creating excess traffic).
- The timer counts down; when zero is reached, the update message is sent, and the timer is randomly set once again

The expiration timer governs the validity of a route

- When a router receives update information for a route, the expiration timer is set to 180 s for that particular route.
- Every time a new update for the route is received, the timer is reset.
- If there is a problem on an internet and no update is received within the allotted 180 s, the route is considered expired and the hop count of the route is set to 16, which means the destination is unreachable.
- Every route has its own expiration timer.

Timers in RIP

Garbage collection timer is used to purge a route from the forwarding table.

- When the information about a route becomes invalid, the router does not immediately purge that route from its table.
- Instead, it continues to advertise the route with a metric value of 16.
- At the same time, a garbage collection timer is set to 120 s for that route.
- When the count reaches zero, the route is purged from the table.
- This timer allows neighbors to become aware of the invalidity of a route prior to purging.

Performance of RIP

Update messages – have a very simple format and are sent only to neighbors (local)

- They do not normally create traffic because the routers try to avoid sending them at the same time.

Convergence of forwarding tables - the distance-vector algorithm can converge slowly if the domain is large, but, because RIP allows only 15 hops in a domain (16 is considered as infinity), there is normally no problem in convergence

- The only problems that may slow down convergence are count-to-infinity and loops created in the domain;
  - Solution - use of **poison-reverse and split-horizon strategies added to the RIP** extension

Robustness - In distance-vector routing, each router sends what it knows about the whole domain to its neighbors

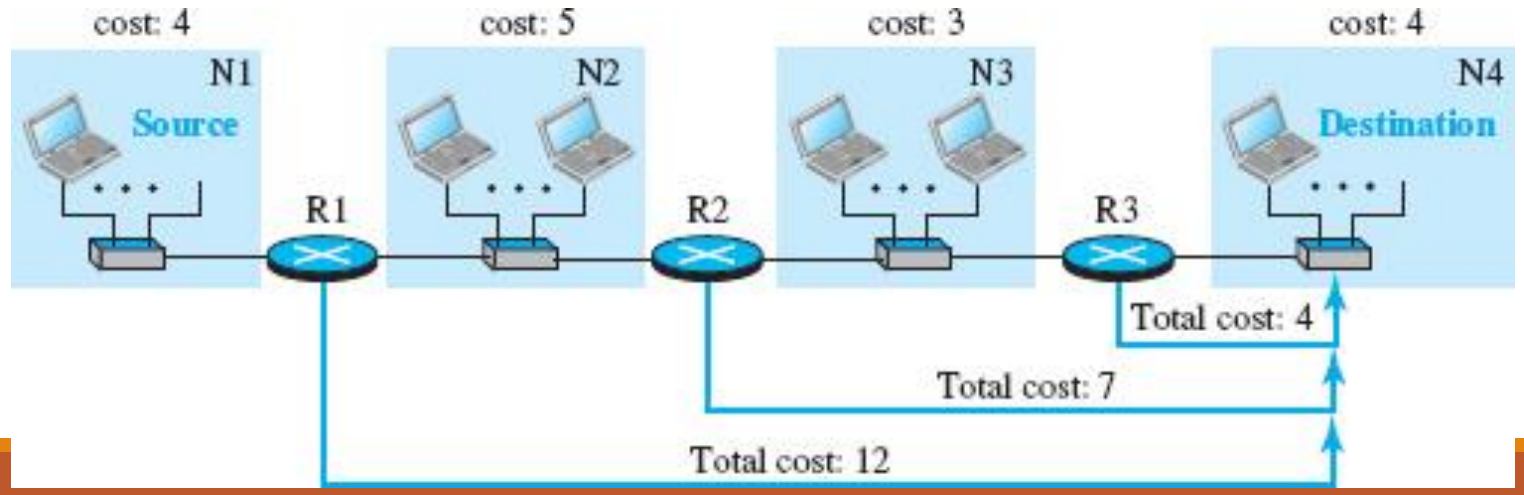
- Calculation of the forwarding table depends on information received from immediate neighbors, which in turn receive their information from their own neighbors
- If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.

Open Shortest Path First (OSPF)

- **An intradomain routing protocol** like RIP, but it is based on the link-state routing protocol
- OSPF is an *open protocol, which means that the* specification is a public document.
- OSPF concept:
  - Metric
  - Forwarding tables
  - Areas
  - Link state advertisement
  - Implementation – OSPF messages, authentication, algorithm
  - Performance

OSPF –Metric

- Cost of reaching a destination from the host is calculated from the source router to the destination network
- Each link (network) can be assigned a weight based on the throughput, round-trip time, reliability, and so on.
  - An administration can also decide to use the hop count as the cost
- An interesting point about the cost in OSPF is that different service types (TOSs) can have different weights as the cost.



Forwarding Tables

- Each OSPF router can create a forwarding table after finding the shortest path tree between itself and the destination using Dijkstra's algorithm,
- If we use the hop count for OSPF, the tables will be exactly the same as RIP because both protocols use the shortest-path trees to define the best route from a source to a destination.

Forwarding table for R1

Destination network	Next router	Cost
N1	——	4
N2	——	5
N3	R2	8
N4	R2	12

Forwarding table for R2

Destination network	Next router	Cost
N1	R1	9
N2	——	5
N3	——	3
N4	R3	7

Forwarding table for R3

Destination network	Next router	Cost
N1	R2	12
N2	R2	8
N3	——	3
N4	——	4

Areas

- RIP is normally used in small ASs, OSPF was designed to be able to handle routing in a small or large autonomous system.
- The formation of shortest-path trees in OSPF requires that all routers flood the whole AS with their LSPs to create the global LSDB.
- Although this may not create a problem in a small AS, it may have created a huge volume of traffic in a large AS
- OSPF uses another level of hierarchy in routing: The first level is the autonomous system, the second is the area
  - The AS needs to be divided into small sections called *areas*
  - *Each area acts as a small* independent domain for flooding LSPs.

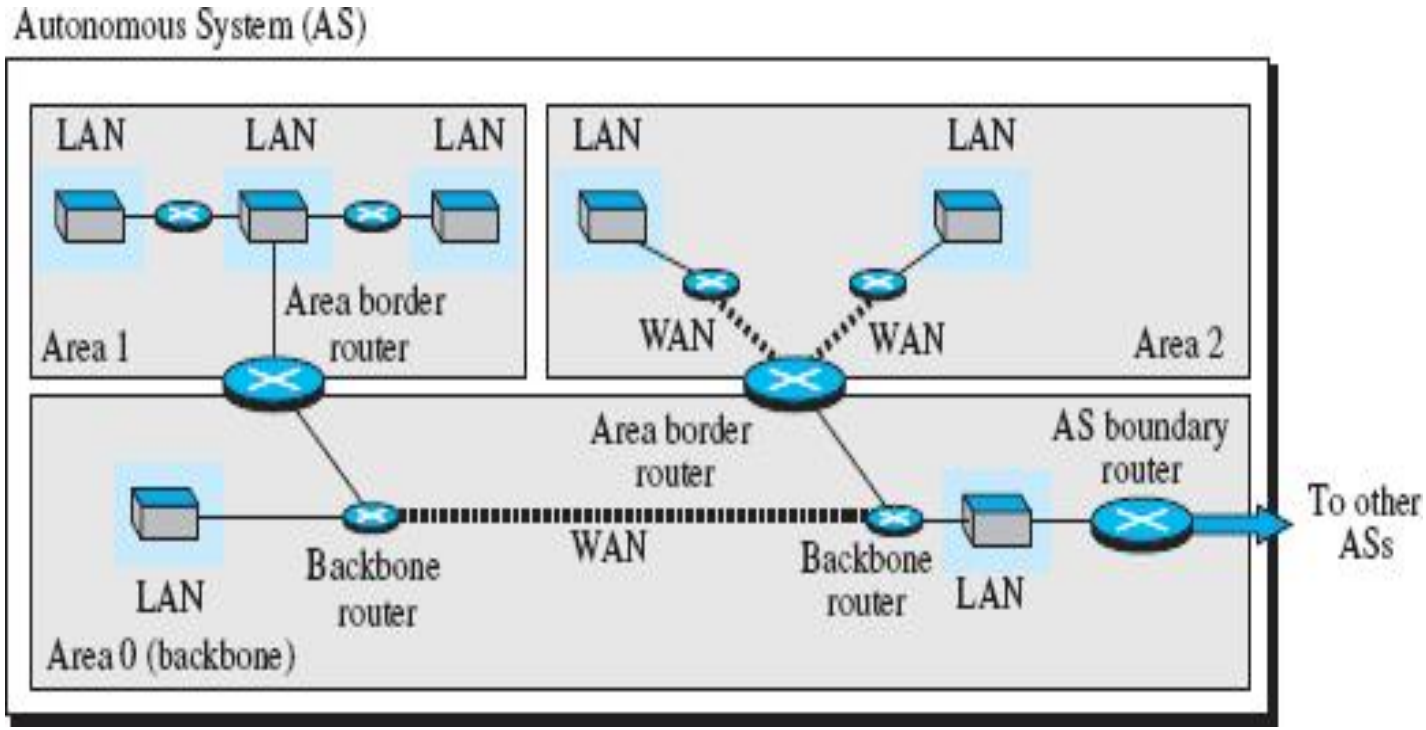
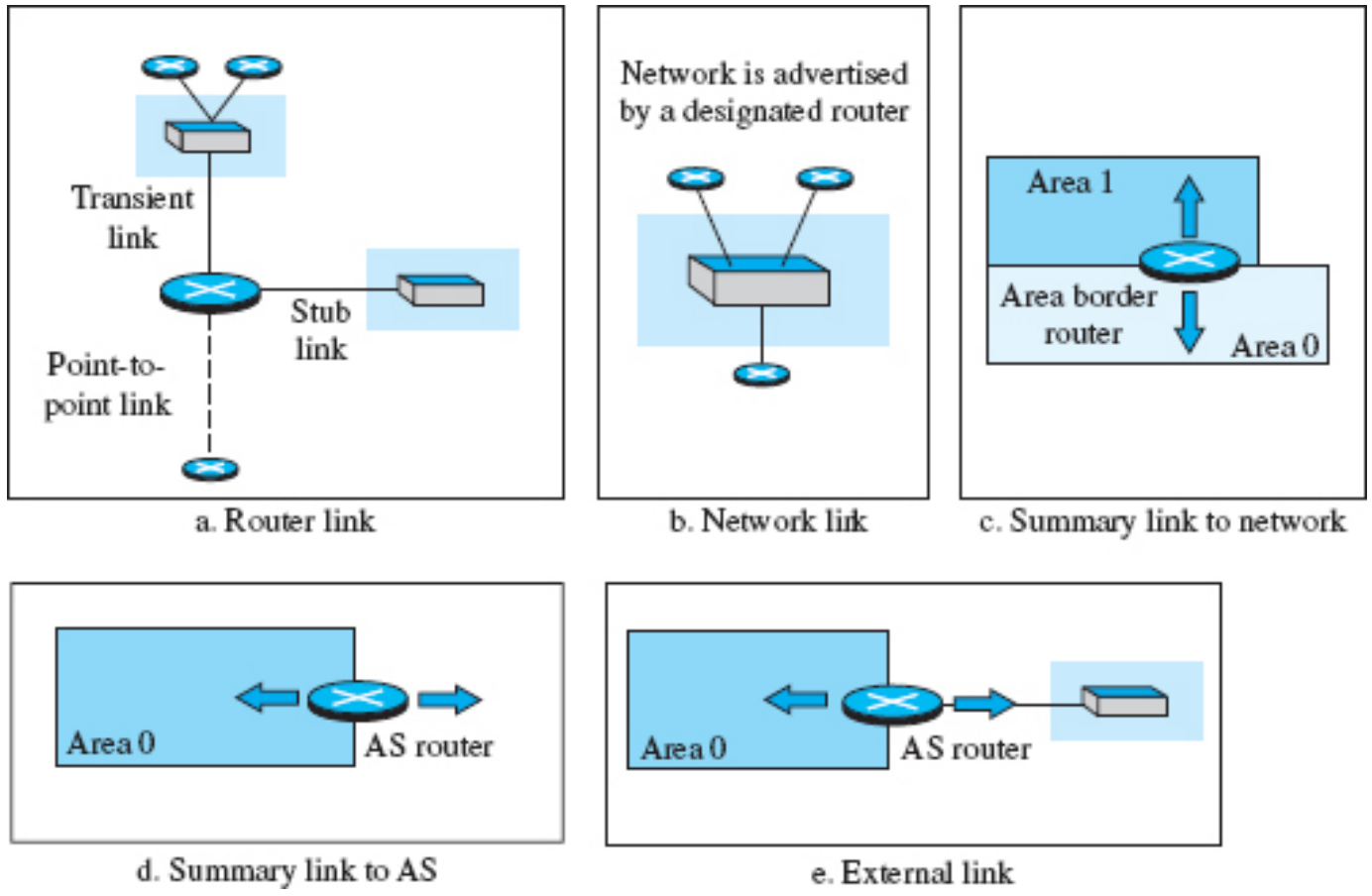


Figure : Areas in an autonomous system

- However, each router in an area needs to know the information about the link states not only in its area but also in other areas.
- For this reason, one of the areas in the AS is designated as the *backbone area, responsible for* gluing the areas together.
- The routers in the backbone area are responsible for passing the information collected by each area to all other areas.
- A router in an area can receive all LSPs generated in other areas
- For the purpose of communication, each area has an area identification.
- The area identification of the backbone is zero

Link-State Advertisement

- OSPF is based on the link-state routing algorithm, which requires that a router advertise the state of each link to all neighbors for the formation of the LSDB.
- We need to advertise the existence of different entities as nodes, the different types of links that connect each node to its neighbors, and the different types of cost associated with each link
  - This means we need different types of advertisements, each capable of advertising different situations
- We can have five types of link-state advertisements:
  - *router link*
  - *network link*
  - *summary link to network*
  - *summary link to AS border router*
  - *external link*



a. Router link

b. Network link

c. Summary link to network

d. Summary link to AS

e. External link

Link-State Advertisement

1. Router link - advertises the existence of a router as a node

In addition to giving the address of the announcing router, this type of advertisement can define one or more types of links that connect the advertising router to other entities

- A *transient link announces a link to a* transient network, a network that is connected to the rest of the networks by one or more routers
  - This type of advertisement should define the address of the transient network and the cost of the link
- A *stub link* advertises a link to a stub network, a network that is not a through network
  - Again, the advertisement should define the address of the network and the cost
- A *point-to-point link should define the address of* the router at the end of the point-to-point line and the cost to get there.

Link-State Advertisement

2. Network link - advertises the network as a node.

However, because a network cannot do announcements itself (it is a passive entity), one of the routers is assigned as the designated router and does the advertising

In addition to the address of the designated router, this type of LSP announces the IP address of all routers (including the designated router as a router and not as speaker of the network), but no cost is advertised because each router announces the cost to the network when it sends a router link advertisement.

3. Summary link to network -This is done by an area border router; it advertises the summary of links collected by the backbone to an area or the summary of links collected by the area to the backbone

- This type of information exchange is needed to glue the areas together.

Link-State Advertisement

4. Summary link to AS - This is done by an AS router that advertises the summary links from other ASs to the backbone area of the current AS, information which later can be disseminated to the areas so that they will know about the networks in other Ass

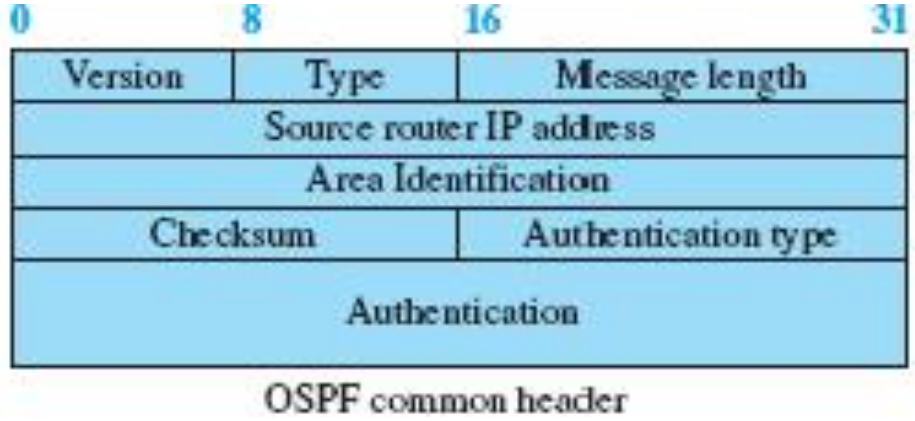
5. External link -This is also done by an AS router to announce the existence of a single network outside the AS to the backbone area to be disseminated into the areas.

OSPF Implementation

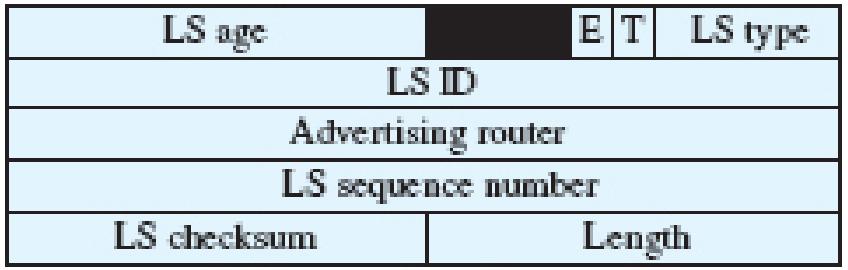
- OSPF is implemented as a program in the network layer that uses the service of the IP for propagation.
- An IP datagram that carries a message from OSPF sets the value of the protocol field to 89.
- This means that, although OSPF is a routing protocol to help IP to route its datagrams inside an AS, the OSPF messages are encapsulated inside datagrams.
- OSPF has gone through two versions: version 1 and version 2.
  - Most implementations use version 2.

OSPF Messages

- OSPF is a very complex protocol; it uses five different types of messages
- OSPF common header format (which is used in all messages) and the link-state general header (which is used in some messages)



OSPF common header



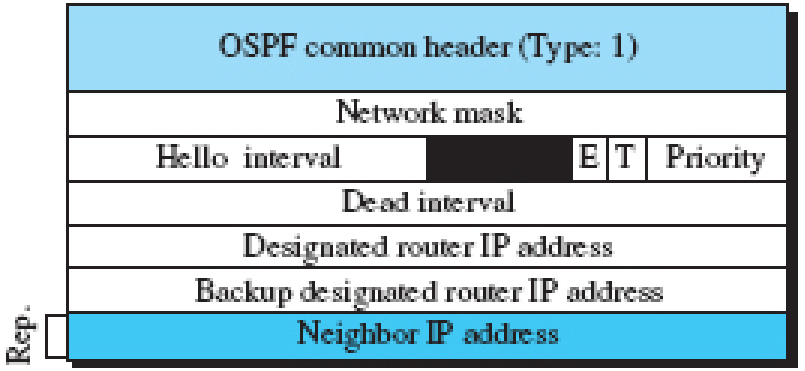
Link-state general header

Legend

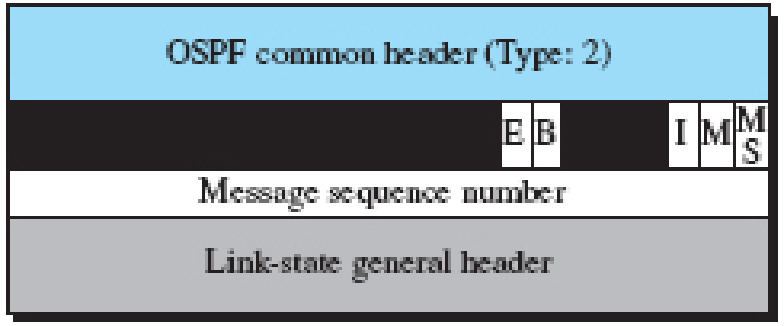
E, T, B, I, M, MS: flags used by OSPF
Priority: used to define the designated router
Rep.: Repeated as required

OSPF Messages

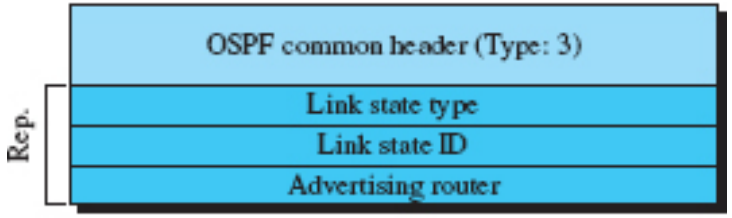
- Five message types used in OSPF
- The *hello message (type 1) is used by a router to introduce itself to* the neighbors and announces all neighbors that it already knows.
- The *database description message (type 2) is normally sent in response to the* hello message to allow a newly joined router to acquire the full LSDB
- The *link-state request message (type 3) is sent by a router that needs information* about a specific LS



Hello message



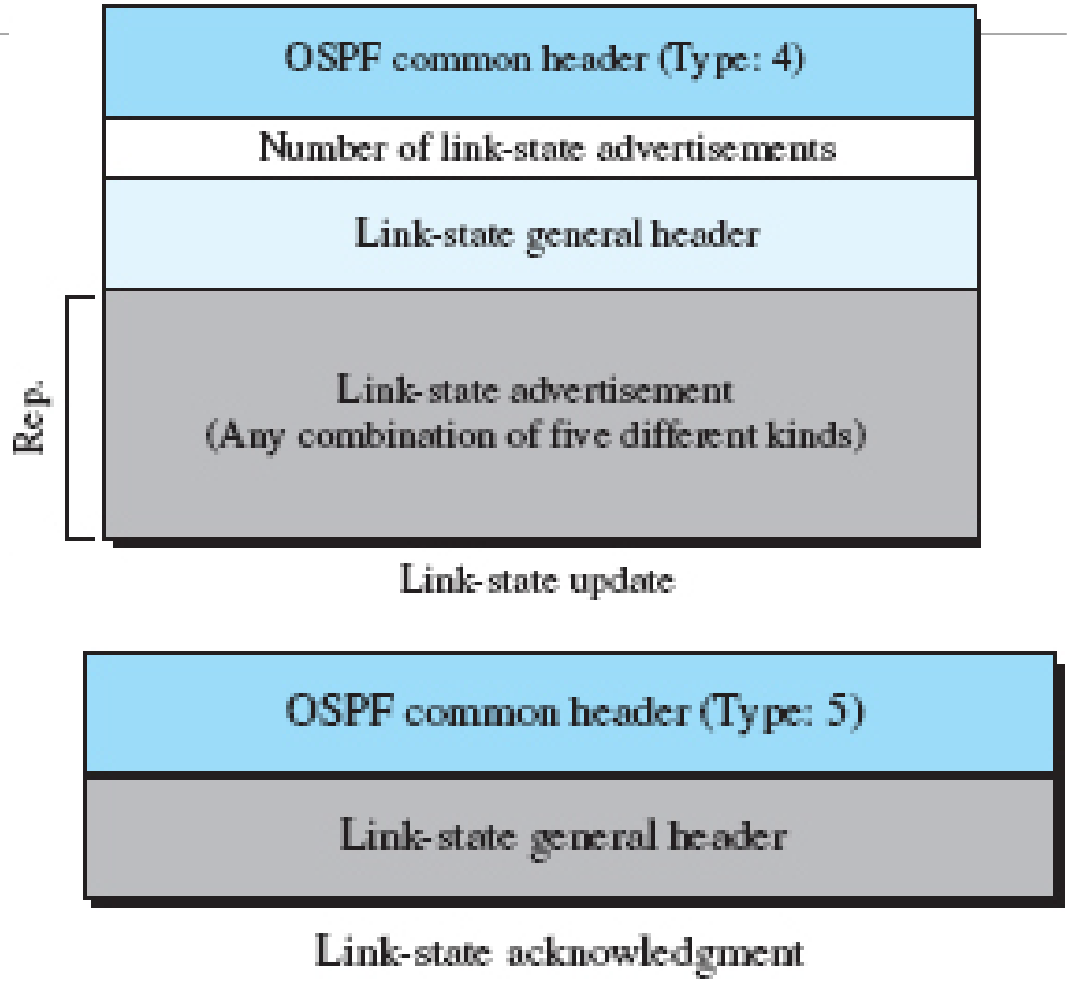
Database description



Link-state request

OSPF Messages

- The *link-state update message (type 4) is the main* OSPF message used for building the LSDB
  - This message, in fact, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link),
- The *link-state acknowledgment message (type 5) is used to* create reliability in OSPF
  - each router that receives a link-state update message needs to acknowledge it.



Authentication

- The OSPF common header has the provision for authentication of the message sender to prevents a malicious entity from sending OSPF messages to a router and causing the router to become part of the routing system to which it actually does not belong

OSPF Algorithm

- OSPF implements the link-state routing algorithm
- Some changes and augmentations need to be added to the algorithm:
  - After each router has created the shortest-path tree, the algorithm needs to use it to create the corresponding routing algorithm.
  - The algorithm needs to be augmented to handle sending and receiving all five types of messages.

Performance

- Update messages - The link-state messages in OSPF have a somewhat complex format
  - They also are flooded to the whole area
  - If the area is large, these messages may create heavy traffic and use a lot of bandwidth.
- Convergence of forwarding tables. - When the flooding of LSPs is completed, each router can create its own shortest-path tree and forwarding table; convergence is fairly quick. However, each router needs to run the Dijkstra's algorithm, which may take some time.
- Robustness - The OSPF protocol is more robust than RIP because, after receiving the completed LSDB, each router is independent and does not depend on other routers in the area
  - Corruption or failure in one router does not affect other routers as seriously as in RIP.

END