Unit-3

CS44 Data Communications and Networking

Outline of 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

Internet Protocol Version

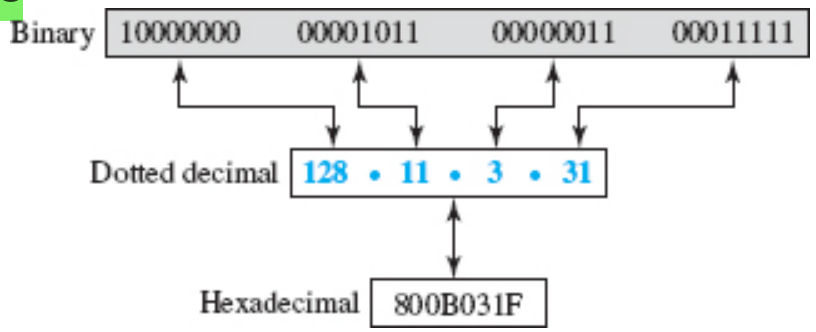
- two versions
  - IP Version 4 (IPv4) - almost depleted
  - IP Version 6 (IPv6)
- Internet address or IP address - identifier used in IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet
- IPv4 address is a 32-bit address
  - Unique - If a device has two connections to the Internet, via two networks, it has two IPv4 addresses
  - universal - the addressing system must be accepted by any host that wants to be connected to the Internet
  - the address of the connection, not the host or the router
    - because if the device is moved to another network, the IP address may be changed

Address space

- total number of addresses used by the protocol
- If a protocol uses *b bits* *to define an address*, *the address space is 2^b* *because each bit can* have two different values (0 or 1).
- 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 restrictions, more than 4 billion devices could be connected to the Internet.

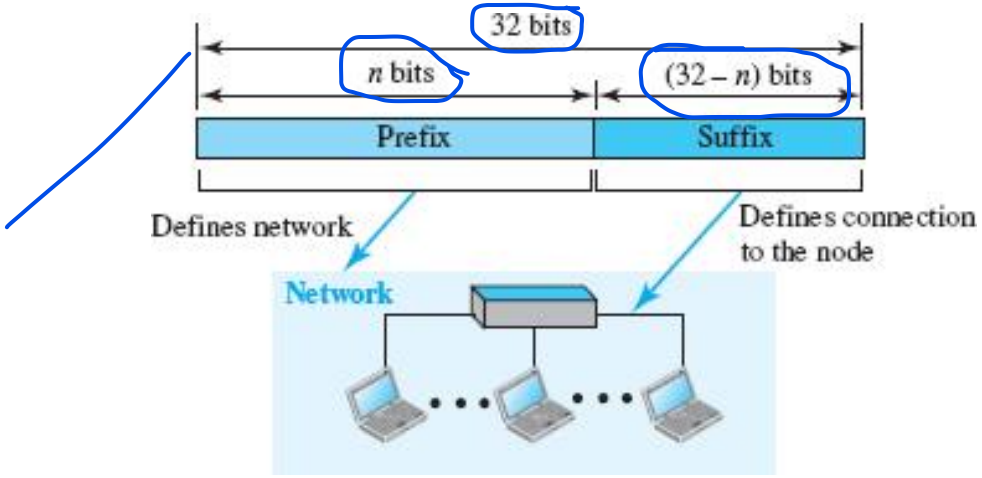
Notation

- binary notation (base 2) - *displayed as 32 bit.*
  - *To make the* address more readable, one or more spaces are usually inserted between each octet (8 bits).
  - Each octet is often referred to as a byte
- dotted-decimal notation (base 256) - usually written in decimal form with a decimal point (dot) separating the bytes
  - *each byte (octet) is only 8* bits, each number in the dotted-decimal notation is between 0 and 255
- hexadecimal notation (base 16)
- Each hexadecimal digit is equivalent to 4 bits. This means that a 32-bit address has eight hexadecimal digits



Hierarchy in Addressing

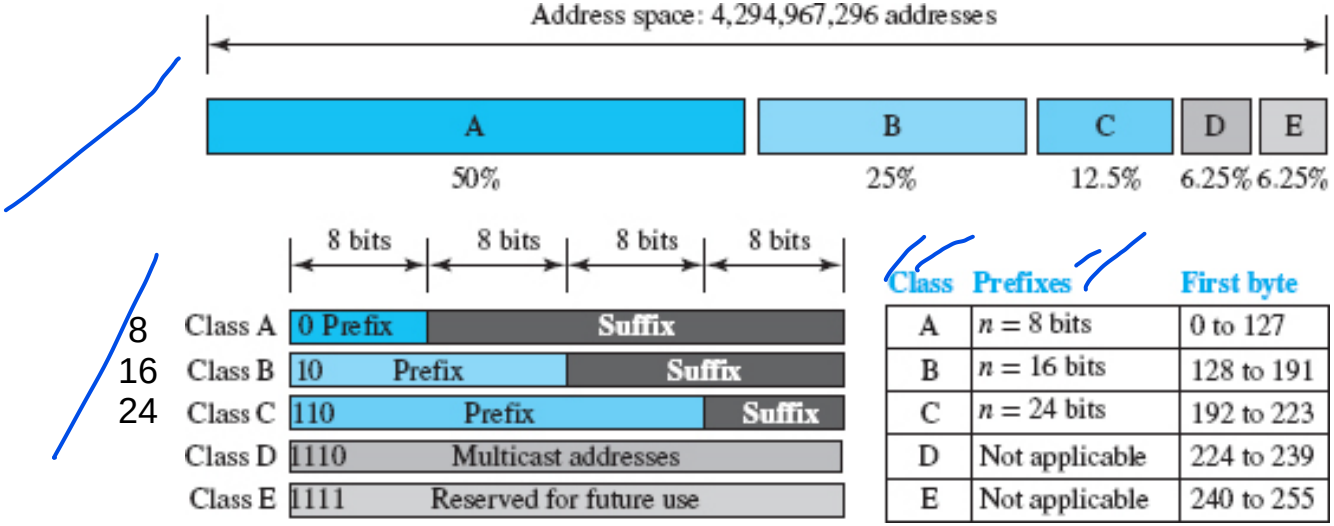
- any communication network that involves delivery - the addressing system is hierarchical
- 32-bit IPv4 address - hierarchical but is divided only into two parts
- *Prefix-defines the network (n-bits)*
  - fixed length -classful addressing - obsolete
  - variable length-classless addressing-
- Address (*suffix*) - *the node* - *connection of* a device to the Internet (32-n bits)



Classful Addressing

Fixed Length

- 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)
- helps us to understand classless 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

Classful Addressing cont…

- class A - network length is 8 bits but because the first bit is 0, we can have only 7 bits as the network identifier.
  - 2^7 = 128 networks in the world
- class B - the network length is 16 bits, but because the first 2 bits are $(10)_2$, we can have only 14 bits as the network identifier
- 2^{14} = 16,384 networks in the world
- class C - network length is 24 bits but because the first 3 bits are $(110)_2$ we can have only 21 bits as the network identifier.
  - 2^{21} = 2,097,152networks in the world
- Class D is not divided into prefix and suffix. It is used for multicast addresses
- class E -All addresses that start with 1111 in binary belong to.
- As in class D, class E is not divided into prefix and suffix and is used as reserve.

Address Depletion

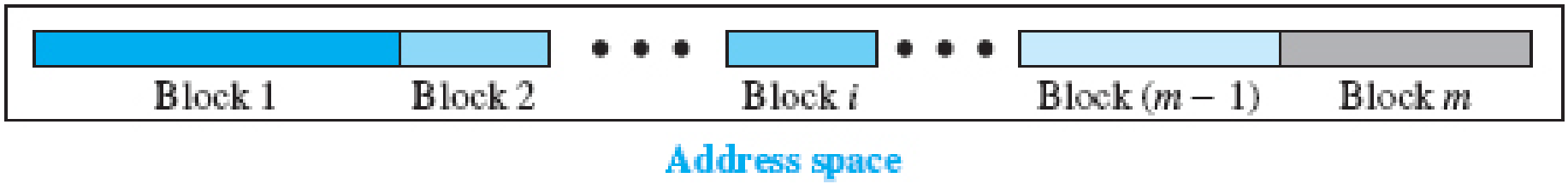
- the addresses were not distributed properly so the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses being available for organizations and individuals that needed to have an Internet connection.
- Eg. class A -assigned to only 128 organizations 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.
- 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

- new architecture announced in 1996 by the Internet authorities
- use the same address space but change the distribution of addresses to provide a fair share to each organization
- Motivation
  - Solution to address depletion
  - Manage Internet Service Providers (ISPs)
    - During the 1990s, Internet Service Providers (ISPs) came into prominence
    - An ISP is an organization that provides Internet access and services for individuals, small businesses, and midsize organizations that do not want to create an Internet site and become involved in providing Internet services (such as electronic mail) for their employees.
    - An ISP is granted a large range of addresses and then subdivides the addresses (in groups of 1, 2, 4, 8, 16, and so on), giving a range of addresses to a household or a small business
      - connected via a dial-up modem, DSL, or cable modem to the ISP

Classless Addressing

- whole address space is divided into non overlapping variable-length blocks - block of 2^0, 2^1, 2^2, …, 2^{32} addresses
  - restrictions - the number of addresses in a block needs to be a power of 2
- An organization can be granted one block of addresses.
- prefix in an address defines the block (network)
- suffix defines the node (device)



Address space

CIDR

Prefix Length: Slash Notation

- how to find the prefix length if an address is given?
- Because the prefix length is not inherent in the address, we need to separately give the length of the prefix.
  - 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

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:
  - the number of addresses
  - the first address in the block -**keep the *n leftmost bits and set the (32 – n)***
  - the last address- **keep the *n leftmost bits and set the (32 – n)*** rightmost bits all to 0s rightmost bits all to 1s
- Because the value of prefix length, *n, is given, we can easily find these three pieces of* information

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)].

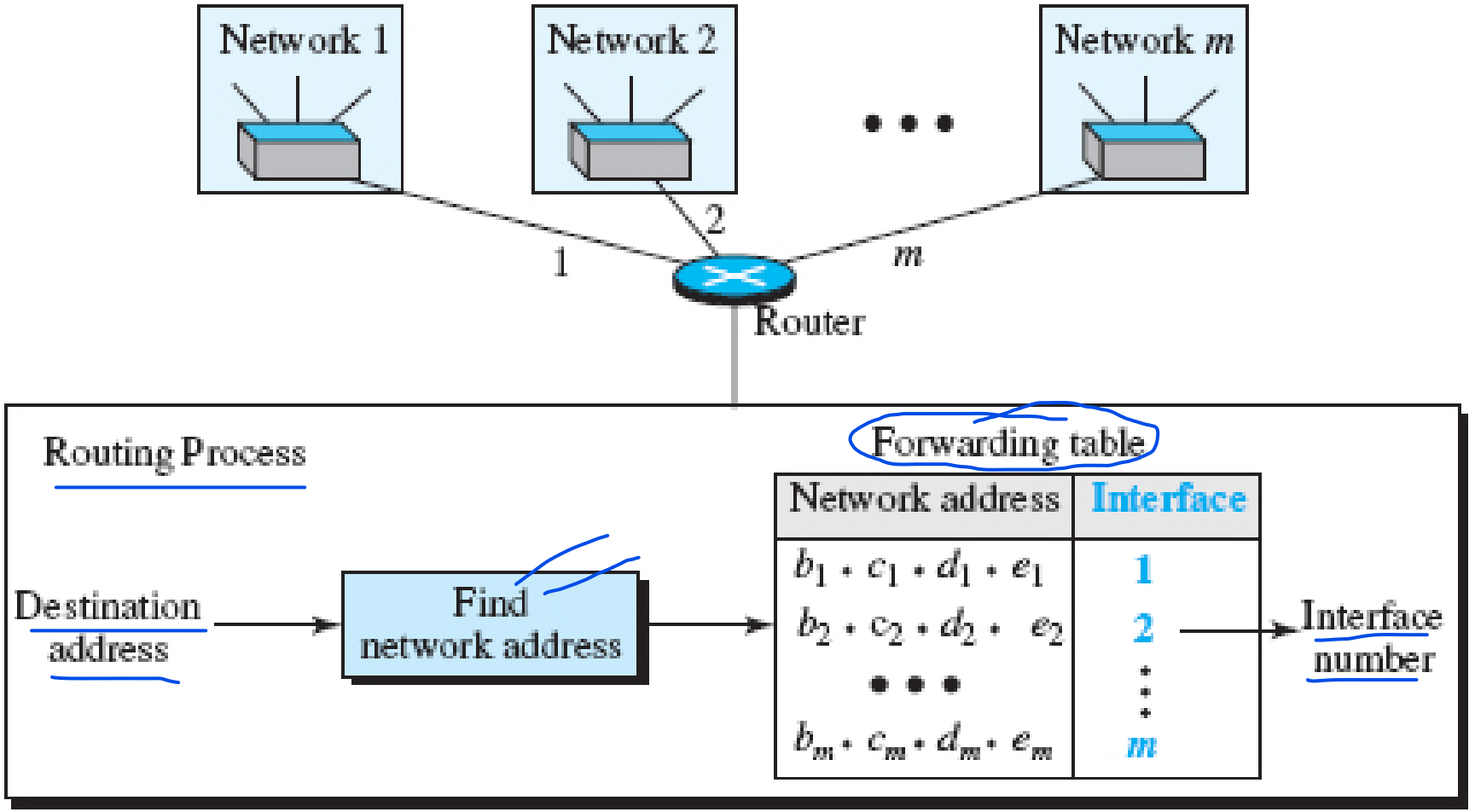
Address Mask

- 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

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

Network Address



Block Allocation

- How are the blocks allocated?
- 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 (or a larger organization that is considered an ISP in this case).
- For the proper operation of the CIDR, two restrictions need to be applied to the allocated block
  - The number of requested addresses, *N, needs to be a power of 2*
  - The requested block needs to be allocated where there are a contiguous number of available addresses in the address space
    - first address needs to be divisible by the number of addresses in the block
    - First address = (prefix in decimal) × 2^{32-n} = *(prefix in decimal)* × *N*

$$n = 32 - \log_2 N$$

n = prefix
N = no of blocks

Example

- 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

- Created more levels of hierarchy
- 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).
- *N* - total number of addresses granted to the organization
- *n* - *prefix length*
- *Nsub*- *the assigned* number of addresses to each subnetwork
- *nsub* - *the prefix length for* each subnetwork
- *steps* 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 : $nsub = 32 - log_2 Nsub$
  - 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.
- each subnetwork information (first and last address) – same process seen before

Example

- An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have three subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks.
  - There are 2^{32-24} = 256 addresses in this block. out of 32 bits, 24 are fixed
  - 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.

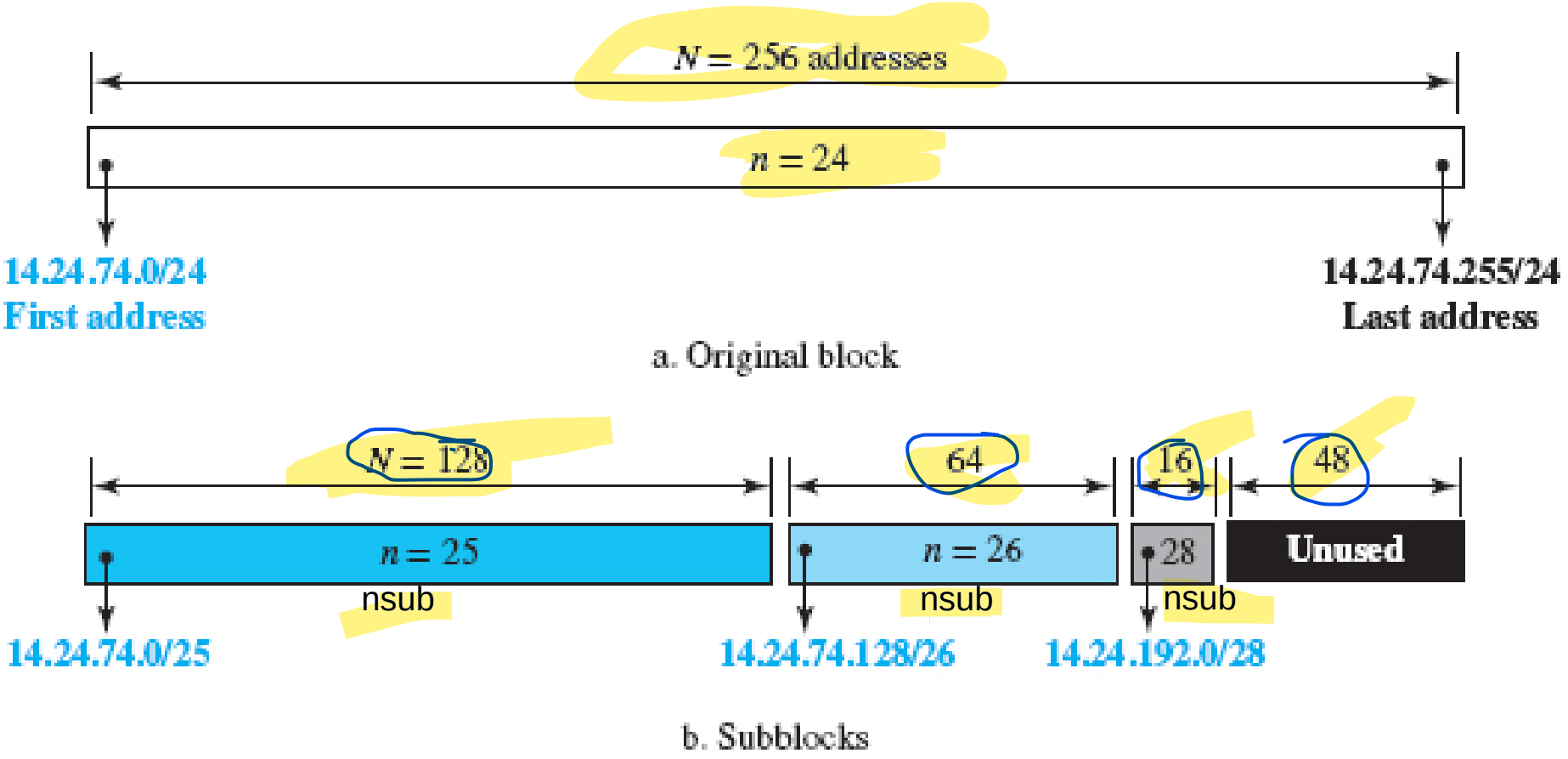
Example

- **a. 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.
- **b. 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 $n2 = 32 - \log2\ 64 = 26$.
  - *The* first address in this block is 14.24.74.128/26;
  - the last address is 14.24.74.191/26.

Example

- **c.** The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either.
  - 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 14.24.74.207/28
- If we add all addresses in the previous subblocks, the result is 208 addresses,
  - means 48 addresses are left in reserve.
- The first address in this range is 14.24.74.208.
- The last address is 14.24.74.255.

Example


N = 256 addresses
n = 24
14.24.74.0/24
First address
14.24.74.255/24
Last address
a. Original block
N = 128
64
16
48
n = 25
n = 26
28
Unused
nsub
nsub
nsub
14.24.74.0/25
14.24.74.128/26
14.24.192.0/28
b. Subblocks

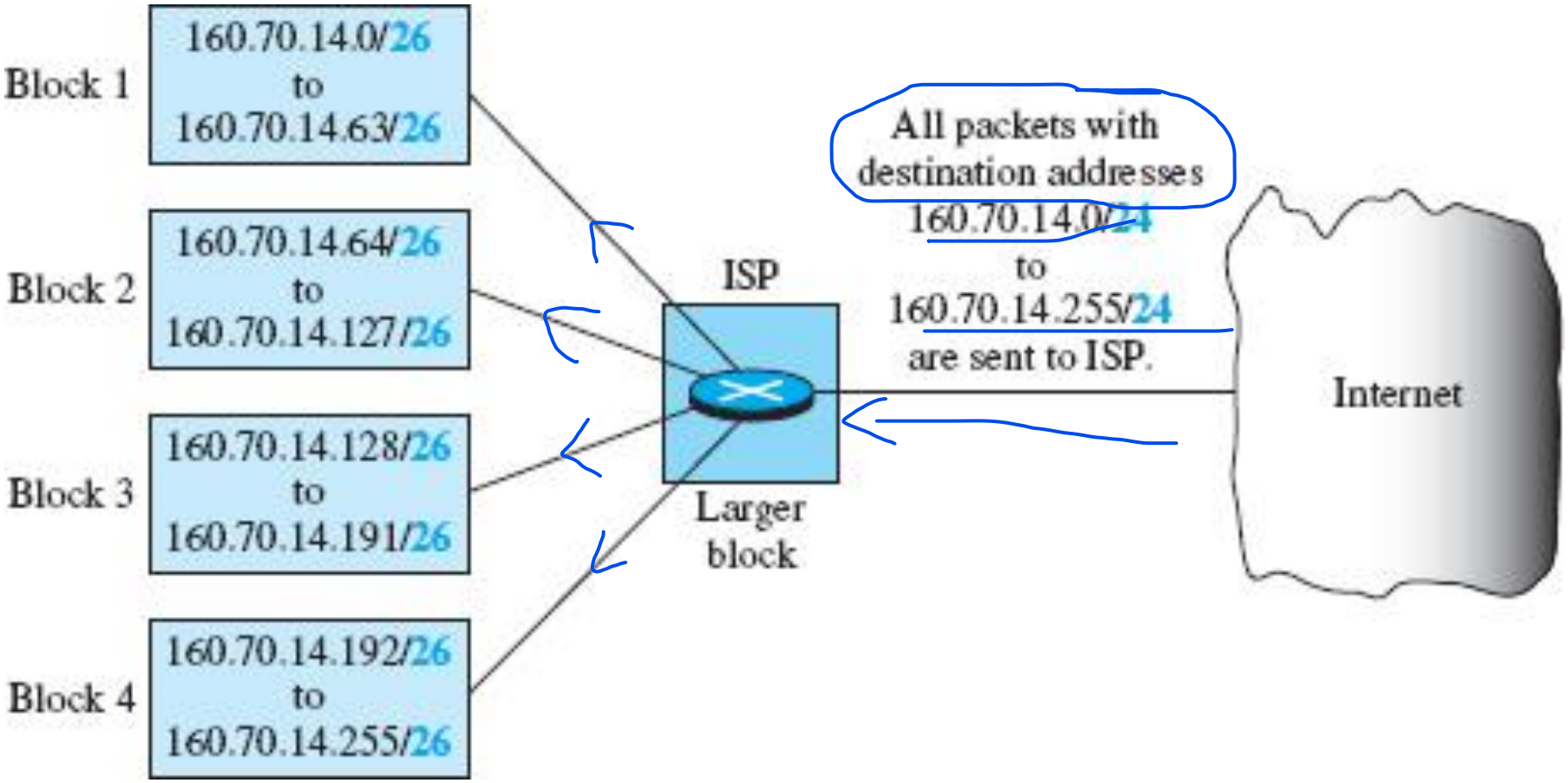
Address Aggregation

- 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.
- ICANN assigns a large block of addresses to an ISP. Each ISP in turn divides its assigned block into smaller subblocks and grants the subblocks to its customers.

Address Aggregation

- Four small blocks of addresses are assigned to four organizations by an ISP.
- The ISP combines these four blocks into one single block and advertises the larger block to the rest of the world.
- Any packet destined for this larger block should be sent to this ISP.
- It is the responsibility of the ISP to forward the packet to the appropriate organization.
- a postal network - All packages coming from outside a country are sent first to the capital and then distributed to the corresponding destination.

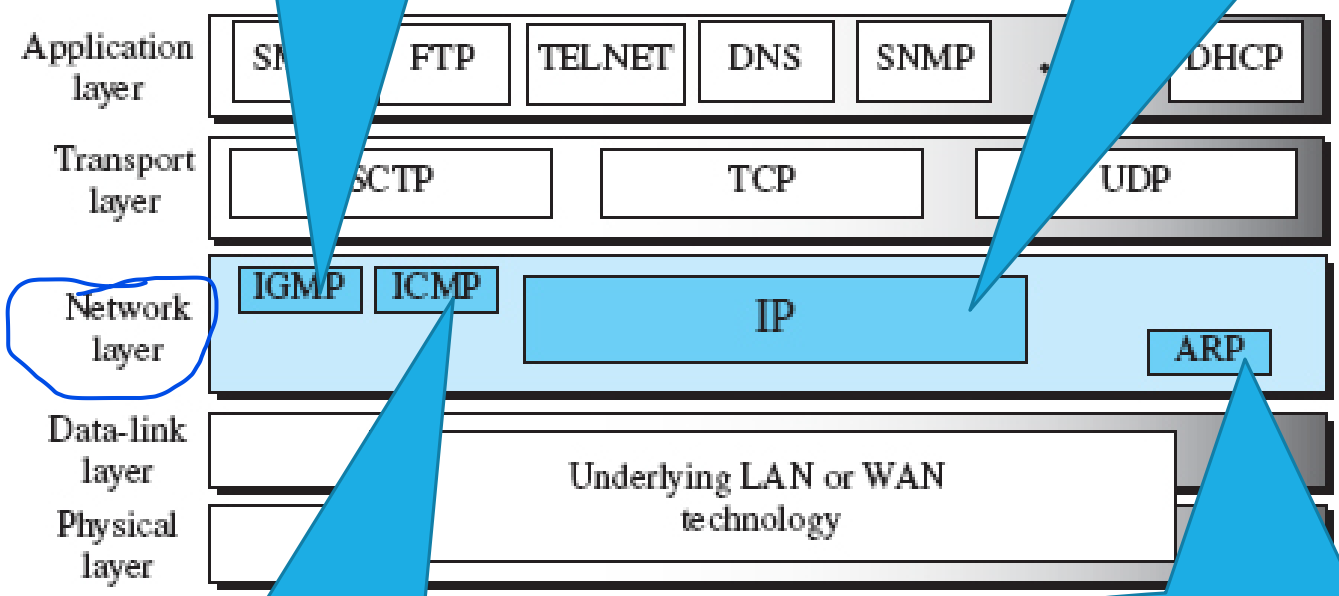
Address Aggregation



Main and Auxiliary Protocols

- The network layer in version 4 - one main protocol and three auxiliary protocols
  - The main protocol, Internet Protocol version 4 (IPv4)
  - The Internet Control Message Protocol version 4 (ICMPv4)
  - The Internet Group Management Protocol (IGMP)
  - The Address Resolution Protocol (ARP)

Main and Auxiliary Protocols



Auxiliary protocol: help IPv4 in multicasting

Main protocol: responsible for packetizing, forwarding, and delivery of a packet

Auxiliary protocol: helps IPv4 to handle some errors that may occur in the network layer delivery

Auxiliary protocol: glue the network and data-link layers in mapping network-layer addresses to link-layer addresses

IPv4

- Unreliable datagram protocol—a best-effort delivery service.
- *Best effort means that IPv4 packets can be corrupted, be lost, arrive* out of order, be delayed or create congestion for the network.
- If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP.
- Common best-effort delivery service is the post office.
  - The post office does its best to deliver the regular mail but does not always succeed.
  - The post office itself does not keep track of every letter and cannot notify a sender of loss or damage of one.
  - If an unregistered letter is lost or damaged in the mail, the would-be recipient will not receive the correspondence and the sender will need to re-create it.

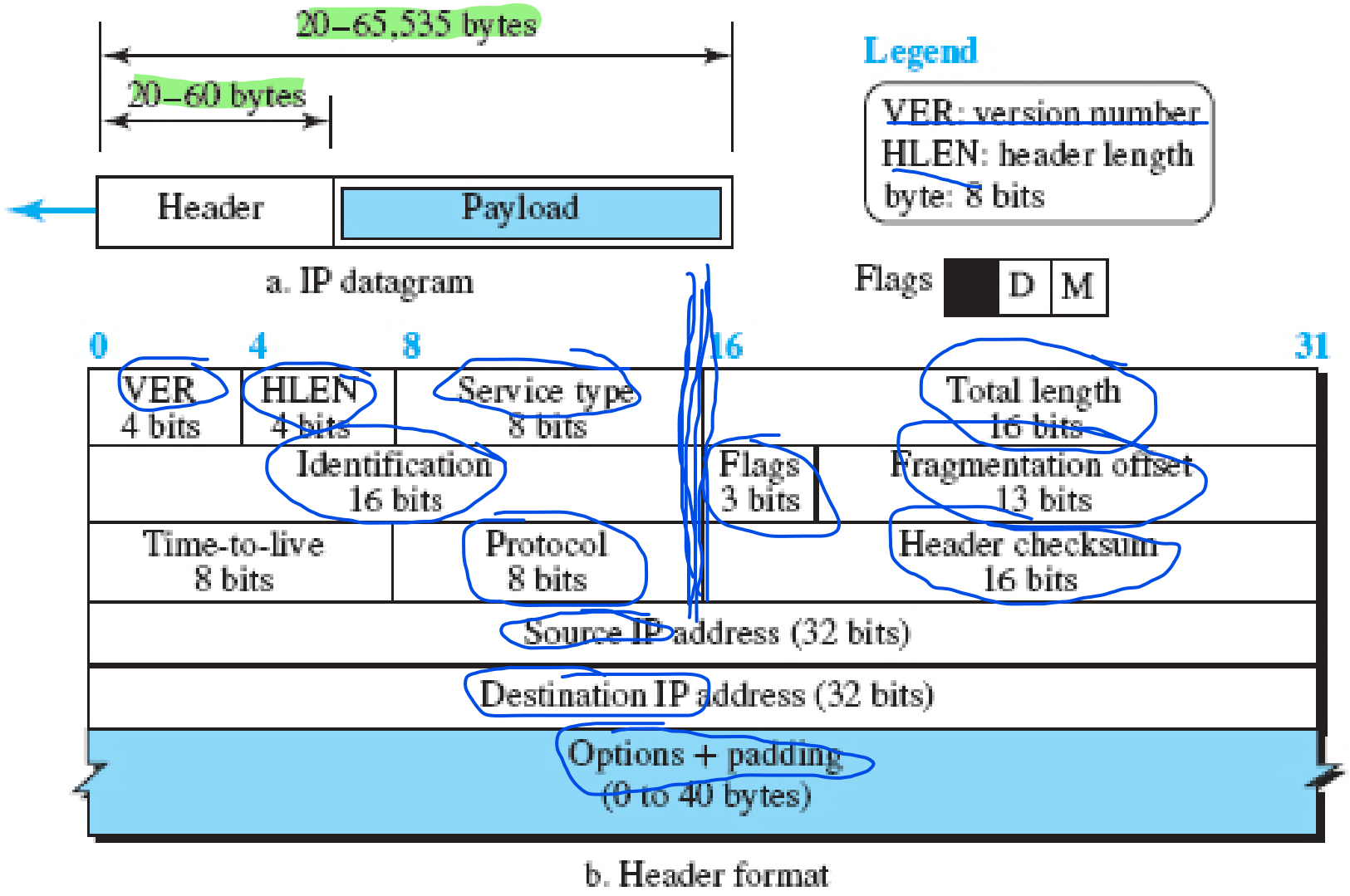
IPv4

- IPv4 is a connectionless protocol 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.
- IPv4 relies on a higher-level protocol to take care of all these problems.

IPv4 Datagram Format

- first service provided by IPv4 – packetizing
- IPv4 format of a packet in which the data coming from the upper layer or other protocols are encapsulated.
- Packets used by the IP are called *datagrams.*
- A datagram is a variable-length packet consisting of two parts:
  - the header is 20 to 60 bytes in length and contains information essential to routing and delivery
    - first 20 bytes are essential and together are called the main header
    - next 40 bytes include options and padding that may or may not be present
    - It is customary in TCP/IP to show the header in 4-byte sections.
  - payload (data).
  - The header

IPv4 Datagram Format



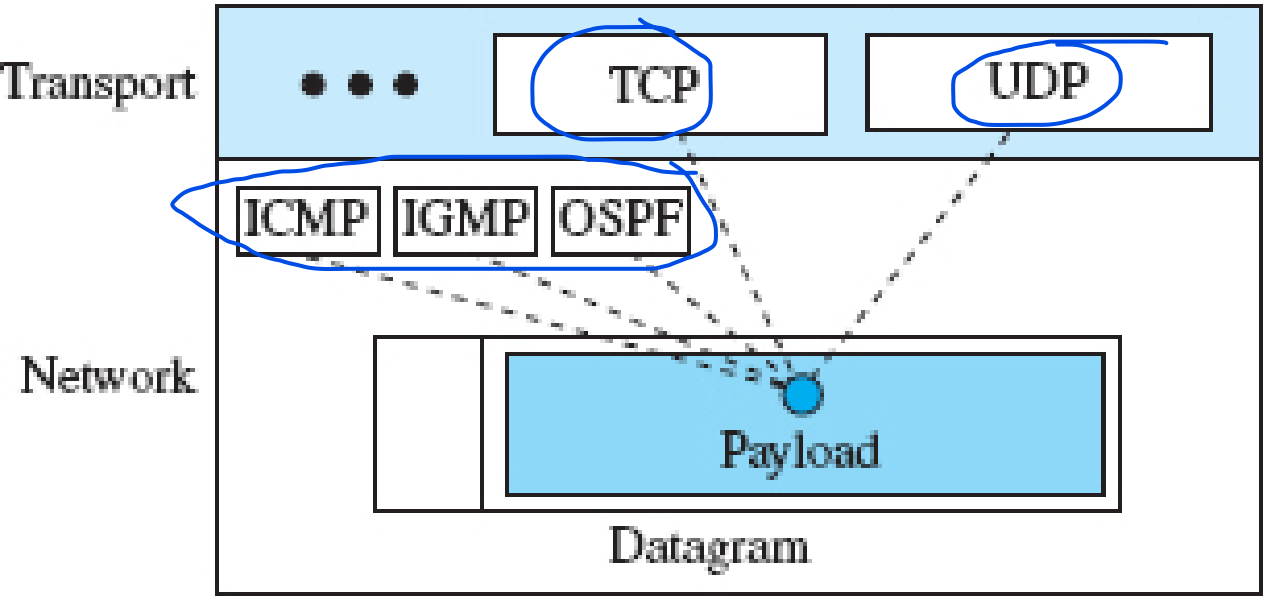
IPv4 Datagram Format

- ***Version*** specifies the IP version.
- ***Header length*** **(**HLEN**)** specifies the length of the header (including options and padding) in terms of 4-byte blocks. For example, if the total header of a packet (including options and padding) is 60B, HL=60B/4B=15 which is 1111 in binary.
- ***Type of service*** specifies the quality-of-service (QoS) requirements of the packet, such as priority level, delay, reliability, throughput, and cost.
- ***Total length*** specifies the total length of the packet in bytes, including the header and data. A total of 16 bits are assigned to this field. Length of data = total length − (HLEN) × 4
  - Helps to determine how much is really data and how much is padding
- ***Identification, flags*****, and *fragment offset*** are used for packet fragmentation and reassembly.

IPv4 Datagram Format

- ***Time to live*** specifies the maximum number of hops after which a packet must be discarded.
  - value is approximately 2 times the maximum number of routes between any two hosts.
  - Each router that processes the datagram decrements this number by one.
  - If this value, after being decremented, is zero, the router discards the datagram.
- ***Protocol*** specifies the protocol of packet in payload.
  - A datagram can carry a packet from other protocols that directly use the service of the IP, such as some routing protocols, any transport-layer protocol or some auxiliary protocols.
  - provides multiplexing at the source and demultiplexing at the destination,

IPv4 Datagram Format



Some protocol values

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

IPv4 Datagram Format

- ***Header checksum*** is a method of error detection
  - the value of some fields, such as TTL, which are related to fragmentation and options, may change from router to router, the checksum needs to be recalculated at each router
- ***Source address*** and ***destination*** *address* are 32-bit fields specifying the source address and the destination address, respectively.
- remain unchanged during the time the IP datagram travels from the source host to the destination host.
- ***Options*** is a rarely used variable-length field to specify security level, timestamp, and type of route.
- used for network testing and debugging
- ***Padding*** is used to ensure that the header is a multiple of 32 bits.

Example

- An IPv4 packet has arrived with the first 8 bits as $(01000010)_2$. The receiver discards the packet. Why?
  - 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?
  - 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?
  - 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. $(450000280001000001 02 \ldots)_{16}$
- How many hops can this packet travel before being dropped? To which upper-layer protocol do the data belong?
  - 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.

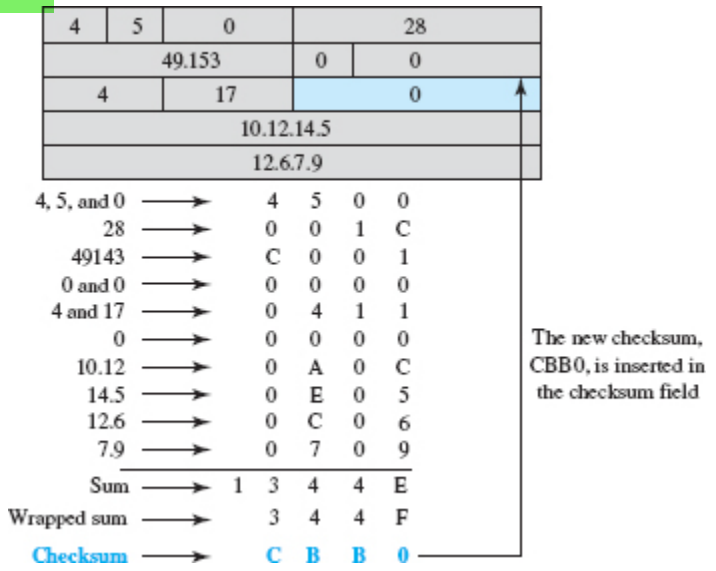
Example

- 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

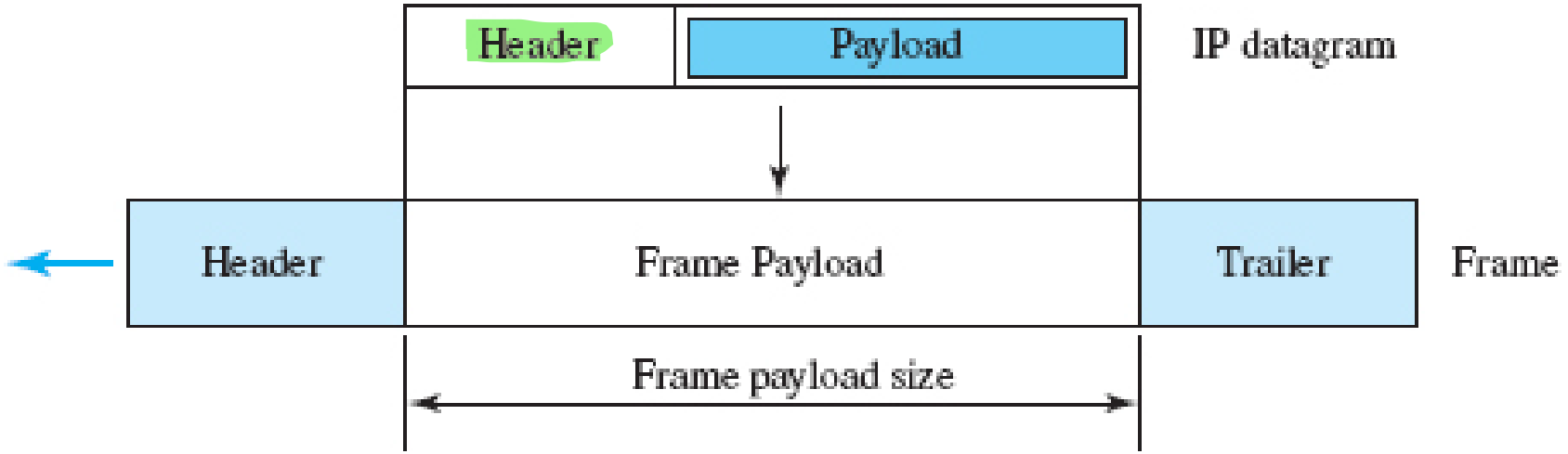


Fragmentation

- 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)

- Each link-layer protocol has its own frame format.
- 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



MTU: Maximum size of frame payload

Fragmentation

- The value of the maximum transfer unit (MTU) differs from one physical network protocol to another.
- For example, the value for a LAN is normally 1500 bytes, but for a WAN it can be larger or smaller.
- need a mechanism that avoids requiring large buffers at intermediate routers to store the fragments.
  - To make the IP protocol independent of the physical network, the designers decided to make the maximum length of the IP datagram equal to 65,535 bytes.
  - This makes transmission more efficient if one day we use a link-layer protocol with an MTU of this size.
  - However, for other physical networks, we must divide the datagram to make it possible for it to pass through these networks. This is called **fragmentation.**
    - payload of the IP datagram is fragmented
    - most parts of the header, with the exception of some options, must be copied by all fragments.

Fragmentation

- When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed.
- A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU.
- A datagram can be fragmented by the source host or any router in the path.
- The *reassembly of the datagram, however, is done only by the* destination host, because each fragment becomes an independent datagram.
- Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all the fragments belonging to the same datagram should finally arrive at the destination host.
- So it is logical to do the reassembly at the final destination.
- An even stronger objection for reassembling packets during the transmission is the loss of efficiency it incurs.

Fragmentation

- 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.
- The **identification field** identifies a datagram originating from the source host.
  - Counter is initialized to a positive number
  - When a datagram is fragmented, the value in the identification field is copied into all fragments
  - identification number helps the destination in reassembling the fragmented packets.
  - The **offset field** indicates the position of a fragment in the sequence of fragments making up the packet. The lengths of all the fragments, with the exception of the last one, must be divisible by 8.

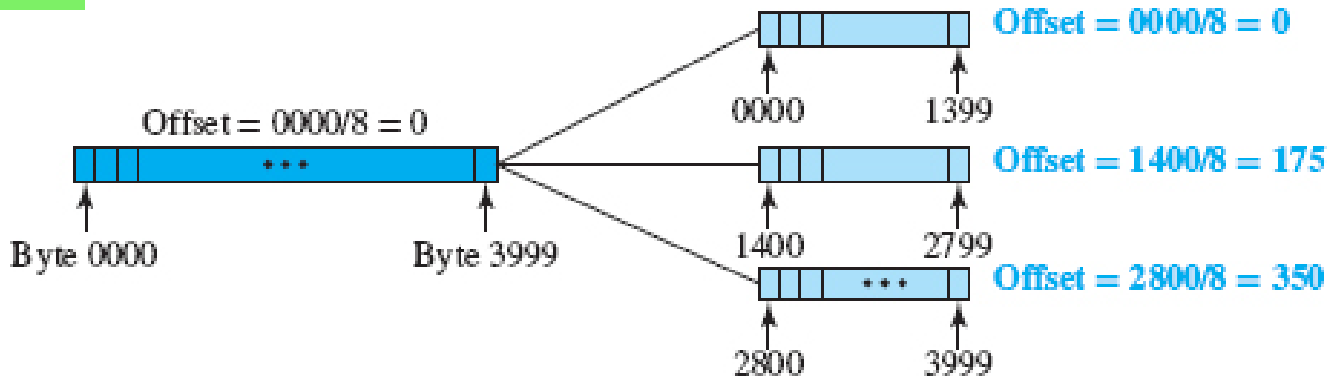
Fragmentation

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*.
  - *If its* value is 1, the machine must not fragment the datagram.
  - If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host
  - If its value is 0, the datagram can be fragmented if necessary.
- The third bit (M bit) is called the *more fragment bit.*
  - *If its value is 1, it means the* datagram is not the last fragment; there are more fragments after this one.
  - If its value is 0, it means this is the last or only fragment.

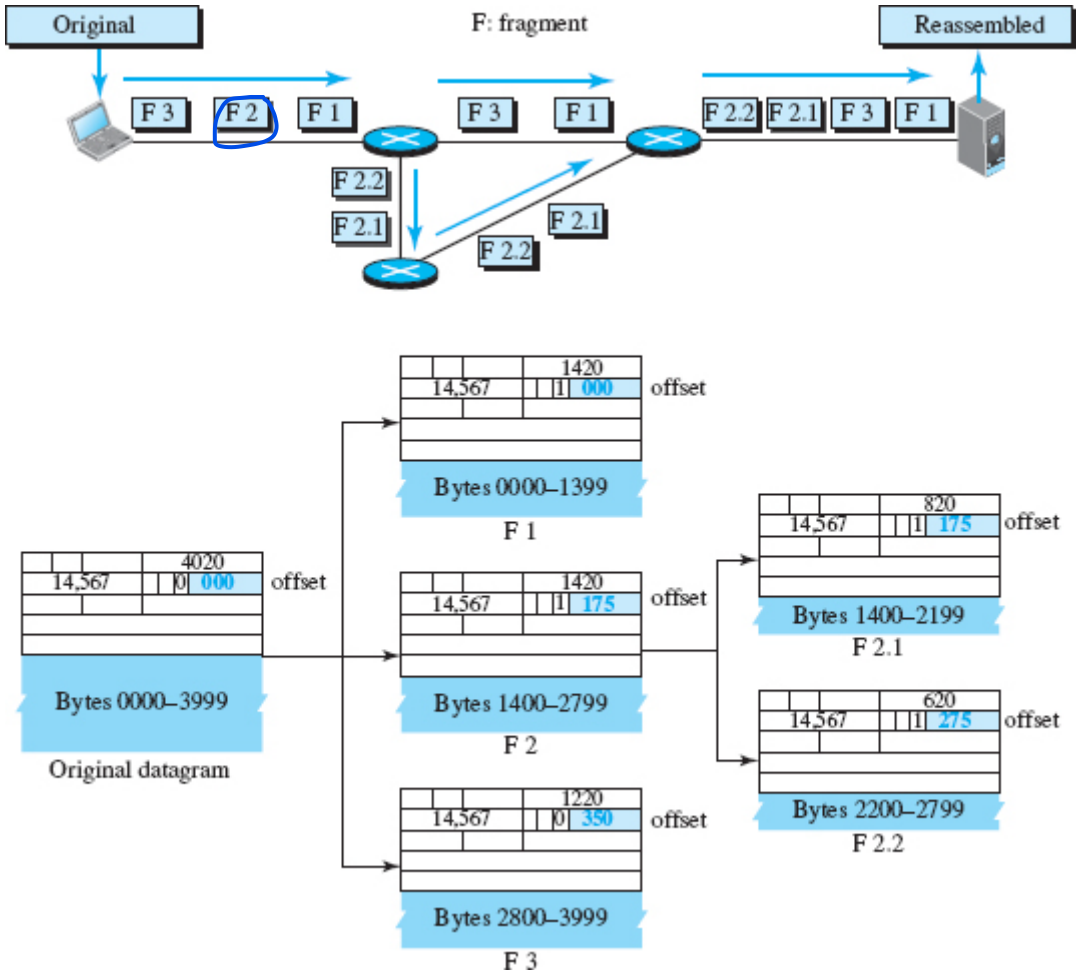
Fragmentation

- The 13-bit *fragmentation offset 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.
- Datagram with a data size of 4000 bytes (numbered 0 to 3999) fragmented into three fragments.
- The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 = 0.
- The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175.
- Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 = 350.



Fragmentation

- The original packet starts at the client; the fragments are reassembled at the Server
- M bit set for all fragments except the last
- the second fragment is itself fragmented later into two fragments of 800 bytes and 600 bytes, but the offset shows the relative position of the fragments to the original data.



Fragmentation

- Reassembling 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.

Fragmentation - example

- Why value of the offset is measured in units of 8 bytes?
- the length of the offset field is only 13 bits long and cannot represent a sequence of bytes greater than 8191.
- This forces hosts or routers that fragment datagrams to choose the size of each fragment so that the first byte number is divisible by 8.

Fragmentation - 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?
  - 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.

Fragmentation - example

- 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?
  - 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).

Fragmentation - 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?
  - 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.

Fragmentation - 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?
  - 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.

Fragmentation - 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?
  - 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

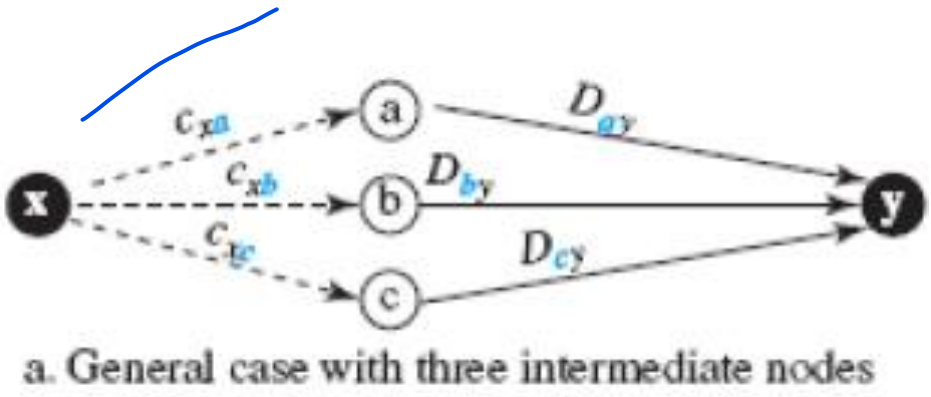
- Goal - to find the best route
- each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbors
- The incomplete trees are exchanged between immediate neighbors to make the trees more and more complete and to represent the whole internet
  - router continuously tells all its neighbors about what it knows about the whole internet (although the knowledge can be incomplete).
- Uses two important topics: the Bellman- Ford equation and the concept of distance vectors

Bellman-Ford Equation

- find the least cost (shortest distance) between a source node, *x, and a destination node, y, through some intermediary nodes* (a, b, c, …) , when following are given
  - the costs between the source and the intermediary nodes
  - the least costs between the intermediary nodes and the destination
- General case in which D_{ij} *is the shortest* distance and c_{ij} *is the cost between nodes i and j*
  - $D_{xy} = min \{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$
- Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths.

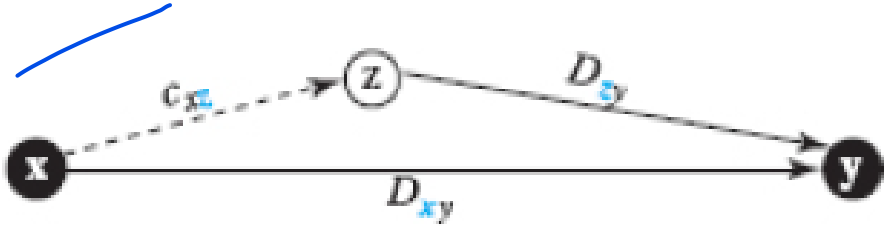
Bellman-Ford Equation

- 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 using
  - $D_{xy} = min\{D_{xy}, (c_{xz} + D_{zy})\}$
- *Graphical idea behind Bellman-Ford equation*



a. General case with three intermediate nodes

(*a → y), (b → y), and (c → y) as previously established* least-cost paths

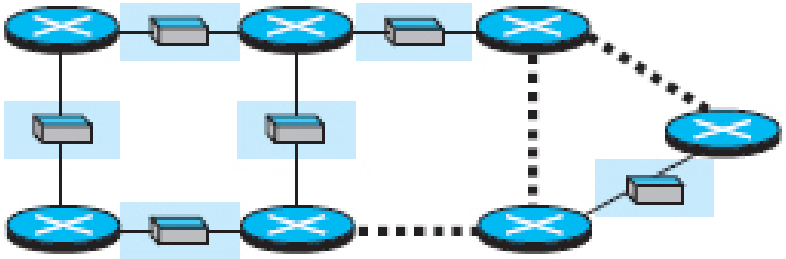


b. Updating a path with a new route

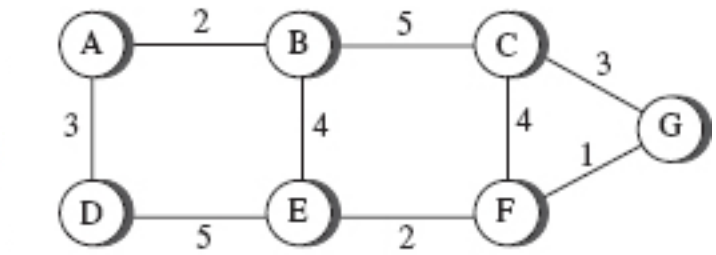
(*x → y) as the new least-cost path*

Distance vector

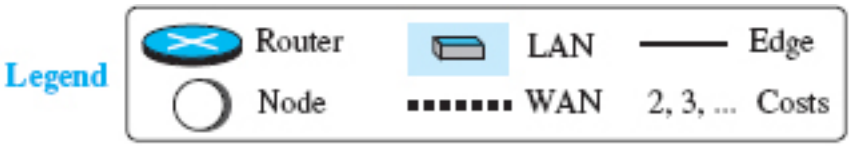
- *A least-cost tree is a combination of least-cost paths from the root of the tree to all destinations*
  - These paths are graphically glued together to form the tree.
- Distance-vector routing unglues these paths and creates a *distance vector, a one-dimensional array to represent the tree*
- *Consider an internet and its graphical representation*



a. An internet

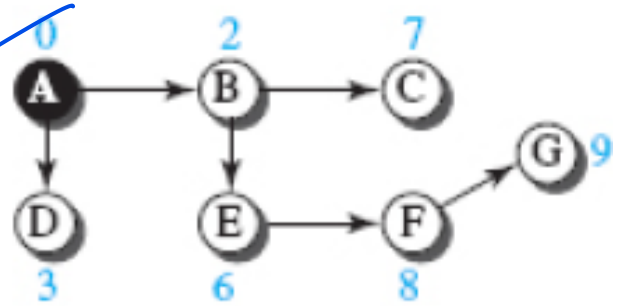


b. The weighted graph



Distance vector

- Tree for Node A and corresponding distance vector



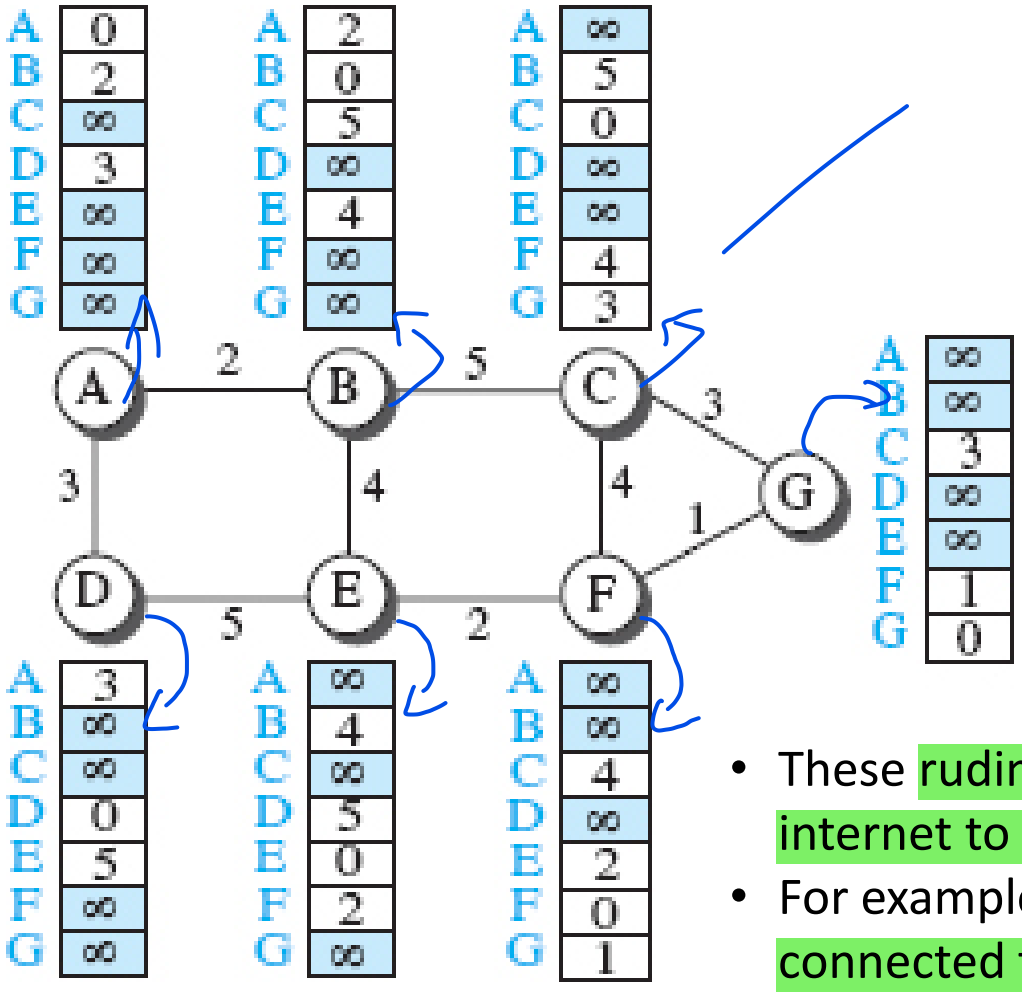
	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

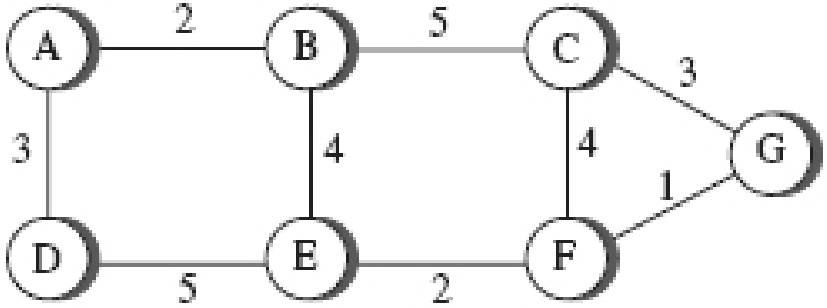
- *Name of the distance vector defines the root, the indexes* define the destinations, and the *value of each cell defines the least cost from* the root to the destination.
- A distance vector gives only the least costs to the destinations not the path to the destinations as the least-cost tree does
- how each node in an internet originally creates the corresponding vector?

Distance vector

- Each node in an internet, when it is booted, creates a very rudimentary distance vector with the minimum information the node can obtain from its neighborhood.
- The node sends some greeting messages out of its interfaces and discovers the identity of the immediate neighbors and the distance between itself and each neighbor
- It then makes a simple distance vector by inserting the discovered distances in the corresponding cells and leaves the value of other cells as infinity
- When we know only one distance between two nodes, it is the least cost.
- These vectors are made asynchronously, when the corresponding node has been booted
  - The existence of all of them in a figure does not mean synchronous creation of them

First distance vector for an internet



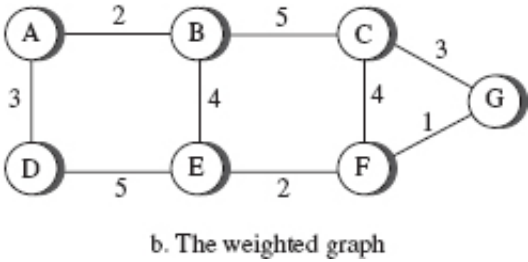


b. The weighted graph

- These rudimentary vectors cannot help the internet to effectively forward a packet.
- For example, node A thinks that it is not connected to node G because the corresponding cell shows the least cost of infinity.

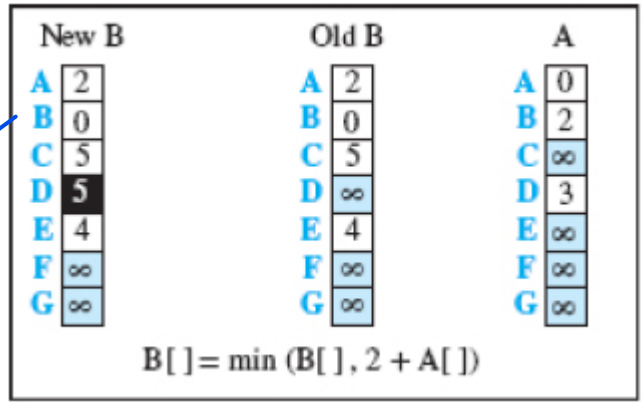
Updating distance vectors

- After each node has created its vector, it sends a copy of the vector to all its immediate neighbors
- After a node receives a distance vector from a neighbor, it updates its distance vector using the Bellman-Ford equation: $D_{xy} = min\{D_{xy}, (c_{xz} + D_{zy})\}$
- Two asynchronous events, happening one after another with some time in between.

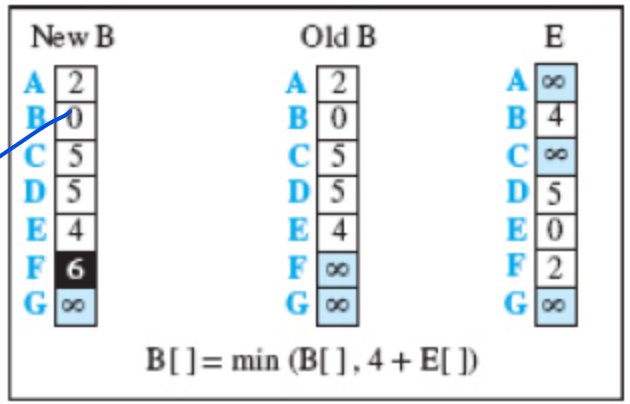


b. The weighted graph

X[]: the whole vector



Node A has sent its vector to node B. Node B updates its vector using the cost $c_{BA} = 2$



Node E has sent its vector to node B. Node B updates its vector using the cost $c_{EA} = 4$

Updating distance vectors

- After the first event, node B has one improvement in its vector; its least cost to node D has changed from infinity to 5 (via node A)
- After the second event, node B has one more improvement in its vector; its least cost to node F has changed from infinity to 6 (via node E)
- Exchanging vectors eventually stabilizes the system and allows all nodes to find the ultimate least cost between themselves and any other node
- After updating a node, it immediately sends its updated vector to all neighbors.
- Even if its neighbors have received the previous vector, the updated one may help more.

Distance-Vector Routing Algorithm

```
Distance_Vector_Routing ( )
{
  // Initialize (create initial vectors for the node)
  D[myself] = 0
  for (y = 1 to N)
  {
    if (y is a neighbor)
      D[y] = c[myself][y]
    else
      D[y] = ∞
  }
  send vector {D[1], D[2], ..., D[N]} to all neighbors
  // Update (improve the vector with the vector received from a neighbor)
  repeat (forever)
  {
    wait (for a vector D_w from a neighbor w or any change in the link)
    for (y = 1 to N)
    {
      D[y] = min [D[y], (c[myself][w] + D_w[y])]  // Bellman-Ford equation
    }
    if (any change in the vector)
      send vector {D[1], D[2], ..., D[N]} to all neighbors
  }
}
```

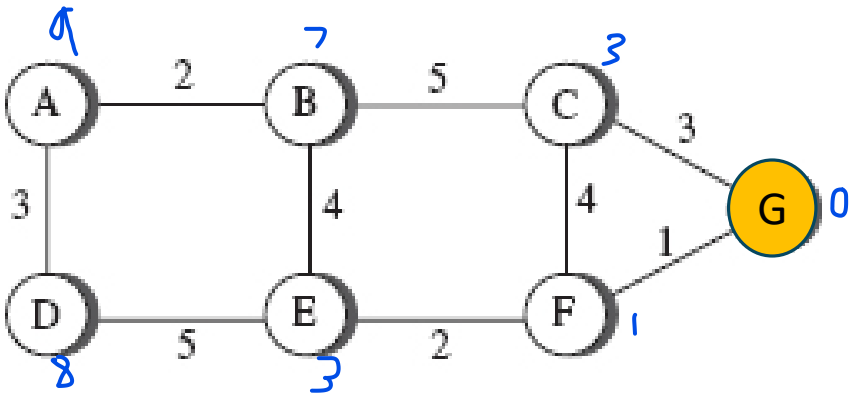
Lines 4 to 11 initialize the vector for the node

Lines 14 to 23 show how the vector can be updated after receiving a vector from the immediate neighbor

The *for loop in lines 17 to 20 allows all entries (cells) in the* vector to be updated after receiving a new vector.

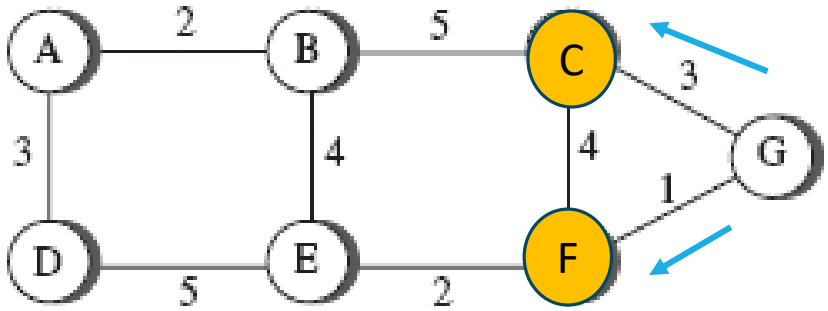
the node sends its vector in line 12, after being initialized, and in line 22, after it is updated

Calculate shortest path for the following network using Bellman ford Algorithm using G as destination



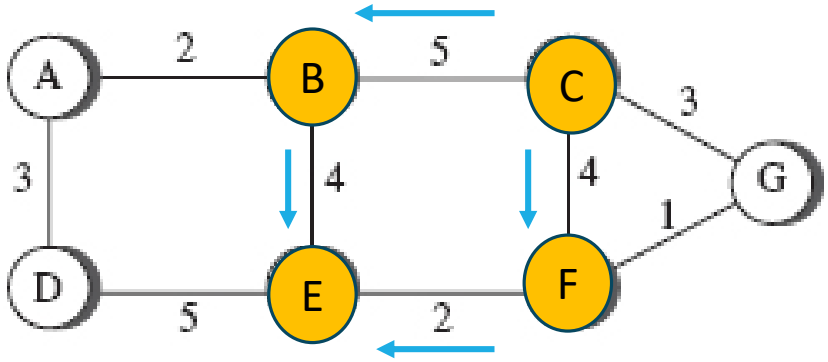
b. The weighted graph

Calculate shortest path for the following network using Bellman ford Algorithm using G as destination



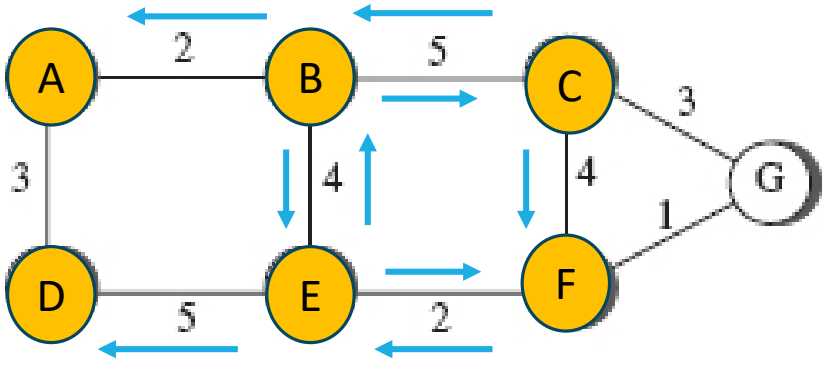
b. The weighted graph

Calculate shortest path for the following network using Bellman ford Algorithm using G as destination



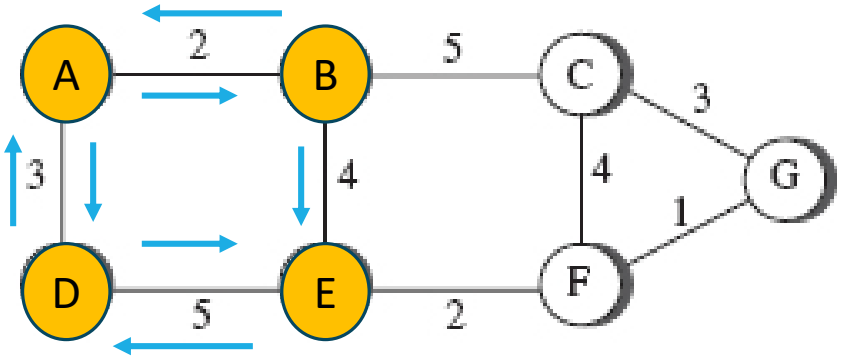
b. The weighted graph

Calculate shortest path for the following network using Bellman ford Algorithm using G as destination



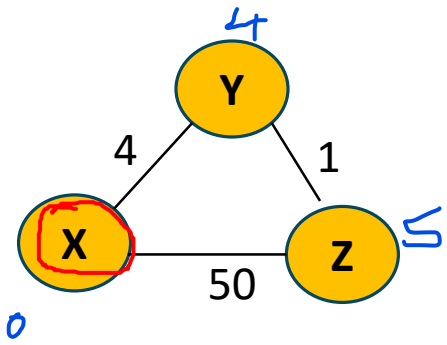
b. The weighted graph

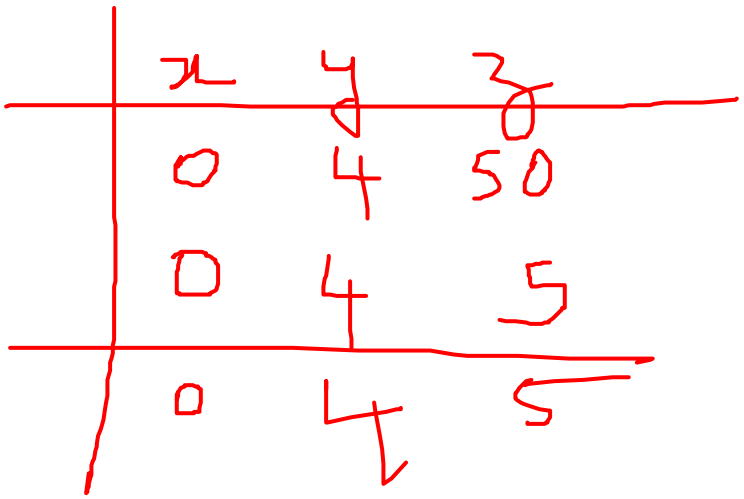
Calculate shortest path for the following network using Bellman ford Algorithm using G as destination



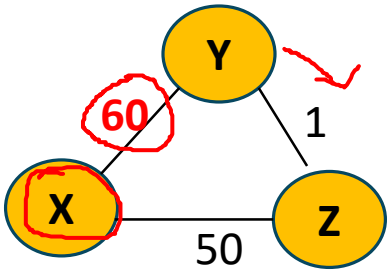
b. The weighted graph

Solve using Bellman ford





Solve using Bellman ford

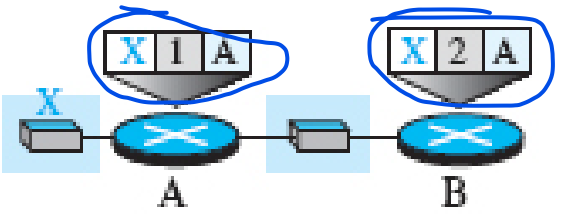


Problem with DVR- *Count to infinity*

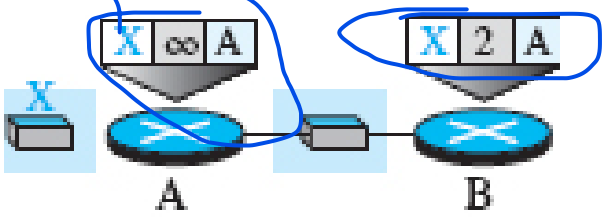
- Any decrease in cost (good news) propagates quickly, but any increase in cost (bad news) will propagate slowly.
- For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance-vector routing, this takes some 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*

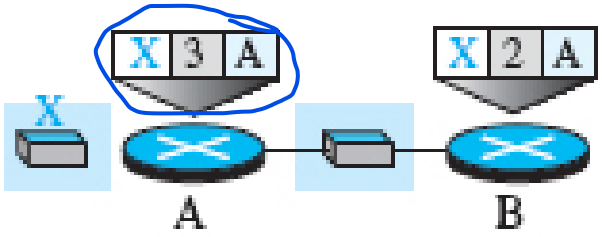
- *Both nodes A and B know how to reach node X*
- *Link between A and X fails. Node A changes its table.*
- *If A can send its table to B immediately, everything is fine.*
  - *However, the system becomes unstable if B sends its forwarding table to A before receiving A's forwarding table.*
  - *Node A receives the update and, assuming that B has found a way to reach X, immediately updates its forwarding table.*



a. Before failure



b. After link failure



c. After A is updated by B

Count to infinity - *Two-Node Loop*

- 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:

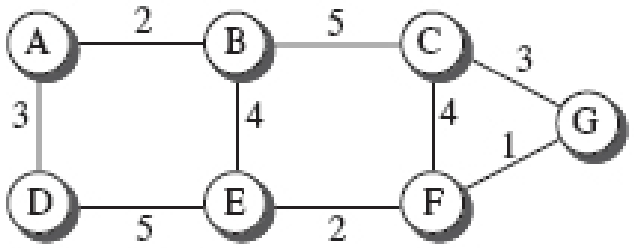
- Normally, the corresponding protocol uses a timer
- If there is no news about a route, the node deletes the route from its table
  - When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the 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

- Directly follows creation of least cost trees and forwarding tables
- *Link state - the characteristic of a link (an edge) that* represents a network in the internet
  - 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)

- To create a least-cost tree, each node needs to have a complete *map of the network, which means it needs to know the state of each* link
- The collection of states for all links - **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

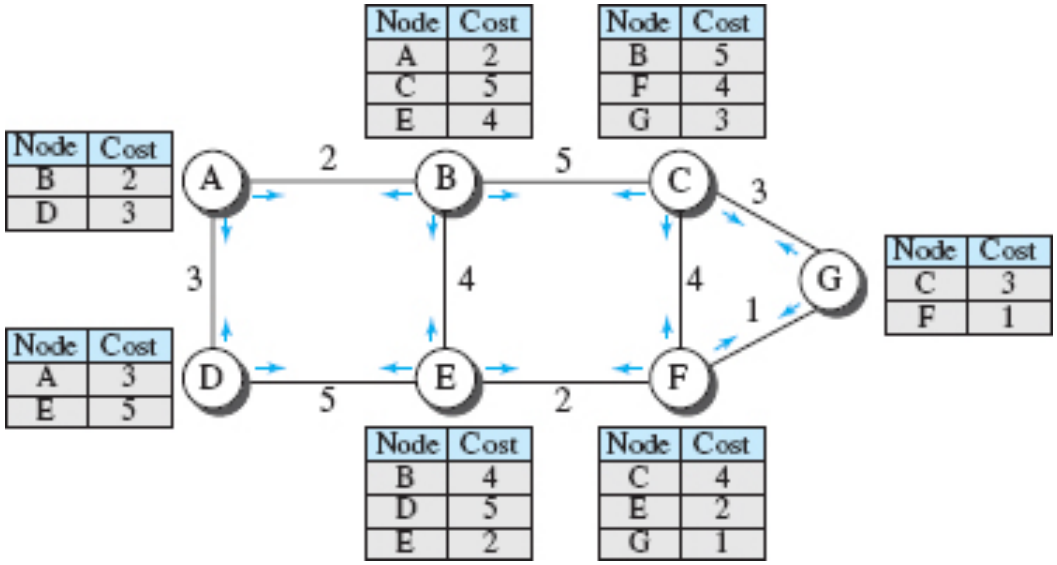
b. Link state database

LSDB creation

"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:
  - the identity of the node
  - the 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



LSDB creation

- When a node receives an LSP from one of its interfaces, it compares the LSP with the copy it may already have.
  - If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP.
  - If it is newer or the first one received, the node discards the old LSP (if there is one) and keeps the received one.
  - It then sends a copy of it out of each interface except the one from which the packet arrived.
    - This guarantees that flooding stops somewhere in the network (where a node has only one interface)
- After receiving all new LSPs, each node creates the comprehensive LSDB, which is same for each node and shows the whole map of the internet

Distance-vector routing algorithm	Link-state routing algorithm
Each router tells its neighbors what it knows about the whole internet	Each router tells the whole internet what it knows about its neighbors.

Formation of Least-Cost Trees

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

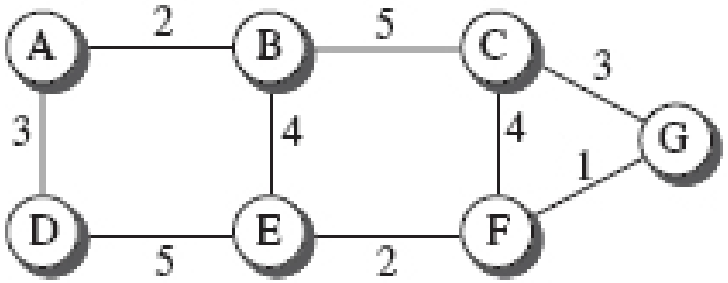
1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB. (Lines 4 to 13)

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. (Lines 16 to 23)

3. The node repeats step 2 until all nodes are added to the tree.

```
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

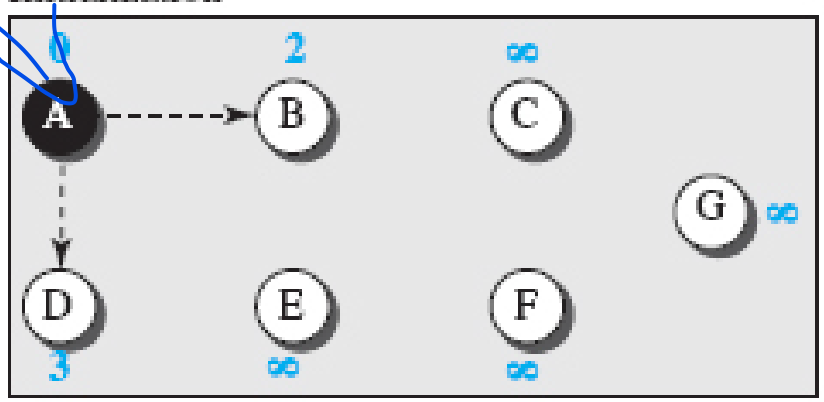


a. The weighted graph

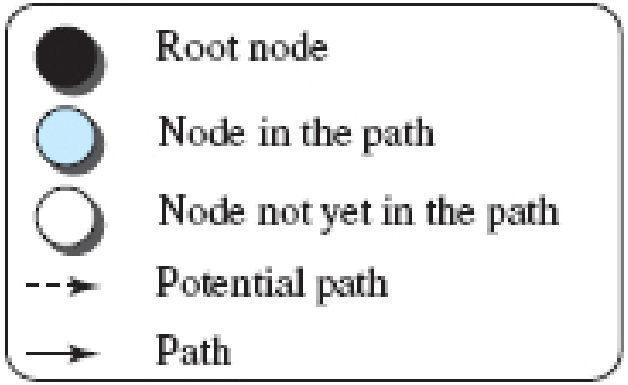
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

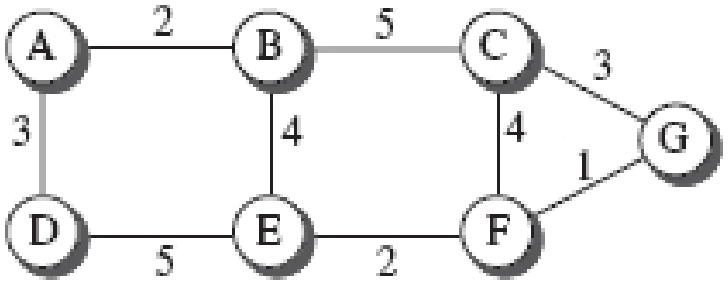
Initialization



Legend



Formation of Least-Cost Trees

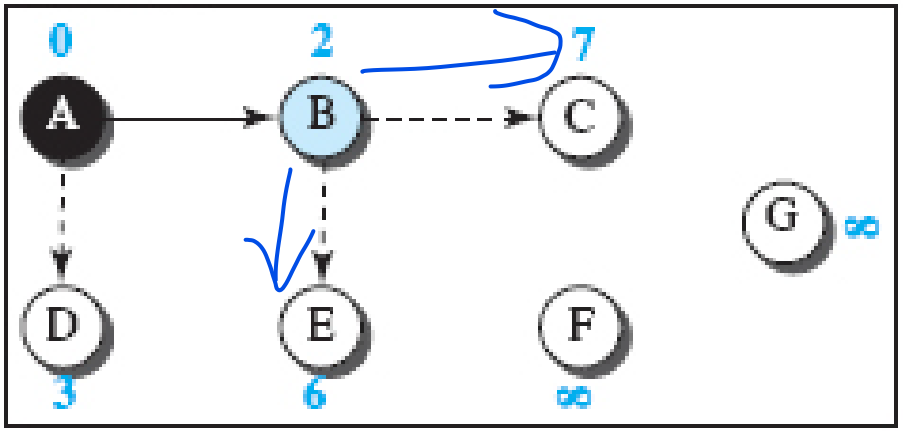


a. The weighted graph

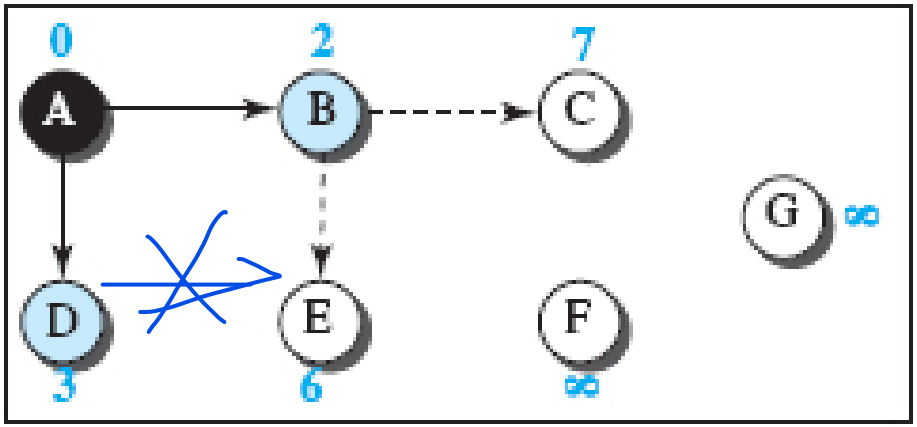
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

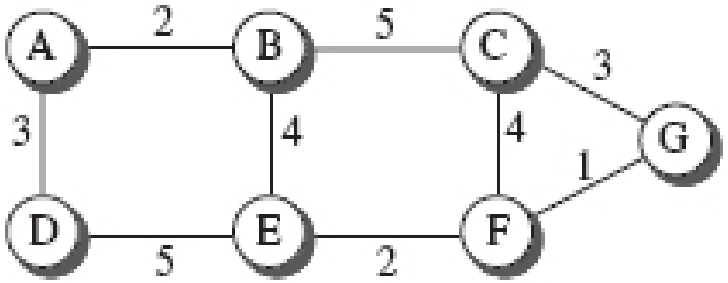
Iteration 1



Iteration 2



Formation of Least-Cost Trees

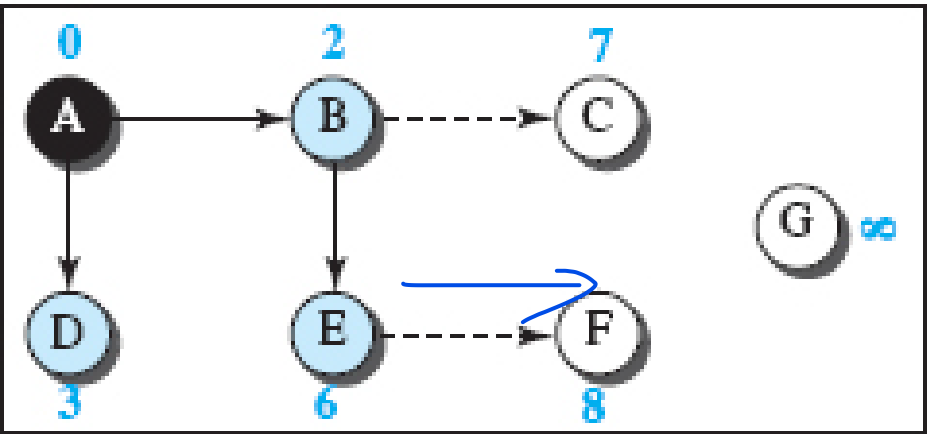


a. The weighted graph

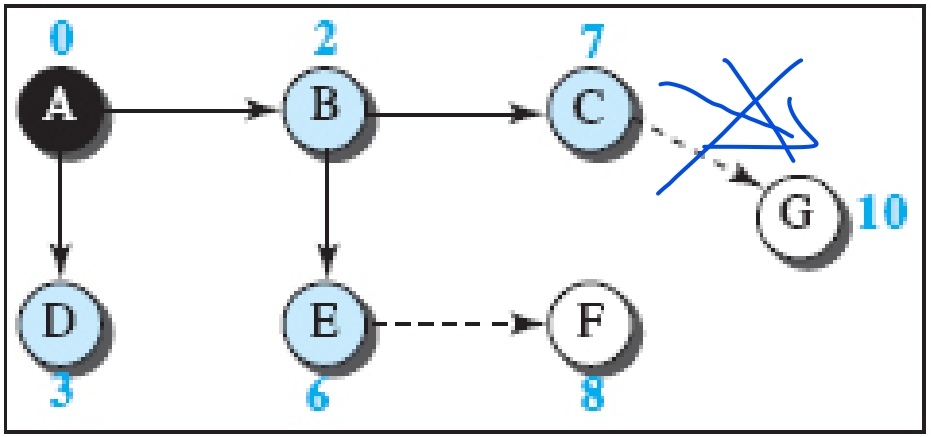
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

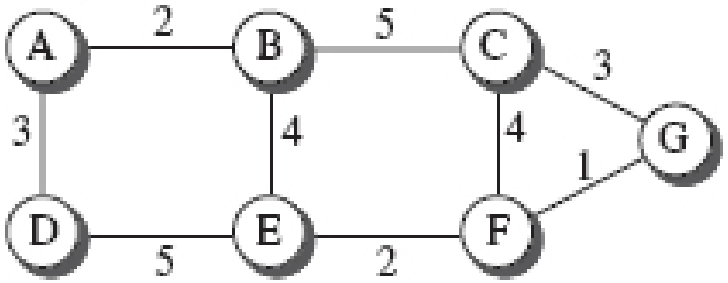
Iteration 3



Iteration 4



Formation of Least-Cost Trees

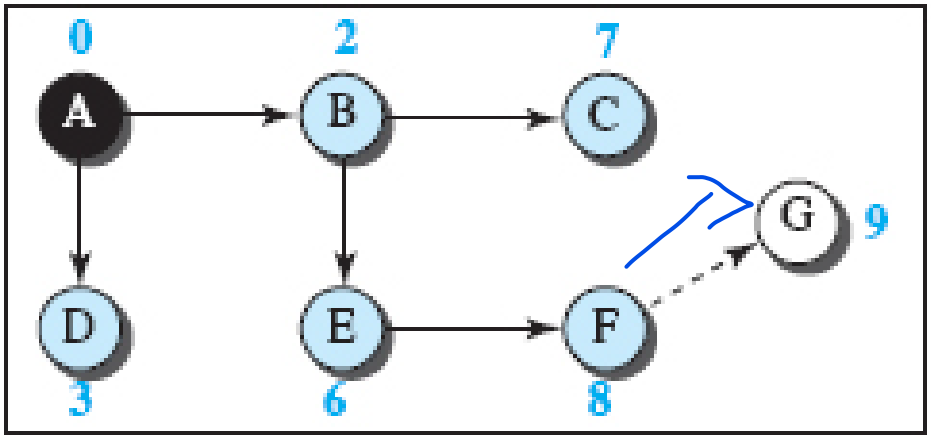


a. The weighted graph

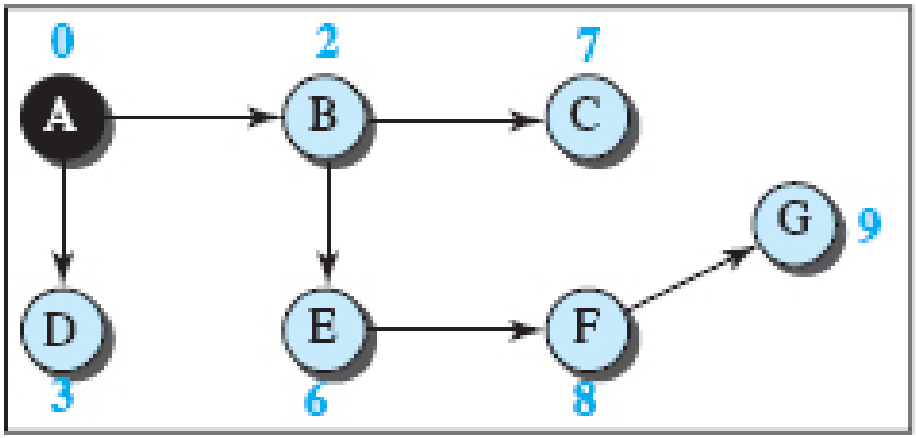
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Iteration 5



Iteration 6



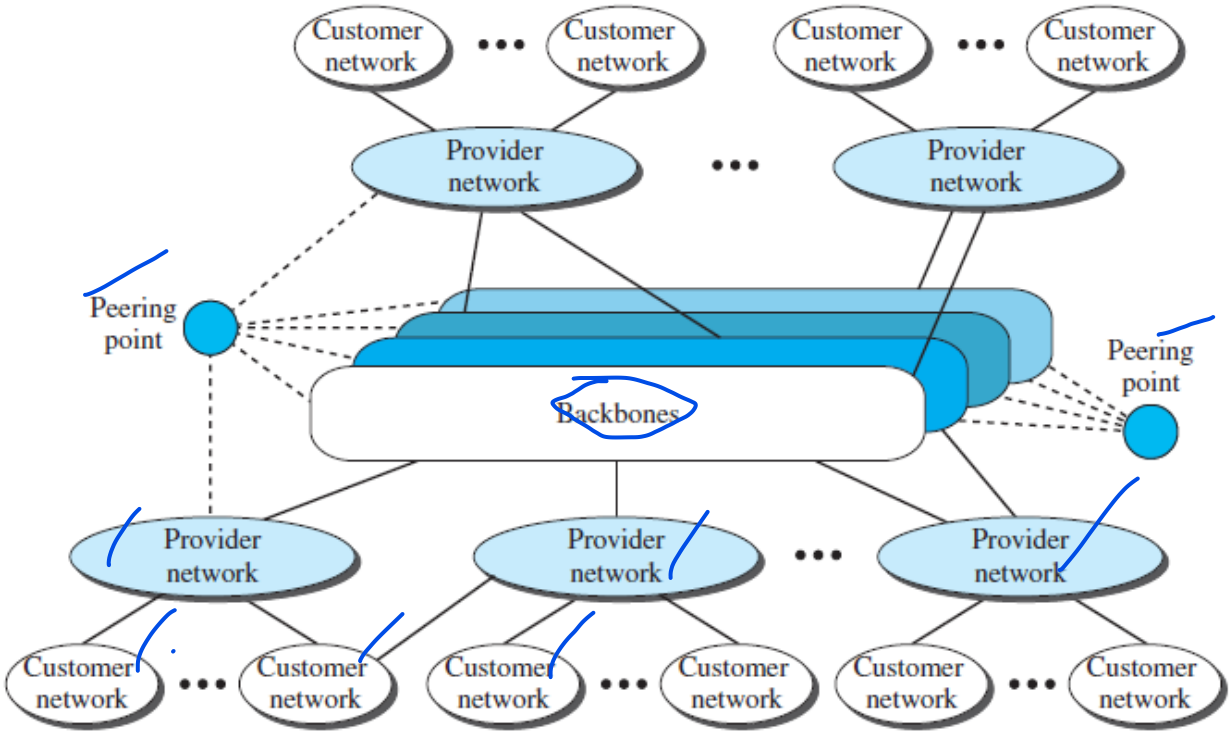
Unicast routing protocols

- 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

- The largest internetwork in the world
- **Internet** (uppercase I), is composed of thousands of interconnected networks

Conceptual (not geographical) view of the Internet



Internet …

- Internet consists of several
  - Backbones (top level)
    - large networks owned by some communication companies such as Sprint, Verizon (MCI), AT&T, and NTT
    - connected through some complex switching systems, called *peering points*
  - provider networks (second level) - smaller networks
    - use the services of the backbones for a fee
    - connected to backbones and sometimes to other provider networks
  - Customer networks (the edge of the Internet)
    - actually use the services provided by the Internet
    - pay fees to provider networks for receiving services
- Backbones and provider networks are also called **Internet Service Providers (ISPs)**
  - Backbones = *international ISPs*
  - provider networks = *national or regional ISPs*

Hierarchical Routing

- The Internet today is made up of a huge number of networks and routers that connect them.
- Routing in the Internet cannot be done using one single protocol for two reasons:
  - a scalability problem - *the size of the* forwarding tables becomes huge, searching for a destination in a forwarding table becomes time consuming, and updating creates a huge amount of traffic
  - an administrative issue- *related to the Internet structure*
    - each ISP is run by an administrative authority
    - The administrator needs to have control in its system.
    - organization must be able to use as many subnets and routers as it needs, may desire that the routers be from a particular manufacturer, may wish to run a specific routing algorithm to meet the needs of the organization, and may want to impose some policy on the traffic passing through its ISP

Hierarchical Routing

- Hierarchical routing means considering each ISP as an **autonomous system (AS)**
- **Each AS can run a routing protocol that meets its needs, but the** global Internet runs a global protocol to glue all ASs together
  - The routing protocol run in each AS is referred to as *intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP);*
    - RIP and OSP
  - *the* global routing protocol is referred to as *inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP)*
    - BGP
- Several intradomain routing protocols -each AS is free to choose one, but only one interdomain protocol that handles routing between these entities

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)

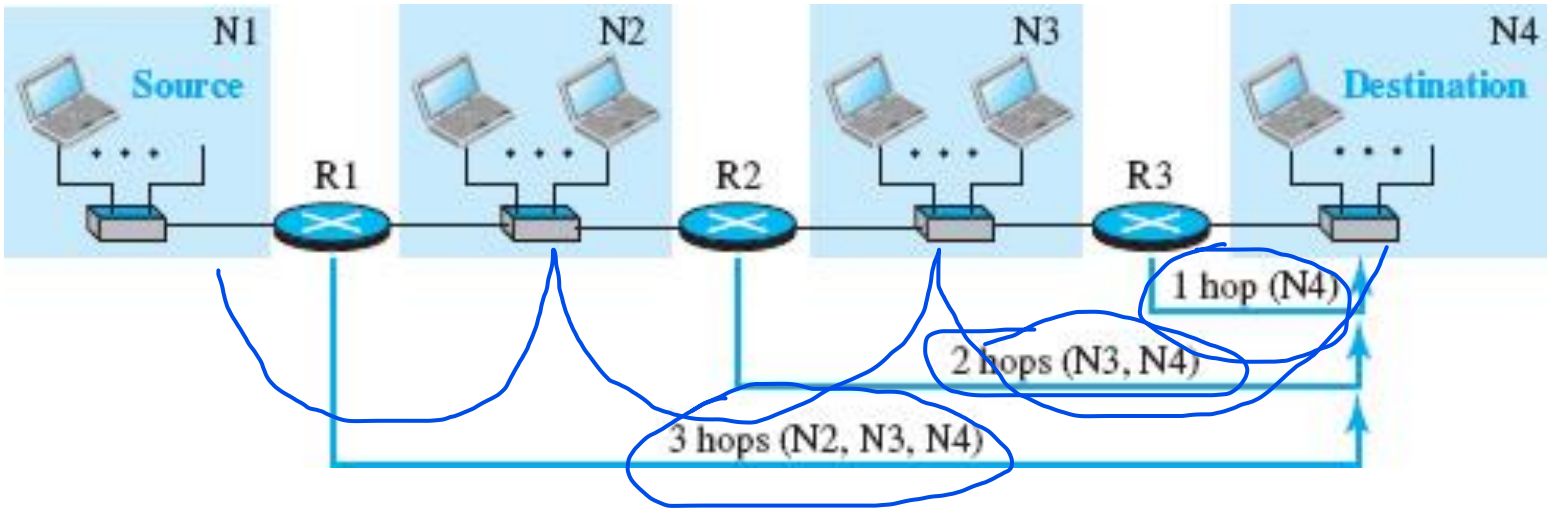
- intradomain routing protocols based on the distance-vector routing algorithm
- started as part of the Xerox Network System (XNS), but it was the Berkeley Software Distribution (BSD) version of UNIX that helped make the use of widespread RIP
- RIP concept:
  - Hop count
  - Forwarding tables
  - Implementation – 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

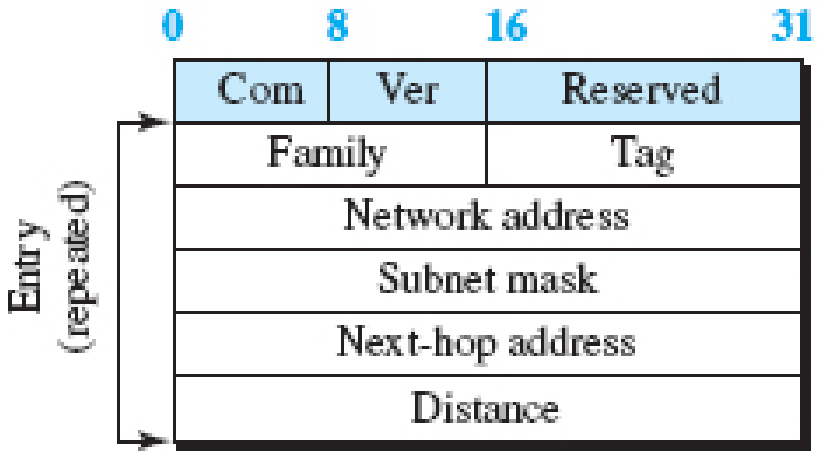
- Although a forwarding table in RIP defines only the next router in the second column, it gives the information about the whole least-cost tree
- For example, R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 is R3; R3 defines that there is no next router 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

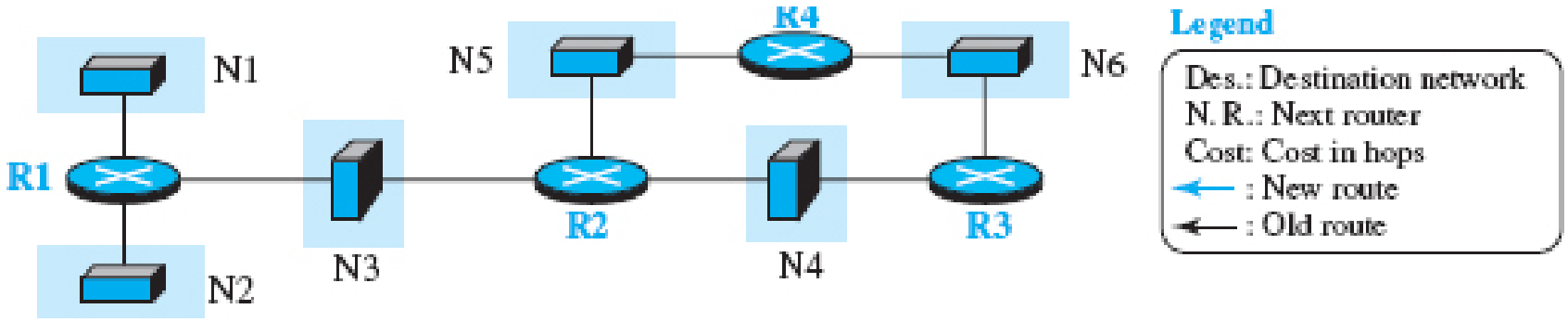
- Same as the distance-vector routing algorithm
- some changes to be made to enable a router to update its forwarding table:
  - Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message.
  - The receiver adds one hop to each cost 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

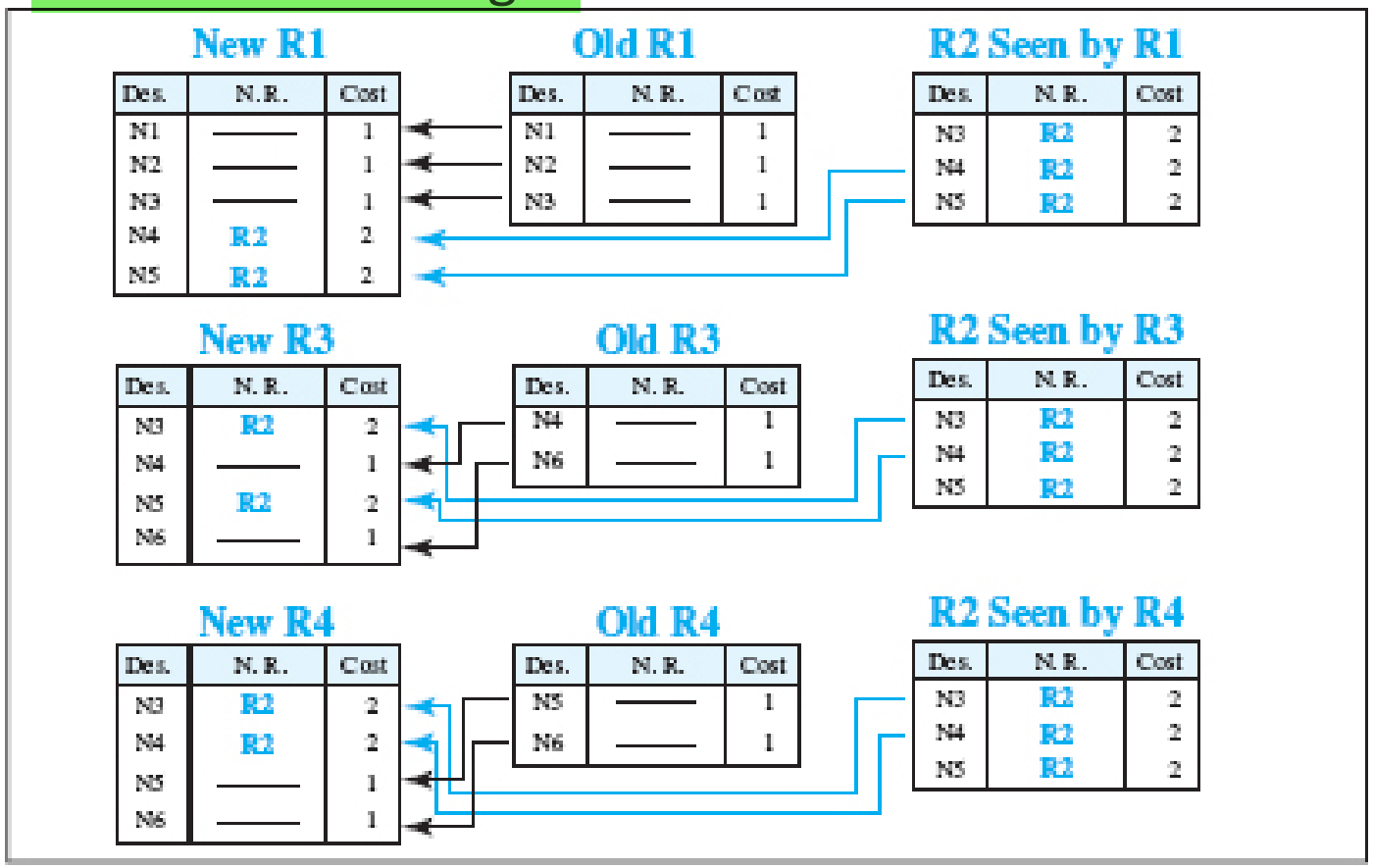
R4

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

- Thestabilized 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

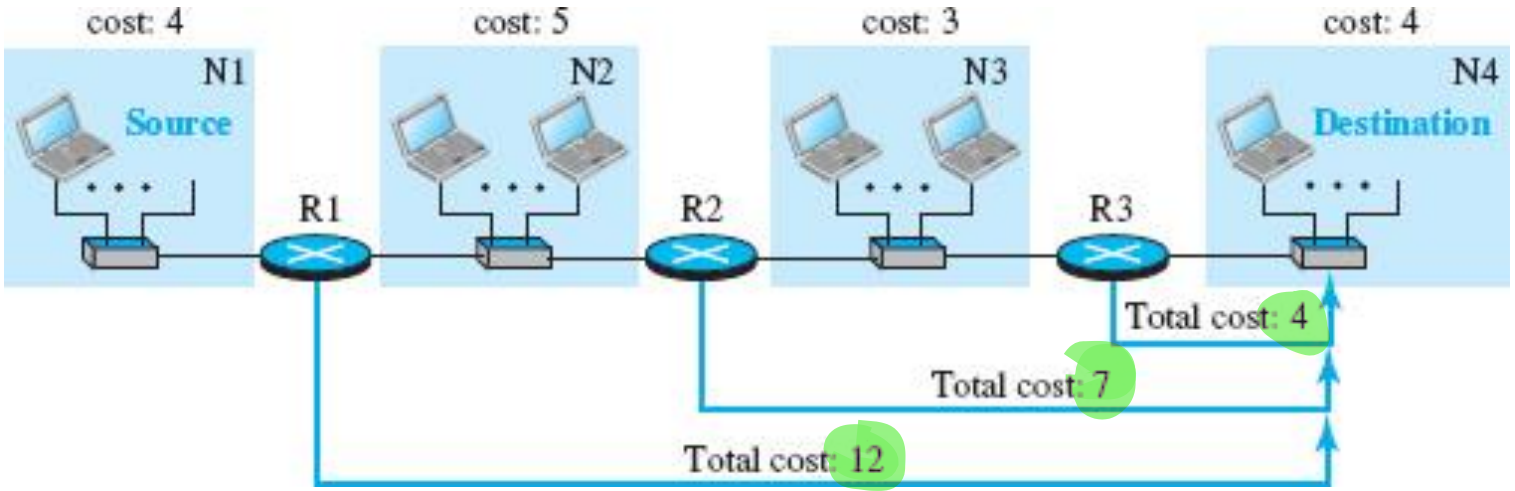
- **Update messages – have a very simple** format and are sent only to neighbors (local)
  - They do not normally create traffic because the routers try to avoid sending them at the same time.
- **Convergence of forwarding tables - the distance-vector** algorithm can converge slowly if the domain is 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

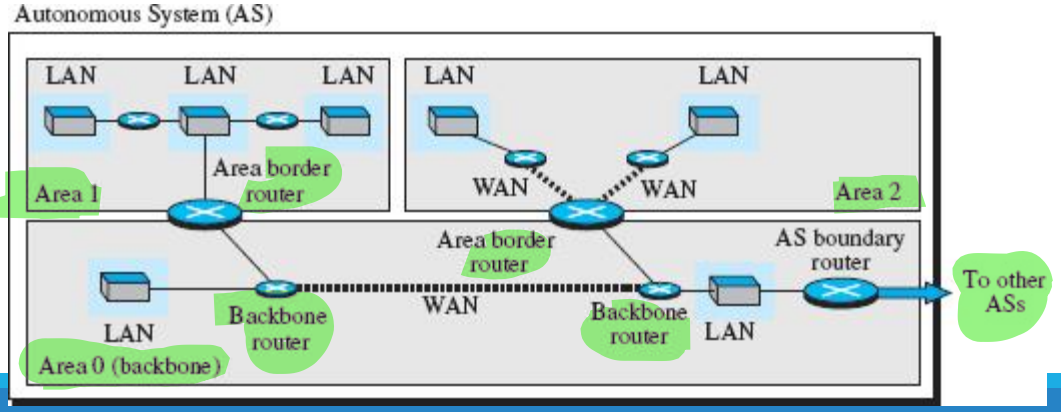
Destination network	Next router	Cost
N1	R1	9
N2	——	5
N3	——	3
N4	R3	7

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.

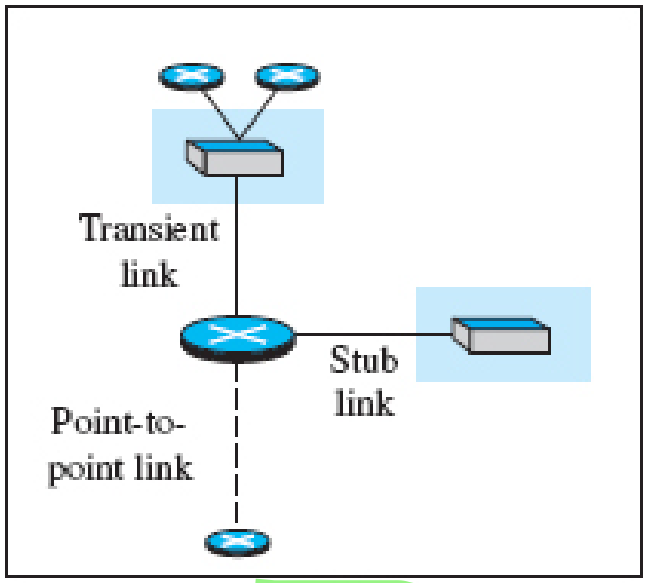


- However, each router in an area needs to know the information about the link states not only in its area but also in other areas.
- For this reason, one of the areas in the AS is designated as the *backbone area, responsible for* gluing the areas together.
- The routers in the backbone area are responsible 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

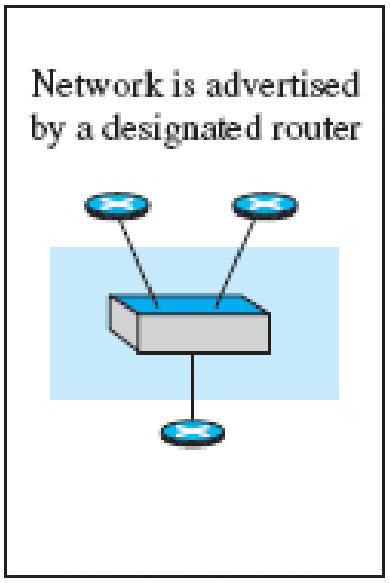
Link-State Advertisement

- 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*

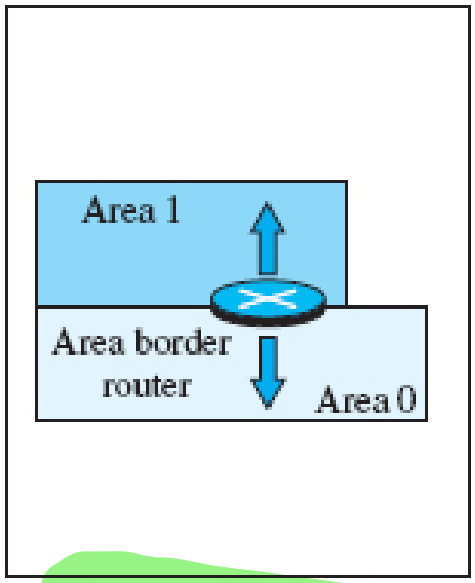
Link-State Advertisement



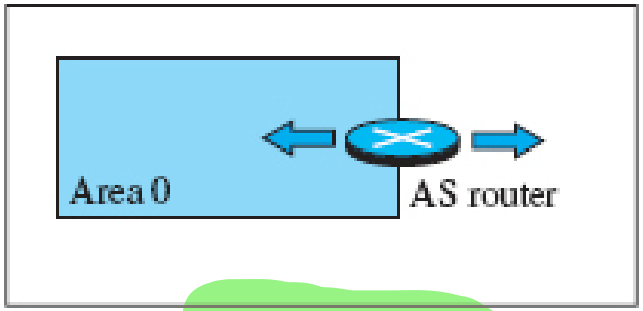
a. Router link



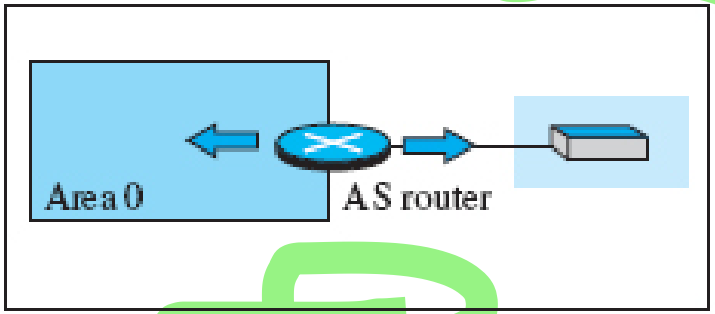
b. Network link



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

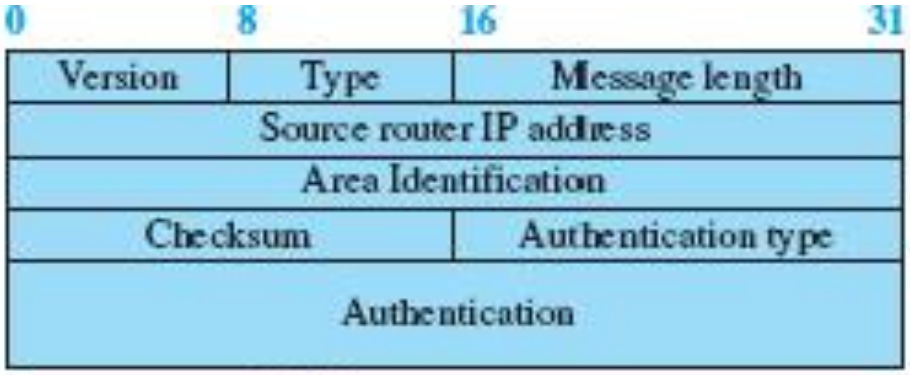
5. **External link -This is also done by an AS router to announce the** existence of a single network outside the AS to the backbone area to be disseminated into the areas.

OSPF Implementation

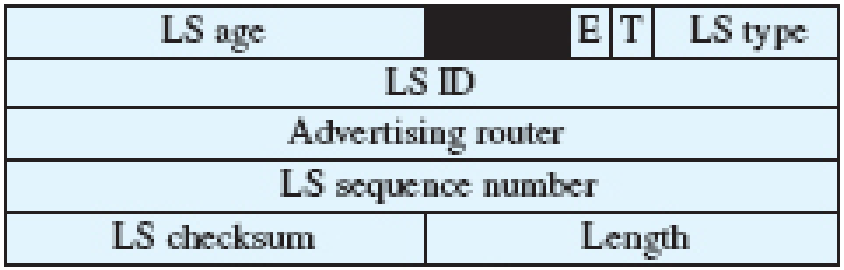
- OSPF is implemented as a program in the network layer that uses the service of the IP for propagation.
- An IP datagram that carries a message from OSPF sets the value of the protocol field to 89.
- This means that, although OSPF is a routing protocol to help IP to route its datagrams inside an AS, 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

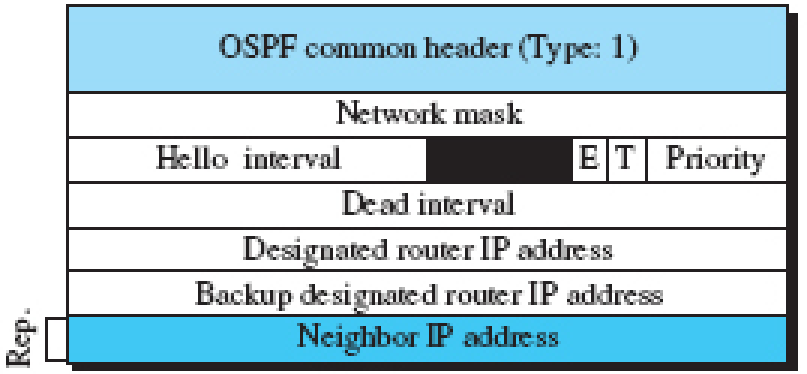
Legend

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

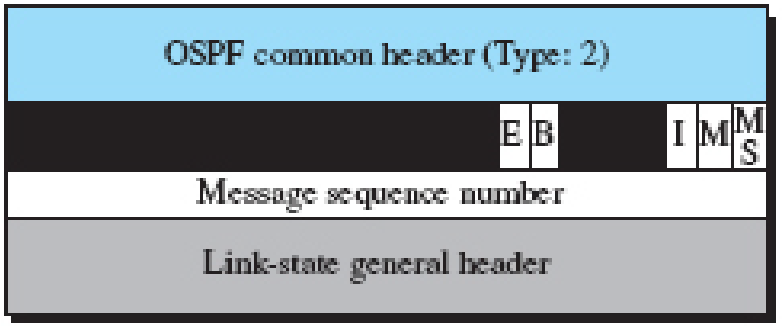
OSPF Messages

Five message types used in OSPF

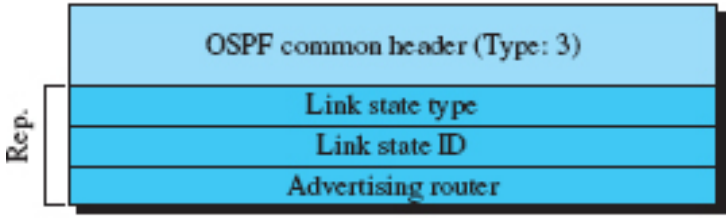
- The *hello message (type 1) is used by a router to introduce itself to* the neighbors and announces all neighbors that it already knows.
- The *database description message (type 2) is normally sent in response to the* hello message to allow a newly joined router to acquire the full LSDB
- The *link-state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



Hello message



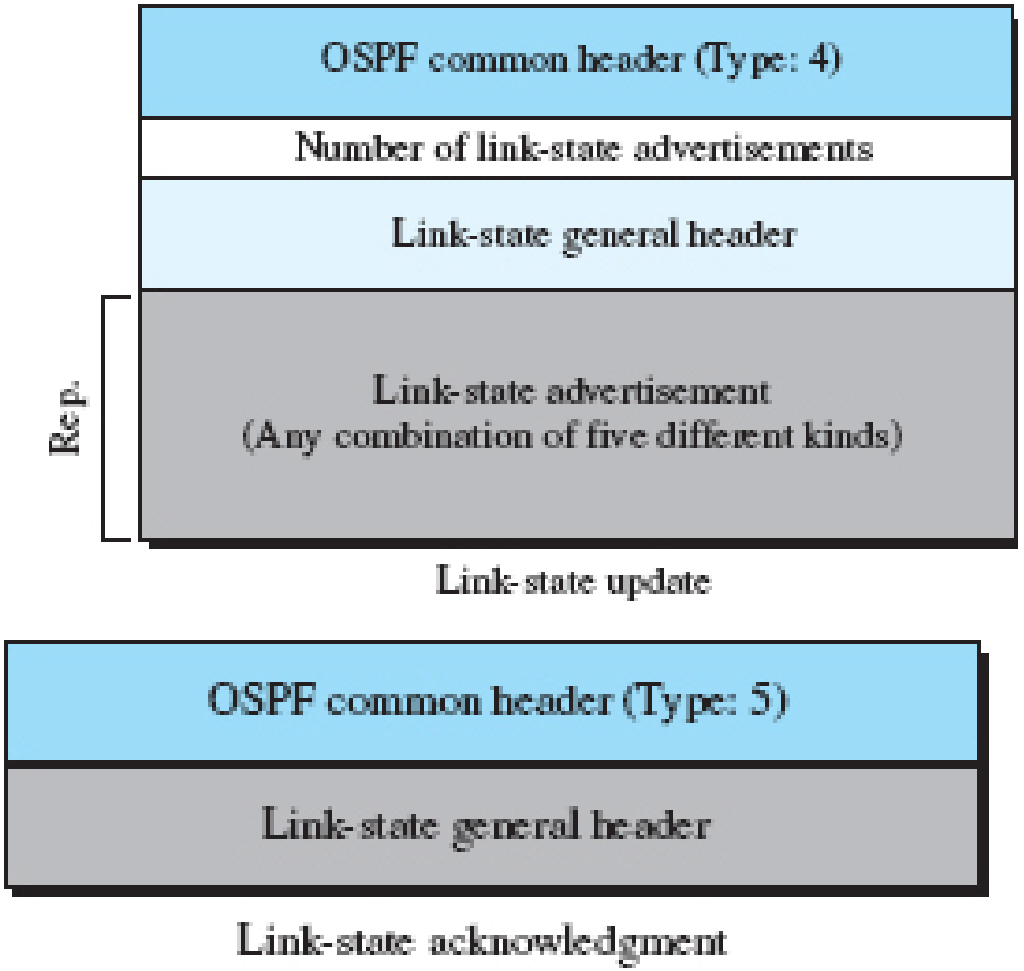
Database description



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.