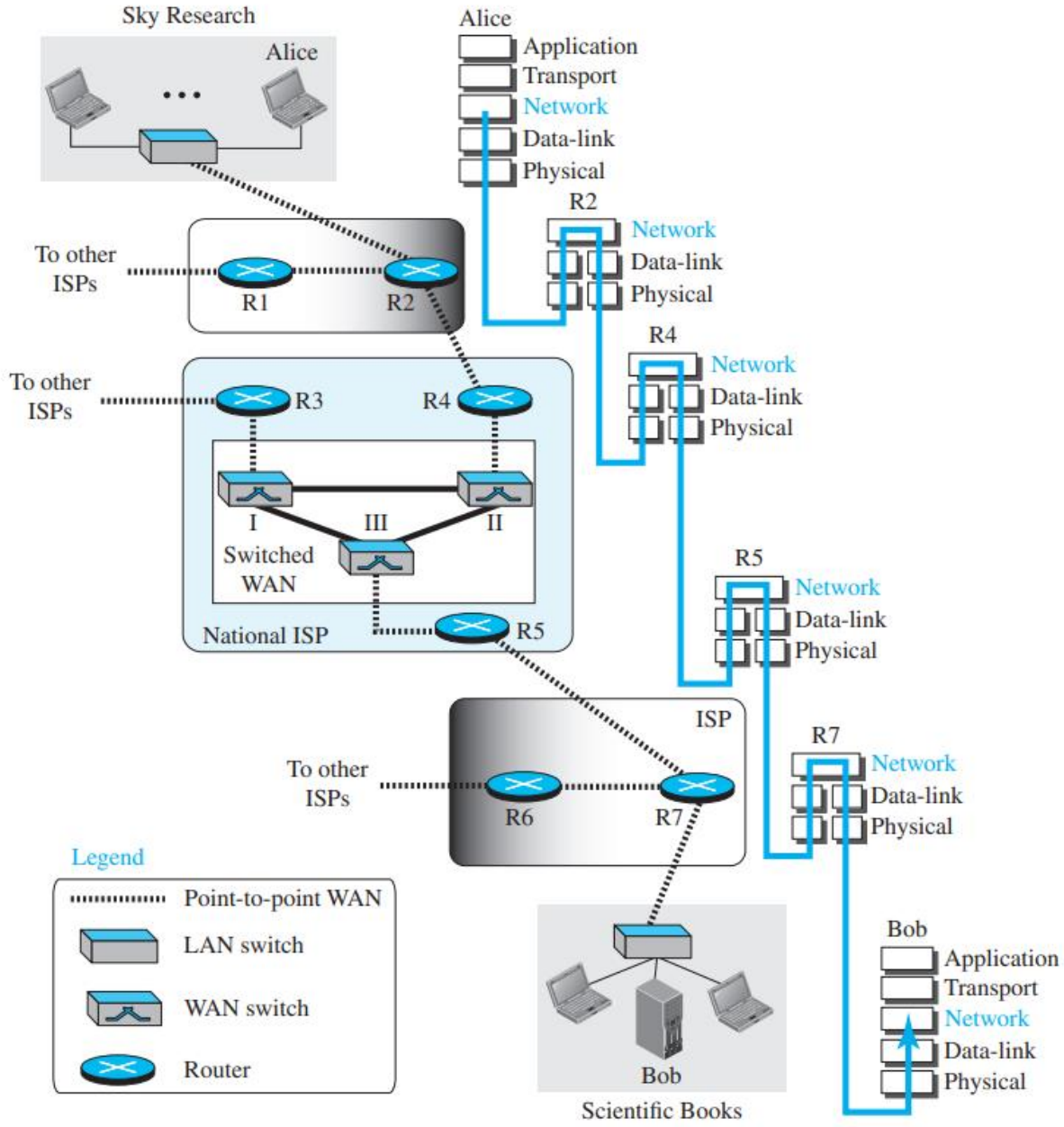
MODULE 3

Network Layer

Network Layer Services:

Figure shows the communication between Alice and Bob at the network layer.



To better understand the role of the network layer (or the internetwork layer), we need to think about the connecting devices (routers or switches) that connect the LANs and WANs.

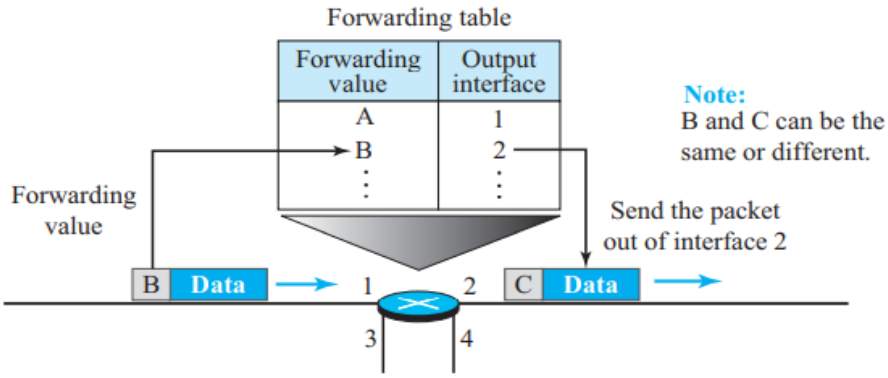
The network layer is involved at the source host, destination host, and all routers in the path (R2, R4, R5, and R7). At the source host (Alice), the network layer accepts a packet from a transport layer, encapsulates the packet in a datagram, and delivers the packet to the data-link layer. At the destination host (Bob), the datagram is decapsulated, and the packet is extracted and delivered to the corresponding transport layer.

Packetizing:

- ✓ Encapsulating the payload (data received from upper layer) in a network-layer packet at the source and decapsulating the payload from the network-layer packet at the destination.
- ✓ One duty of the network layer is to carry a payload from the source to the destination without changing it or using it. The network layer is doing the service of a carrier. The source host receives the payload from an upper-layer protocol, adds a header that contains the source and destination addresses and some other information that is required by the network-layer protocol and delivers the packet to the data-link layer.
- ✓ The source is not allowed to change the content of the payload unless it is too large for delivery and needs to be fragmented. The destination host receives the network-layer packet from its data-link layer, decapsulates the packet, and delivers the payload to the corresponding upper-layer protocol.
- ✓ If the packet is fragmented at the source or at routers along the path, the network layer is responsible for waiting until all fragments arrive, reassembling them, and delivering them to the upper-layer protocol.
- ✓ The routers in the path are not allowed to decapsulate the packets they received unless the packets need to be fragmented. The routers are not allowed to change source and destination addresses either. They just inspect the addresses for the purpose of forwarding the packet to the next network on the path.

Routing and Forwarding:

- ✓ The network layer is responsible for routing the packet from its source to the destination. A physical network is a combination of networks (LANs and WANs) and routers that connect them.
- ✓ The network layer is responsible for finding the best one among these possible routes. The network layer needs to have some specific strategies for defining the best route.
- ✓ This is done by running some routing protocols to help the routers coordinate their knowledge about the neighbourhood and to come up with consistent tables to be used when a packet arrives.
- ✓ If *routing* is applying strategies and running some routing protocols to create the decision-making tables for each router, *forwarding* can be defined as the action applied by each router when a packet arrives at one of its interfaces.
- ✓ The decision-making table a router normally uses for applying this action is sometimes called the forwarding table and sometimes the routing table.
- ✓ When a router receives a packet from one of its attached networks, it needs to forward the packet to another attached network (in unicast routing) or to some attached networks (in multicast routing). To make this decision, the router uses a piece of information in the packet header, which can be the destination address or a label, to find the corresponding output interface number in the forwarding table. The figure shows the idea of the forwarding process in a router.



Other Services

Error Control:

- ✓ Though error control also can be implemented in the network layer, the designers of the network layer in the Internet ignored this issue for the data being carried by the network layer.
- ✓ One reason for this decision is the fact that the packet in the network layer may be fragmented at each router, which makes error checking at this layer inefficient.
- ✓ The designers of the network layer have added a checksum field to the datagram to control any corruption in the header, but not in the whole datagram.
- ✓ This checksum may prevent any changes or corruptions in the header of the datagram.

Flow Control:

- ✓ Flow control regulates the amount of data a source can send without overwhelming the receiver. If the upper layer at the source computer produces data faster than the upper layer at the destination computer can consume it, the receiver will be overwhelmed with data.
- ✓ To control the flow of data, the receiver needs to send some feedback to the sender to inform the latter that it is overwhelmed with data.
- ✓ The network layer in the Internet does not directly provide any flow control. The datagrams are sent by the sender when they are ready, without any attention to the readiness of the receiver.

- ✓ A few reasons for the lack of flow control in the design of the network layer are:

  - ➢ Since there is no error control in this layer, the job of the network layer at the receiver is so simple that it may rarely be overwhelmed.
  - ➢ The upper layers that use the service of the network layer can implement buffers to receive data from the network layer as they are ready and do not have to consume the data as fast as it is received.
  - ➢ Flow control is provided for most of the upper-layer protocols that use the services of the network layer, so another level of flow control makes the network layer more complicated and the whole system less efficient.

Congestion Control:

- ✓ Congestion in the network layer is a situation in which too many datagrams are present in an area of the Internet.
- ✓ Congestion may occur if the number of datagrams sent by source computers is beyond the capacity of the network or routers.

- ✓ In this situation, some routers may drop some of the datagrams. However, as more datagrams are dropped, the situation may become worse because, due to the error control mechanism at the upper layers, the sender may send duplicates of the lost packets.
- ✓ If the congestion continues, sometimes a situation may reach a point where the system collapses, and no datagrams are delivered.

Quality of Service:

- ✓ As the Internet has allowed new applications such as multimedia communication (real-time communication of audio and video), the quality of service (QoS) of the communication has become more and more important.
- ✓ The Internet has thrived by providing better quality of service to support these applications. To keep the network layer untouched, these provisions are mostly implemented in the upper layer.

Security:

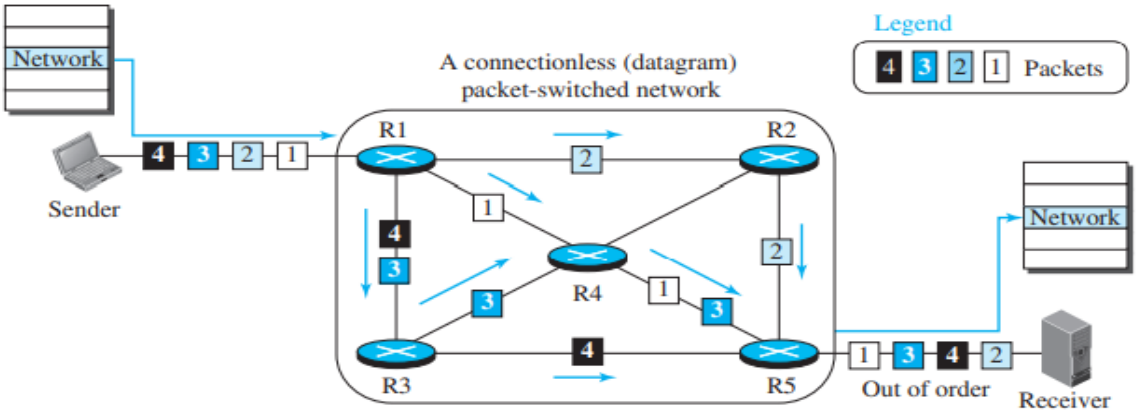
- ✓ Security was not a concern when the Internet was originally designed because it was used by a small number of users at universities for research activities; other people had no access to the Internet.
- ✓ The network layer was designed with no security provision. Today, security is a big concern.
- ✓ To provide security for a connectionless network layer, we need to have another virtual level that changes the connectionless service to a connection-oriented service.

Packet Switching:

- ✓ From the routing and forwarding, we infer that a kind of switching occurs at the network layer. A router, in fact, is a switch that creates a connection between an input port and an output port.
- ✓ In data communication switching techniques are divided into two broad categories, circuit switching and packet switching, only packet switching is used at the network layer because the unit of data at this layer is a packet.
- ✓ At the network layer, a message from the upper layer is divided into manageable packets and each packet is sent through the network. The source of the message sends the packets one by one; the destination of the message receives the packets one by one.
- ✓ The destination waits for all packets belonging to the same message to arrive before delivering the message to the upper layer. The connecting devices in a packet-switched network still need to decide how to route the packets to the destination.
- ✓ Packet-switched network can use two different approaches to route the packets: the datagram approach and the virtual circuit approach.

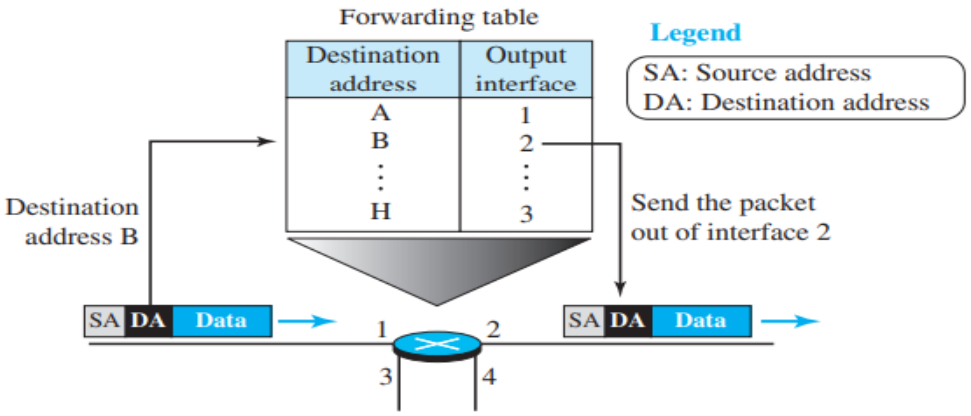
Datagram Approach: Connectionless Service

- ✓ When the Internet started, the network layer was designed to provide a connectionless service in which the network-layer protocol treats each packet independently, with each packet having no relationship to any other packet.
- ✓ The idea was that the network layer is only responsible for delivery of packets from the source to the destination. In this approach, the packets in a message may or may not travel the same path to their destination.



A connectionless packet-switched network

- ✓ Each packet is routed based on the information contained in its header: source and destination addresses. The destination address defines where it should go; the source address defines where it comes from.
- ✓ The router routes the packet based only on the destination address. The source address may be used to send an error message to the source if the packet is discarded. The figure shows the forwarding process in a router in this case.

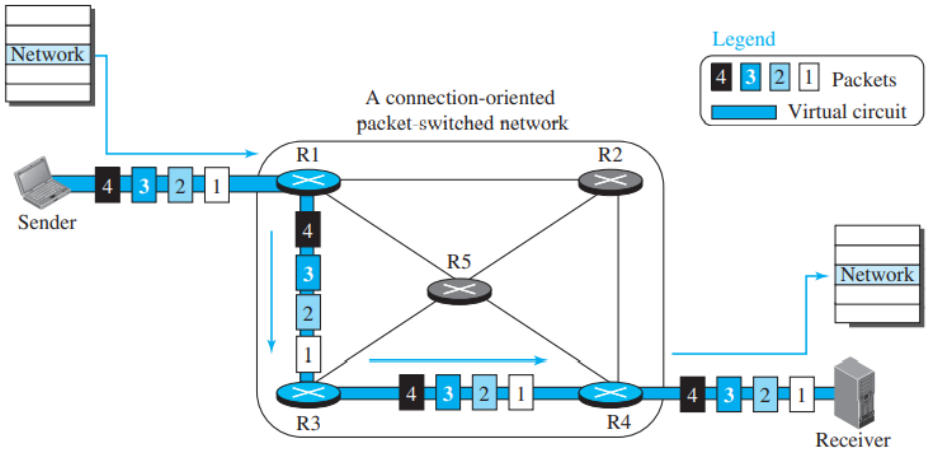


Forwarding process in a router when used in a connectionless network

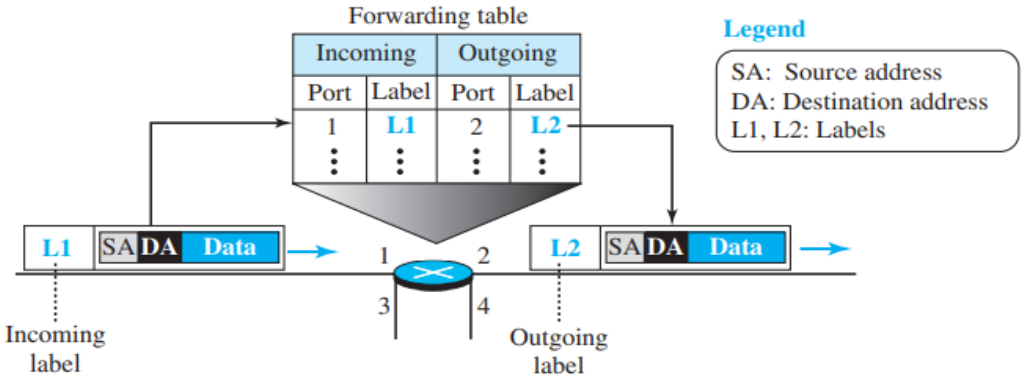
Virtual-Circuit Approach: Connection-Oriented Service

- ✓ In a connection-oriented service (also called virtual-circuit approach), there is a relationship between all packets belonging to a message. Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams can all follow the same path.
- ✓ In this type of service, not only must the packet contain the source and destination addresses, it must also contain a flow label, a virtual circuit identifier that defines the virtual path the packet should follow.
- ✓ Although it looks as though the use of the label may make the source and destination addresses unnecessary during the data transfer phase, parts of the Internet at the network layer keep these addresses.

- ✓ One reason is that part of the packet path may still be using the connectionless service. Another reason is that the protocol at the network layer is designed with these addresses, and it may take a while before they can be changed. Figure shows the concept of connection-oriented service.



- ✓ Each packet is forwarded based on the label in the packet. To follow the idea of connection-oriented design to be used in the Internet, we assume that the packet has a label when it reaches the router.



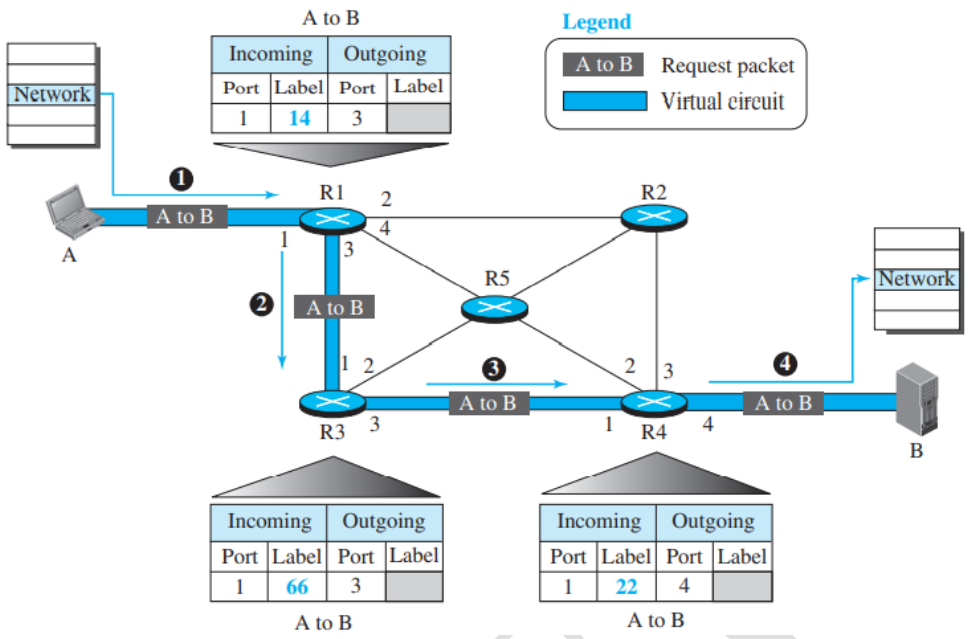
Forwarding process in a router when used in a virtual-circuit network

- ✓ In this case, the forwarding decision is based on the value of the label, or virtual circuit identifier, as it is sometimes called.
- ✓ To create a connection-oriented service, a three-phase process is used: setup, data transfer, and teardown. In the setup phase, the source and destination address of the sender and receiver are used to make table entries for the connection-oriented service. In the teardown phase, the source and destination inform the router to delete the corresponding entries. Data transfer occurs between these two phases.

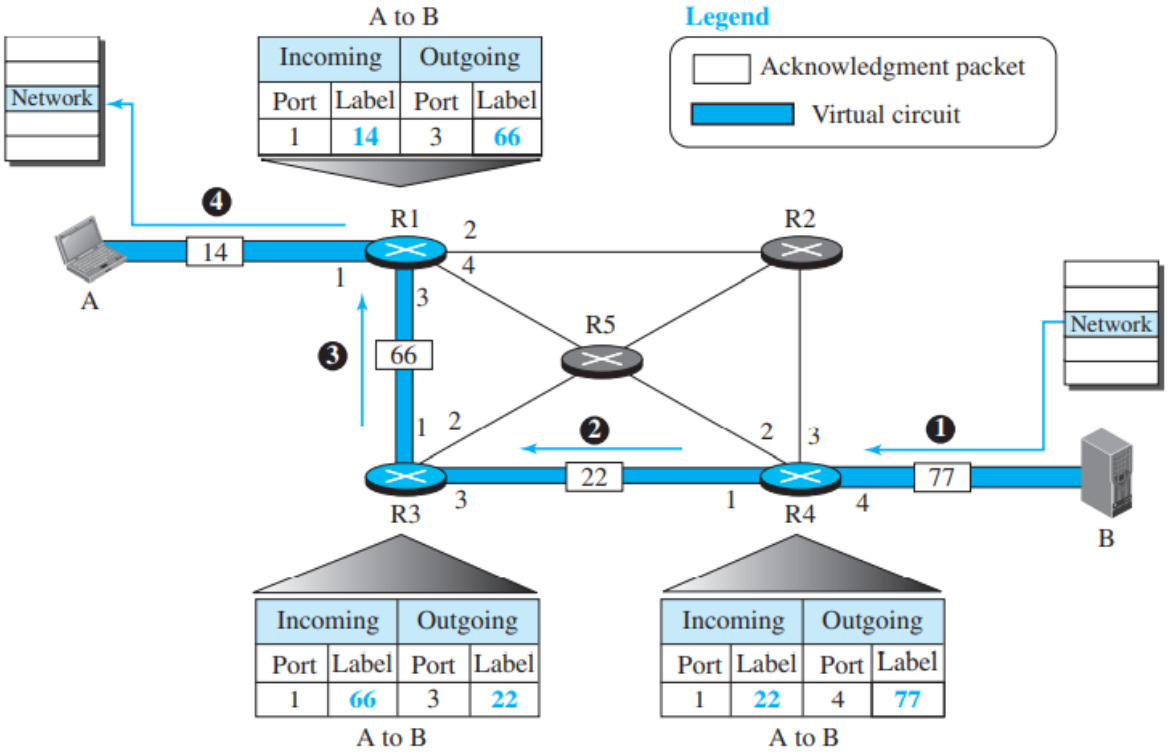
Setup Phase

In the setup phase, a router creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to destination B. Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.

Request packet: A request packet is sent from the source to the destination. This auxiliary packet carries the source and destination addresses. The figure shows this process.



- Source A sends a request packet to router R1.
- Router R1 receives the request packet. It knows that a packet going from A to B goes out through port 3. How the router has obtained this information is a point covered later. For the moment, assume that it knows the output port. The router creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The router assigns the incoming port (1) and chooses an available incoming label (14) and the outgoing port (3). It does not yet know the outgoing label, which will be found during the acknowledgment step. The router then forwards the packet through port 3 to router R3.
- Router R3 receives the setup request packet. The same events happen here as at router R1; three columns of the table are completed: in this case, incoming port (1), incoming label (66), and outgoing port (3).
- Router R4 receives the setup request packet. Again, three columns are completed: incoming port (1), incoming label (22), and outgoing port (4).
- Destination B receives the setup packet, and if it is ready to receive packets from A, it assigns a label to the incoming packets that come from A, in this case 77, as shown in Figure. This label lets the destination know that the packets come from A, and not from other sources.
- Acknowledgment Packet: A special packet, called the acknowledgment packet, completes the entries in the switching tables. Figure shows the process.

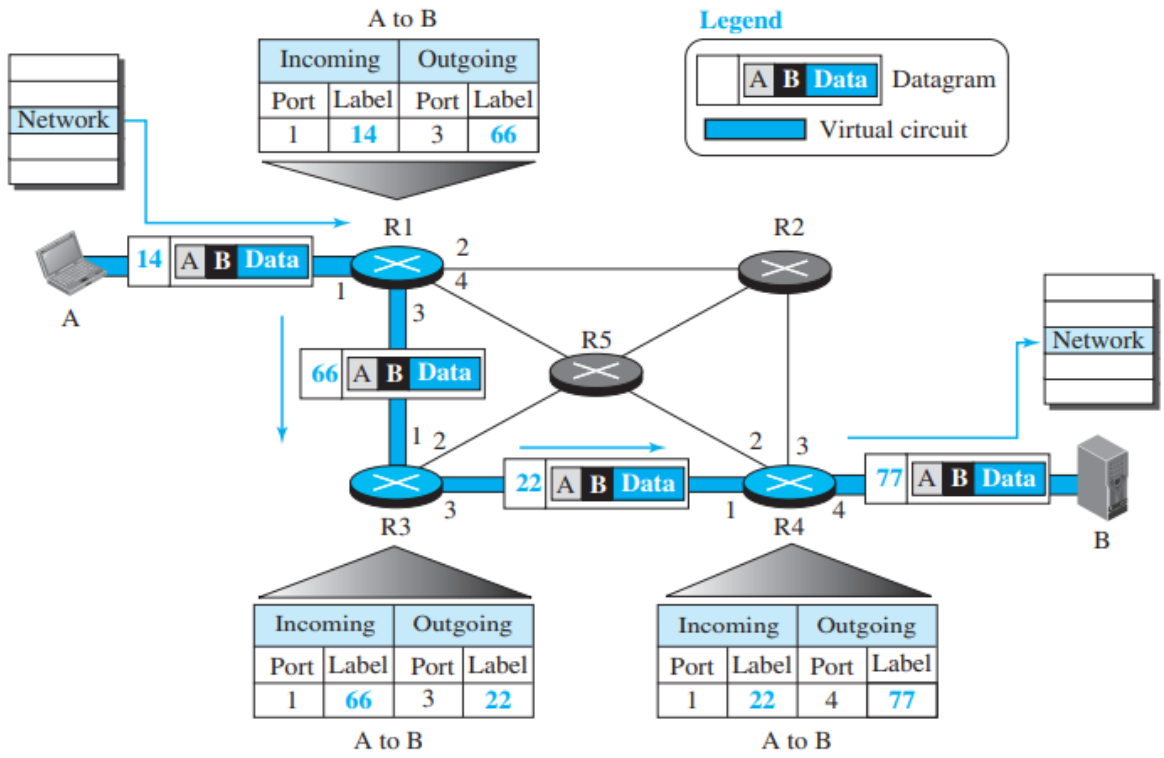


Sending acknowledgments in a virtual-circuit network

- The destination sends an acknowledgment to router R4. The acknowledgment carries the global source and destination addresses so the router knows which entry in the table is to be completed. The packet also carries label 77, chosen by the destination as the incoming label for packets from A. Router R4 uses this label to complete the outgoing label column for this entry. Note that 77 is the incoming label for destination B, but the outgoing label for router R4.
- Router R4 sends an acknowledgment to router R3 that contains its incoming label in the table, chosen in the setup phase. Router R3 uses this as the outgoing label in the table.
- Router R3 sends an acknowledgment to router R1 that contains its incoming label in the table, chosen in the setup phase. Router R1 uses this as the outgoing label in the table.
- Finally, router R1 sends an acknowledgment to source A that contains its incoming label in the table, chosen in the setup phase.
- The source uses this as the outgoing label for the data packets to be sent to destination B.

Data-Transfer Phase

The second phase is called the data-transfer phase. After all routers have created their forwarding table for a specific virtual circuit, then the network-layer packets belonging to one message can be sent one after another. The figure shows the flow of a single packet, but the process is the same for 1, 2, or 100 packets. The source computer uses the label 14, which it has received from router R1 in the setup phase. Router R1 forwards the packet to router R3 but changes the label to 66. Router R3 forwards the packet to router R4 but changes the label to 22. Finally, router R4 delivers the packet to its destination with the label 77. All the packets in the message follow the same sequence of labels, and the packets arrive in order at the destination.



Flow of one packet in an established virtual circuit

Teardown Phase

In the teardown phase, source A, after sending all packets to B, sends a special packet called a teardown packet. Destination B responds with a confirmation packet. All routers delete the corresponding entries from their tables.

IPV4 Addresses:

- ✓ The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address.
- ✓ An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is 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.
- ✓ IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses.
- ✓ IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space:

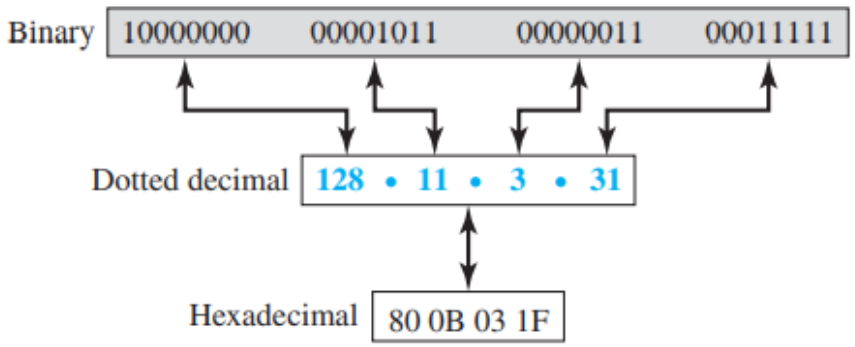
- ✓ A protocol like IPv4 that defines addresses has an address space. An address space is the total number of addresses used by the protocol.
- ✓ If a protocol uses b bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1).

- ✓ IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation:

- There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16).
- In binary notation, an IPv4 address is displayed as 32 bits. 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.
- 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. This format is referred to as dotted-decimal notation.
- Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255.
- Sometimes IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming.

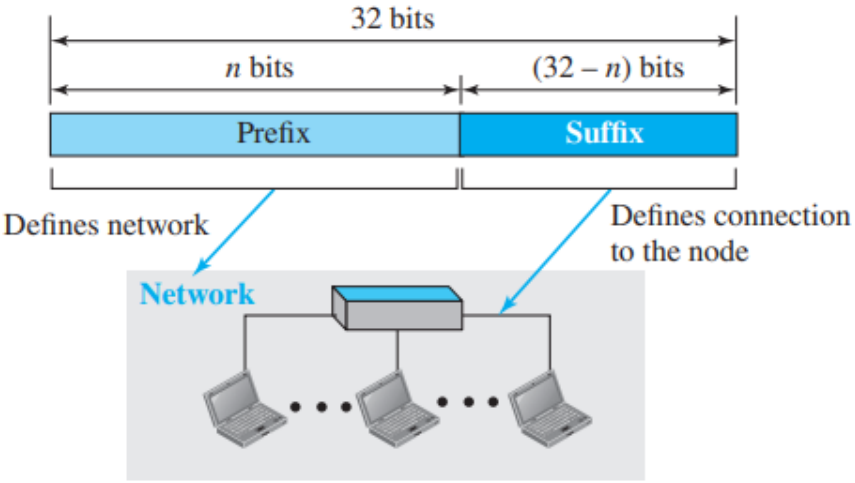
Figure shows an IP address in the three discussed notations.



Three different notations in IPv4 addressing

Hierarchy in Addressing:

- In any communication network that involves delivery, such as a telephone network or a postal network, the addressing system is hierarchical.
- In a postal network, the postal address (mailing address) includes the country, state, city, street, house number, and the name of the mail recipient.
- Similarly, a telephone number is divided into the country code, area code, local exchange, and the connection.
- A 32-bit IPv4 address is also hierarchical but 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).
- Figure shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is n bits, and the suffix length is (32 − n) bits.

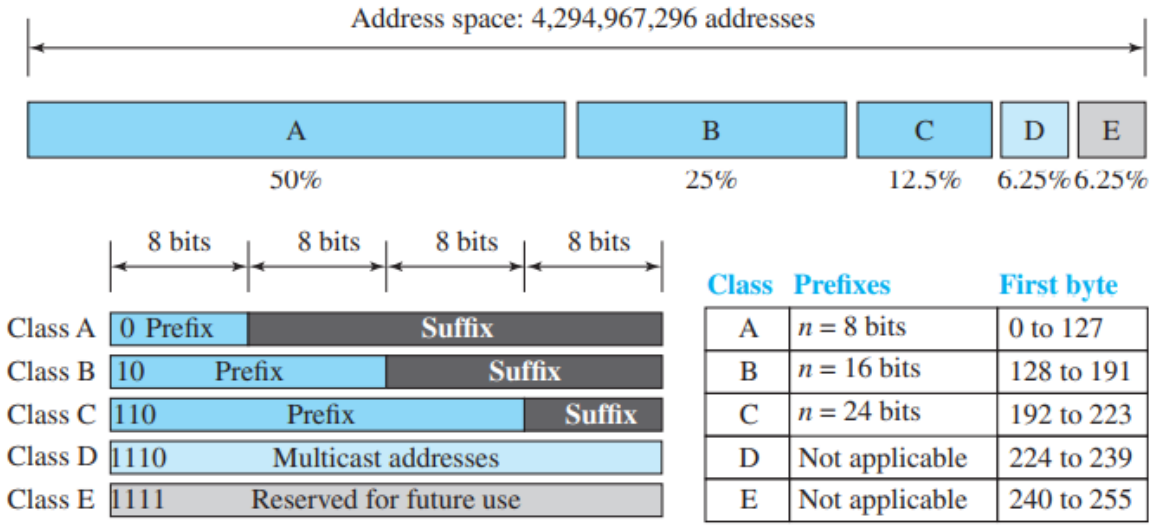


Hierarchy in addressing

- A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing.
- The new scheme, which is referred to as classless addressing, uses a variable-length network prefix. First, we briefly discuss classful addressing; then we concentrate on classless addressing.

Classful Addressing

- ✓ 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 (class A, B, C, D, and E), as shown in Figure. This scheme is referred to as classful addressing.
- ✓ In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only 2^7 = 128 networks in the world that can have a class A address.
- ✓ In class B, the network length is 16 bits, but since the first two 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.
- ✓ All addresses that start with $(110)_2$ belong to class C. In class C, the network length is 24 bits, but since three 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	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

Occupation of the address space in classful addressing

✓ 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.

1. Address Depletion:

- ❖ The reason that classful addressing has become obsolete is address depletion.
- ❖ Since the addresses were not distributed properly, the Internet was faced with the problem of the addresses being rapidly used up, resulting in no more addresses available for organizations and individuals that needed to be connected to the Internet.
- ❖ To understand the problem, let us think about class A. This class can be assigned to only 128 organizations in the world, but each organization needs to have a single network (seen by the rest of the world) with 16,777,216 nodes (computers in this single network).
- ❖ Since there may be 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 flaw in design. 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.
- ❖ Class E addresses were almost never used, wasting the whole class.

2. Subnetting and Supernetting:

- ❖ To alleviate address depletion, two strategies were proposed and, to some extent, implemented: Subnetting and Supernetting.
- ❖ In subnetting, a class A or class B block is divided into several subnets. Each subnet has a larger prefix length than the original network.

- For example, if a network in class A is divided into four subnets, each subnet has a prefix of n_{sub} = 10. At the same time, if all the addresses in a network are not used, subnetting allows the addresses to be divided among several organizations.
- This idea did not work because most large organizations were not happy about dividing the block and giving some of the unused addresses to smaller organizations.
- While subnetting was devised to divide a large block into smaller ones, supernetting was devised to combine several class C blocks into a larger block to be attractive to organizations that need more than the 256 addresses available in a class C block. This idea did not work either because it makes the routing of packets more difficult.

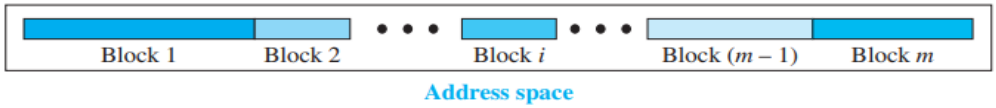
3. Advantage of Classful Addressing:

- Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately.
- The prefix length in classful addressing is inherent in the address; no extra information is needed to extract the prefix and the suffix.

Classless Addressing

- ✓ Subnetting and supernetting in classful addressing did not really solve the address depletion problem.
- ✓ With the growth of the Internet, it was clear that a larger address space was needed as a long-term solution.
- ✓ The larger address space requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed.
- ✓ Although the long-range solution has already been devised and is called IPv6, a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization.
- ✓ The short-term solution still uses IPv4 addresses, but it is called classless addressing. The class privilege was removed from the distribution to compensate for the address depletion.
- ✓ 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.
- ✓ In classless addressing, the whole address space is divided into variable length blocks. The prefix in an address defines the block (network); the suffix defines the node (device).

Figure shows the division of the whole address space into nonoverlapping blocks.



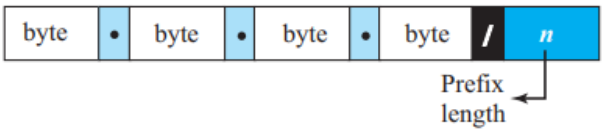
Variable-length blocks in classless addressing

- ✓ Unlike classful addressing, the prefix length in classless addressing is variable. We can have a prefix length that ranges from 0 to 32.
- ✓ The size of the network is inversely proportional to the length of the prefix. A small prefix means a larger network; a large prefix means a smaller network.

- ✓ We need to emphasize that the idea of classless addressing can be easily applied to classful addressing. An address in class A can be thought of as a classless address in which the prefix length is 8.
- ✓ An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

1. Prefix Length: Slash Notation:

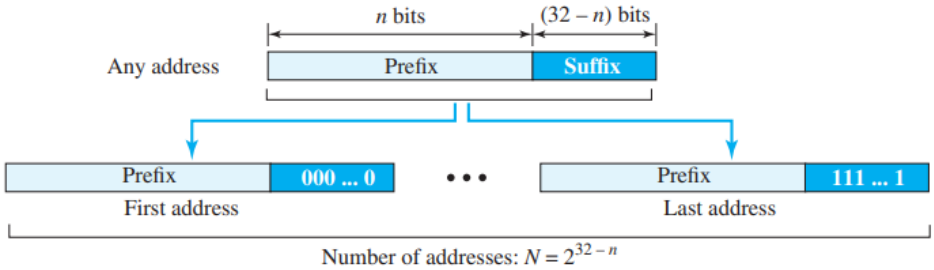
   - ❖ Since the prefix length is not inherent in the address, we need to separately give the length of the prefix. The prefix length, n, is added to the address, separated by a slash. The notation is informally referred to as slash notation and formally as classless interdomain routing or CIDR.
   - ❖ An address in classless addressing can then be represented as shown in figure



Examples:
12.24.76.8/8
23.14.67.92/12
220.8.24.255/25

2. 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, and the last address.
   - ❖ Since the value of prefix length, n, is given, we can easily find these three pieces of information, as shown in Figure.
   - ❖ The number of addresses in the block is found as N = 232−n .
   - ❖ To find the first address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 0s.
   - ❖ To find the last address, we keep the n leftmost bits and set the (32 − n) rightmost bits all to 1s.



   - ❖ Example: A classless address is given as 167.199.170.82/27. We can find the above 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

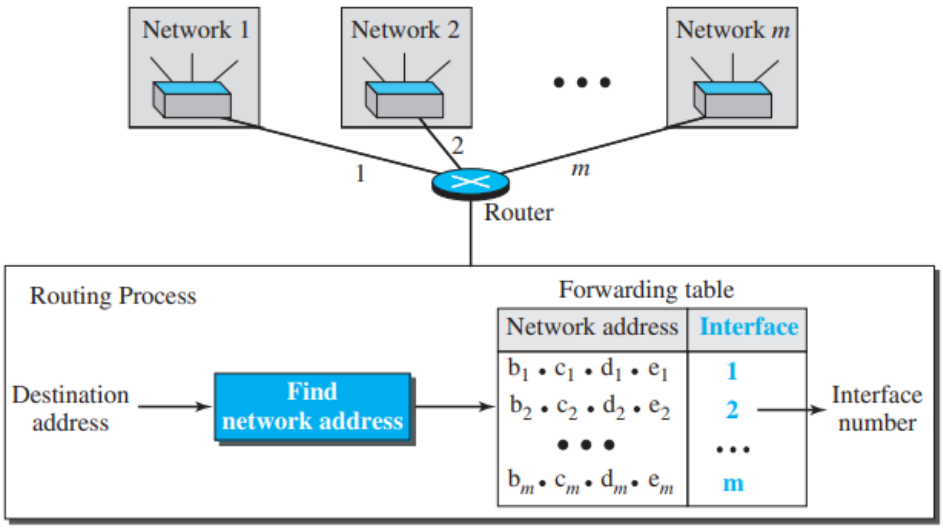
3. Address Mask

- 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 bit-wise operations NOT, AND, and OR.
- The number of addresses in the block N = NOT (mask) + 1.
- The first address in the block = (Any address in the block) AND (mask).
- The last address in the block = (Any address in the block) OR [(NOT (mask)].

- Example: 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

4. Network Address

- The first address, the network address, is particularly important because it is used in routing a packet to its destination network.
- Let us assume that an internet is made 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 from which interface the packet should be sent out.
- When the packet arrives at the network, it reaches its destination host using another strategy that we discuss later. Figure shows the idea.
- 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

5. Block Allocation

- The ultimate responsibility of block allocation is given to a global authority called the Internet Corporation for Assigned Names and Numbers (ICANN).
- 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.
- The number of requested addresses, N, needs to be a power of 2. The reason is that N = 2^{32-n} or $n = 32 - log_2 N$. If N is not a power of 2, we cannot have an integer value for n.
- The requested block needs to be allocated where there is an adequate number of contiguous addresses available in the address space. There is a restriction on choosing the first address in the block. The first address needs to be divisible by the number of addresses in the block. The reason is that the first address needs to be the prefix followed by (32 − n) number of 0s. The decimal value of the first address is then

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

6. 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).
- 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}. Then the following 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 3 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.

Solution:

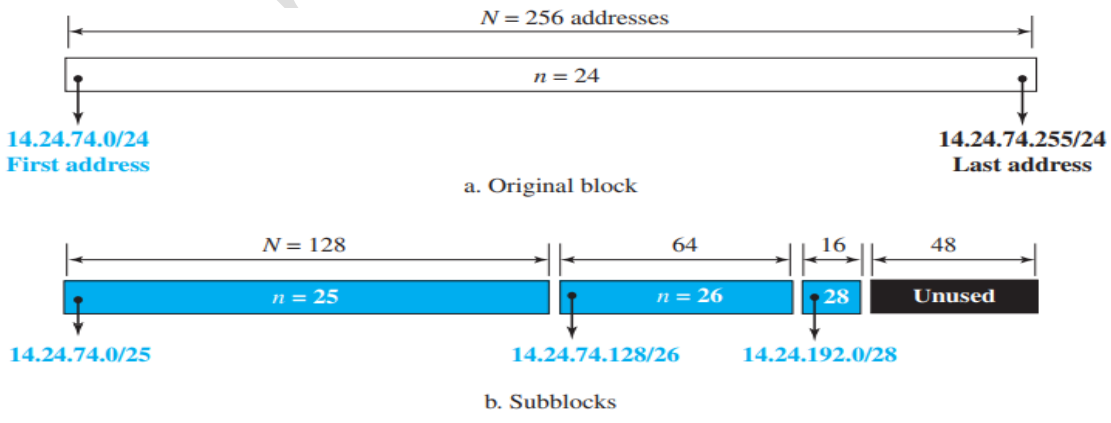
There are 2^{32-24} = 256 addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24. To satisfy the third requirement, we assign addresses to subblocks, starting with the largest and ending with the smallest one.

* 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 $n_1 = 32 - log_2 128$ = 25. The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.

* 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 $n_2 = 32 - log_2 64$ = 26. The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.

* 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 $n_3 = 32 - log_2 16$ = 28. The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

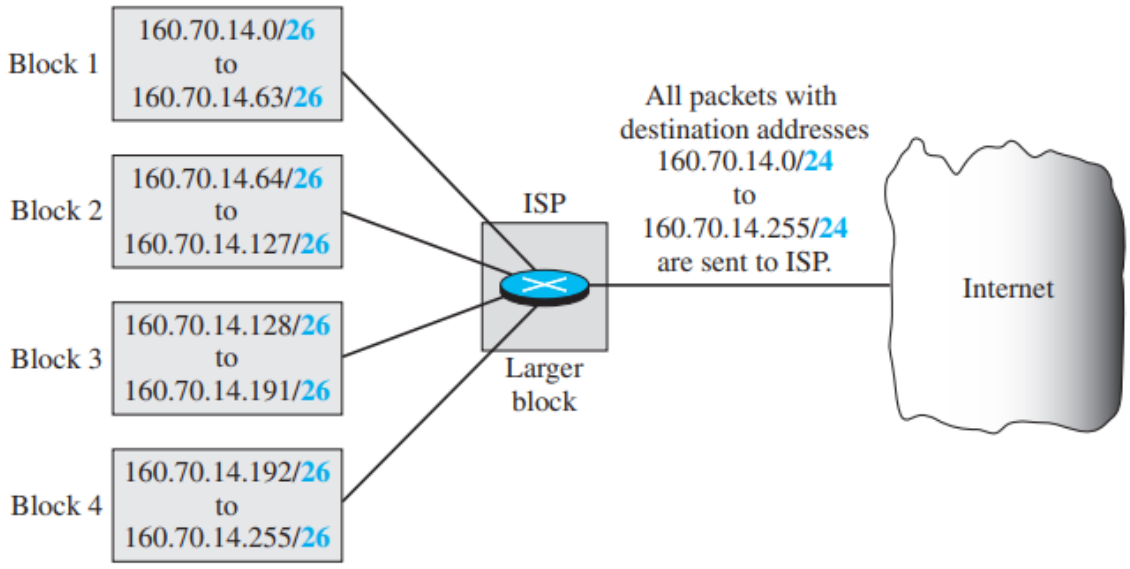
If we add all addresses in the previous subblocks, the result is 208 addresses, which means 48 addresses are left in reserve. The first address in this range is 14.24.74.208. The last address is 14.24.74.255. We do not know about the prefix length yet. Figure shows the configuration of blocks. We have shown the first address in each block.



7. 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.

- Example
  Figure shows how 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. This is like routing we can find in a postal network. All packages coming from outside a country are sent first to the capital and then distributed to the corresponding destination.



8. Special Addresses

- There are five special addresses that are used for special purposes: this-host address, limited-broadcast address, loopback address, private addresses, and multicast addresses.
- This-host Address: The only address in the block 0.0.0.0/32 is called the this-host address. It is used whenever a host needs to send an IP datagram, but it does not know its own address to use as the source address.
- Limited-broadcast Address: The only address in the block 255.255.255.255/32 is called the limited-broadcast address. It is used whenever a router or a host needs to send a datagram to all devices in a network. The routers in the network, however, block the packet having this address as the destination; the packet cannot travel outside the network.

- ❖ Loopback Address: The block 127.0.0.0/8 is called the loopback address. A packet with one of the addresses in this block as the destination address never leaves the host; it will remain in the host. Any address in the block is used to test a piece of software in the machine.
- ❖ Private Addresses: Four blocks are assigned as private addresses: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16.
- ❖ Multicast Addresses: The block 224.0.0.0/4 is reserved for multicast addresses.

Dynamic Host Configuration Protocol (DHCP):

We have seen that a large organization or an ISP can receive a block of addresses directly from ICANN and a small organization can receive a block of addresses from an ISP.

After a block of addresses are assigned to an organization, the network administration can manually assign addresses to the individual hosts or routers.

Address assignment in an organization can be done automatically using the Dynamic Host Configuration Protocol (DHCP).

DHCP is an application-layer program, using the client-server paradigm, that helps TCP/IP at the network layer. DHCP has found such widespread use in the Internet that it is often called a plug-and-play protocol.

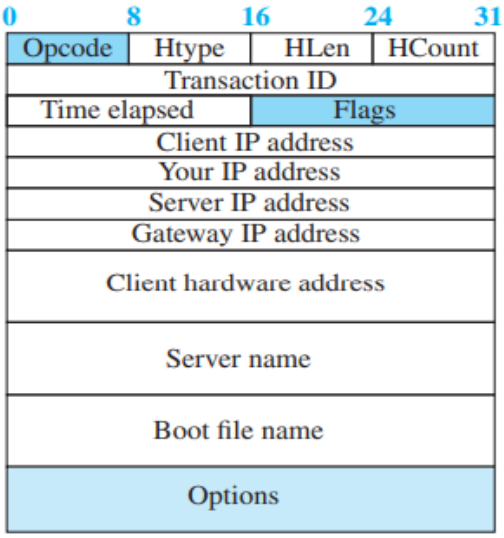
A network manager can configure DHCP to assign permanent IP addresses to the host and routers. DHCP can also be configured to provide temporary, on demand, IP addresses to hosts.

The second capability can provide a temporary IP address to a traveller to connect her laptop to the Internet while she is staying in the hotel. It also allows an ISP with 1000 granted addresses to provide services to 4000 households, assuming not more than one-fourth of customers use the Internet at the same time.

Four pieces of information are normally needed: the computer address, the prefix, the address of a router, and the IP address of a name server. DHCP can be used to provide these pieces of information to the host.

DHCP Message Format

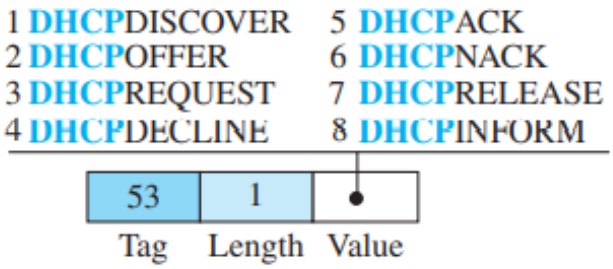
- ✓ DHCP is a client-server protocol in which the client sends a request message, and the server returns a response message.
- ✓ The general format of the DHCP message in figure. Most of the fields are explained in the figure, but we need to discuss the option field, which plays an important role in DHCP.
- ✓ The 64-byte option field has a dual purpose. It can carry either additional information or some specific vendor information.
- ✓ The server uses a number, called a magic cookie, in the format of an IP address with the value of 99.130.83.99. When the client finishes reading the message, it looks for this magic cookie.
- ✓ If present, the next 60 bytes are options. An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field.



Fields:

Opcode: Operation code, request (1) or reply (2)
Htype: Hardware type (Ethernet, ...)
HLen: Length of hardware address
HCount: Maximum number of hops the packet can travel
Transaction ID: An integer set by the client and repeated by the server
Time elapsed: The number of seconds since the client started to boot
Flags: First bit defines unicast (0) or multicast (1); other 15 bits not used
Client IP address: Set to 0 if the client does not know it
Your IP address: The client IP address sent by the server
Server IP address: A broadcast IP address if client does not know it
Gateway IP address: The address of default router
Server name: A 64-byte domain name of the server
Boot file name: A 128-byte file name holding extra information
Options: A 64-byte field with dual purpose described in text

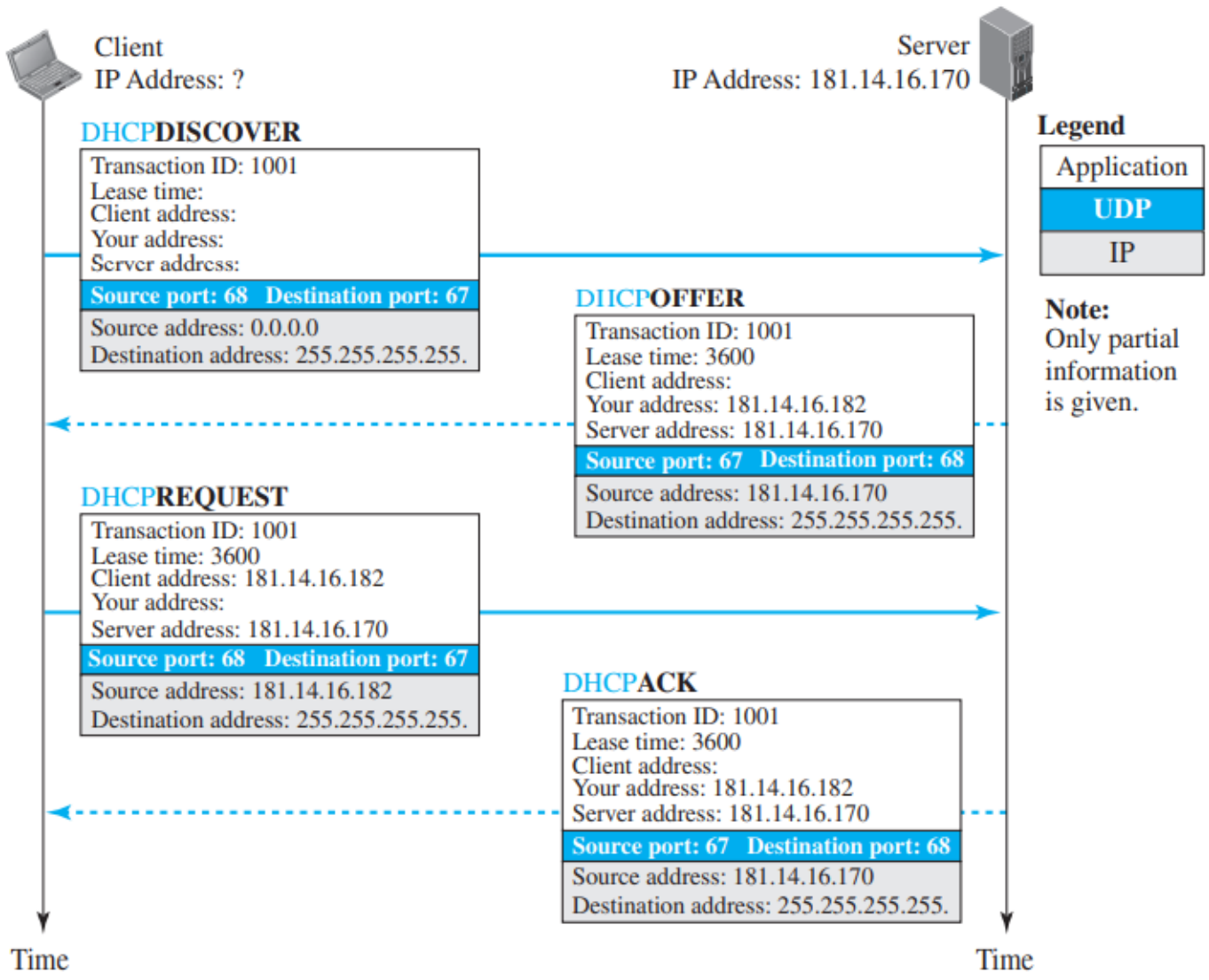
✓ There are several tag fields that are mostly used by vendors. If the tag field is 53, the value field defines one of the 8 message types shown in Figure .



Option format

DHCP Operation

Figure shows a simple scenario.



- ✓ The joining host creates a DHCPDISCOVER message in which only the transaction ID field is set to a random number. No other field can be set because the host has no knowledge with which to do so. This message is encapsulated in a UDP user datagram with the source port set to 68 and the destination port set to 67. The user datagram is encapsulated in an IP datagram with the source address set to 0.0.0.0 ("this host") and the destination address set to 255.255.255.255 (broadcast address). The reason is that the joining host knows neither its own address nor the server address.
- ✓ The DHCP server or servers (if more than one) responds with a DHCPOFFER message in which the your address field defines the offered IP address for the joining host and the server address field includes the IP address of the server. The message also includes the lease time for which the host can keep the IP address. This message is encapsulated in a user datagram with the same port numbers, but in the reverse order. The user datagram in turn is encapsulated in a datagram with the server address as the source IP address, but the destination address is a broadcast address, in which the server allows other DHCP servers to receive the offer and give a better offer if they can
- ✓ The joining host receives one or more offers and selects the best of them. The joining host then sends a DHCPREQUEST message to the server that has given the best offer. The fields with known value are set. The message is encapsulated in a user datagram with port numbers as the first message. The user datagram is encapsulated in an IP datagram with the source address set to the new client address, but the destination address still is set to the broadcast address to let the other servers know that their offer was not accepted.
- ✓ Finally, the selected server responds with a DHCPACK message to the client if the offered IP address is valid. If the server cannot keep its offer (for example, if the address is offered to another host in between), the server sends a DHCPNACK message, and the client needs to repeat the process. This message is also broadcast to let other servers know that the request is accepted or rejected.

Two Well-Known Ports:

- The DHCP uses two well-known ports (68 and 67) instead of one well-known and one ephemeral.
- The reason for choosing the well-known port 68 instead of an ephemeral port for the client is that the response from the server to the client is broadcast.
- Remember that an IP datagram with the limited broadcast message is delivered to every host on the network. Now assume that a DHCP client and a DAYTIME client, for example, are both waiting to receive a response from their corresponding server, and both have accidentally used the same temporary port number (56017, for example).
- Both hosts receive the response message from the DHCP server and deliver the message to their clients. The DHCP client processes the message; the DAYTIME client is totally confused with a strange message received.
- Using a well-known port number prevents this problem from happening. The response message from the DHCP server is not delivered to the DAYTIME client, which is running on the port number 56017, not 68. The temporary port numbers are selected from a different range than the well-known port numbers.

Using FTP:

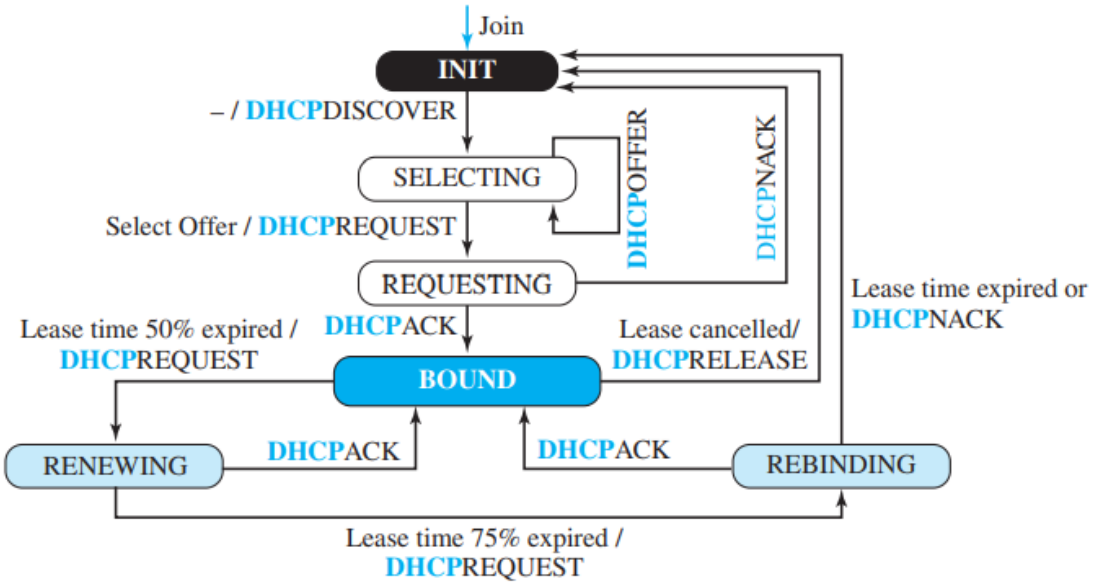
- The server does not send all the information that a client may need for joining the network.
- In the DHCPACK message, the server defines the pathname of a file in which the client can find complete information such as the address of the DNS server.
- The client can then use a file transfer protocol to obtain the rest of the needed information.

Error Control:

- DHCP uses the service of UDP, which is not reliable. To provide error control, DHCP uses two strategies. First, DHCP requires that UDP use the checksum.
- Second, the DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request.
- To prevent a traffic jam when several hosts need to retransmit a request (for example, after a power failure), DHCP forces the client to use a random number to set its timers.

Transition States:

To provide dynamic address allocation, the DHCP client acts as a state machine that performs transitions from one state to another depending on the messages it receives or sends.



FSM for the DHCP client

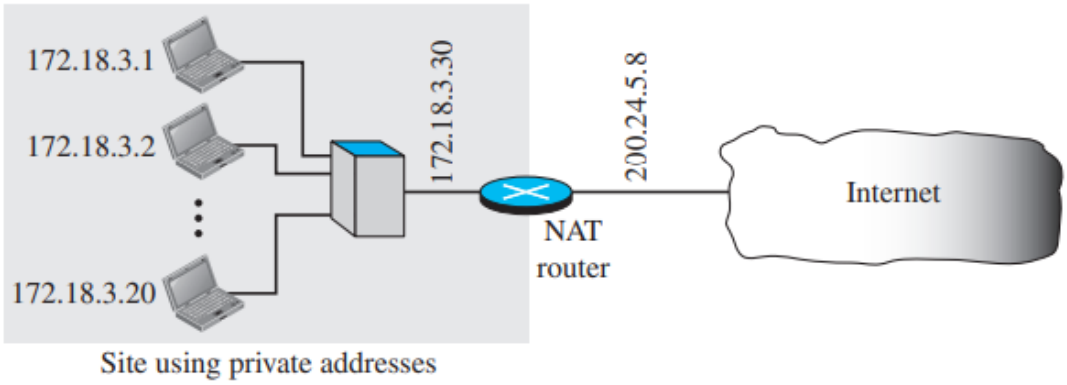
- When the DHCP client first starts, it is in the INIT state (initializing state). The client broadcasts a discover message.
- When it receives an offer, the client goes to the SELECTING state. While it is there, it may receive more offers. After it selects an offer, it sends a request message and goes to the REQUESTING state.
- If an ACK arrives while the client is in this state, it goes to the BOUND state and uses the IP address. When the lease is 50 percent expired, the client tries to renew it by moving to the RENEWING state.
- If the server renews the lease, the client moves to the BOUND state again. If the lease is not renewed and the lease time is 75 percent expired, the client moves to the REBINDING state.
- If the server agrees with the lease (ACK message arrives), the client moves to the BOUND state and continues using the IP address; otherwise, the client moves to the INIT state and requests another IP address.
- Note that the client can use the IP address only when it is in the BOUND, RENEWING, or REBINDING state. The above procedure requires that the client uses three timers:
    - renewal timer (set to 50 percent of the lease time)
    - rebinding timer (set to 75 percent of the lease time)
    - expiration timer (set to the lease time).

Network Address Translation (NAT):

The distribution of addresses through ISPs has created a new problem.

- ✓ Assume that an ISP has granted a small range of addresses to a small business or a household. If the business grows or the household needs a larger range, the ISP may not be able to grant the demand because the addresses before and after the range may have already been allocated to other networks.
- ✓ In most situations only a portion of computers in a small network need access to the Internet simultaneously. This means that the number of allocated addresses does not have to match the number of computers in the network.
- ✓ For example, assume that in a small business with 20 computers the maximum number of computers that access the Internet simultaneously is only 4.
- ✓ Most of the computers are either doing some tasks that does not need Internet access or communicating with each other. This small business can use the TCP/IP protocol for both internal and universal communication.
- ✓ The business can use 20 (or 25) addresses from the private block addresses for internal communication; five addresses for universal communication can be assigned by the ISP.
- ✓ A technology that can provide the mapping between the private and universal addresses, and at the same time support virtual private networks is Network Address Translation (NAT).
- ✓ The technology allows a site to use a set of private addresses for internal communication and a set of global Internet addresses (at least one) for communication with the rest of the world. The site must have only one connection to the global Internet through a NAT-capable router that runs NAT software.

Figure shows a simple implementation of NAT.



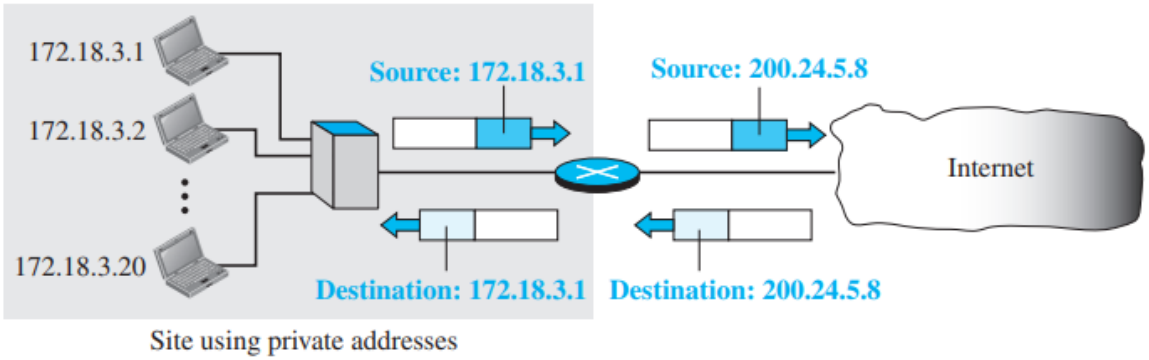
NAT

As the figure shows, the private network uses private addresses. The router that connects the network to the global address uses one private address and one global address. The private network is invisible to the rest of the Internet; the rest of the Internet sees only the NAT router with the address 200.24.5.8.

Address Translation:

All the outgoing packets go through the NAT router, which replaces the source address in the packet with the global NAT address.

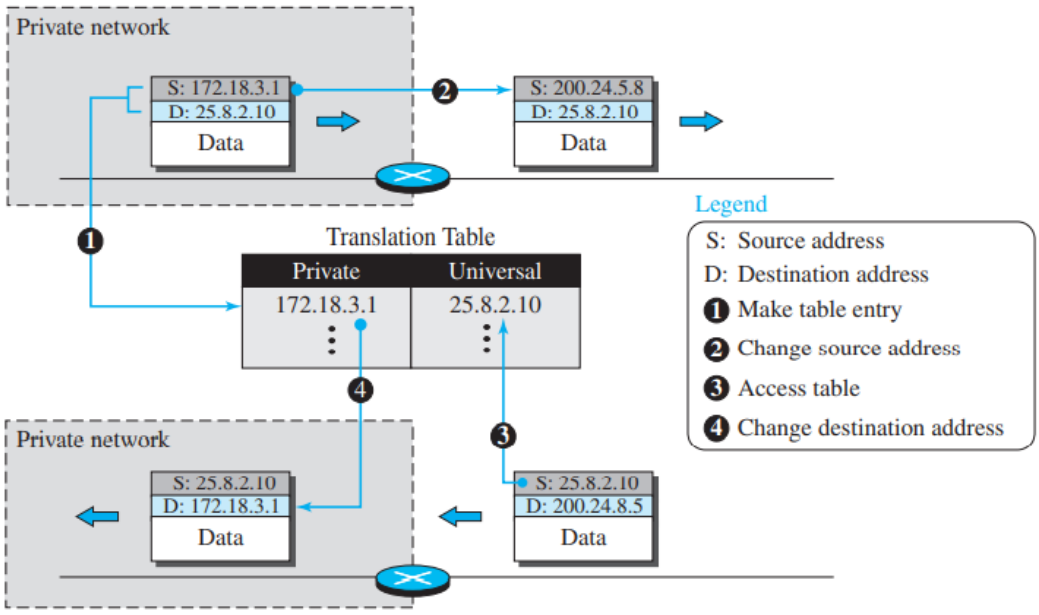
All incoming packets also pass through the NAT router, which replaces the destination address in the packet (the NAT router global address) with the appropriate private address. Figure shows an example of address translation.



Address translation

Translation Table:

- ✓ Using One IP Address
  - ❖ In its simplest form, a translation table has only two columns: the private address and the external address (destination address of the packet).
  - ❖ When the router translates the source address of the outgoing packet, it also makes note of the destination address— where the packet is going.
  - ❖ When the response comes back from the destination, the router uses the source address of the packet (as the external address) to find the private address of the packet. Figure shows the idea.



Translation

- ❖ In this strategy, communication must always be initiated by the private network. The NAT mechanism described requires that the private network start the communication.
- ❖ NAT is used mostly by ISPs that assign a single address to a customer. The customer may be a member of a private network that has many private addresses.
- ❖ In this case, communication with the Internet is always initiated from the customer site, using a client program such as HTTP, TELNET, or FTP to access the corresponding server program.
- ❖ For example, when e-mail that originates from outside the network site is received by the ISP e-mail server, it is stored in the mailbox of the customer until retrieved with a protocol such as POP.

✓ Using a Pool of IP Addresses

- ❖ The use of only one global address by the NAT router allows only one private-network host to access a given external host. To remove this restriction, the NAT router can use a pool of global addresses.
- ❖ For example, instead of using only one global address (200.24.5.8), the NAT router can use four addresses (200.24.5.8, 200.24.5.9, 200.24.5.10, and 200.24.5.11).
- ❖ In this case, four private-network hosts can communicate with the same external host at the same time because each pair of addresses defines a separate connection.
- ❖ There are still some drawbacks. No more than four connections can be made to the same destination. No private-network host can access two external server programs (e.g., HTTP and TELNET) at the same time. And two private-network hosts cannot access the same external server program (e.g., HTTP or TELNET) at the same time.

✓ Using Both IP Addresses and Port Addresses

- ❖ To allow a many-to-many relationship between private-network hosts and external server programs, we need more information in the translation table.
- ❖ For example, suppose two hosts inside a private network with addresses 172.18.3.1 and 172.18.3.2 need to access the HTTP server on external host 25.8.3.2.
- ❖ If the translation table has five columns, instead of two, that include the source and destination port addresses and the transport-layer protocol, the ambiguity is eliminated. Table shows an example of such a table.

Private address	*Private port*	*External address*	*External port*	*Transport protocol*
172.18.3.1	1400	25.8.3.2	80	TCP
172.18.3.2	1401	25.8.3.2	80	TCP
⋮	⋮	⋮	⋮	⋮

Five-column translation table

- ❖ Note that when the response from HTTP comes back, the combination of source address (25.8.3.2) and destination port address (1401) defines the private network host to which the response should be directed. For this translation to work, the ephemeral port addresses (1400 and 1401) must be unique.

Forwarding of IP Packets:

Since the Internet today is made of a combination of links (networks), forwarding means to deliver the packet to the next hop (which can be the destination or the intermediate connecting device). Although the IP protocol was originally designed as a connectionless protocol, today the tendency is to change it to a connection-oriented protocol.

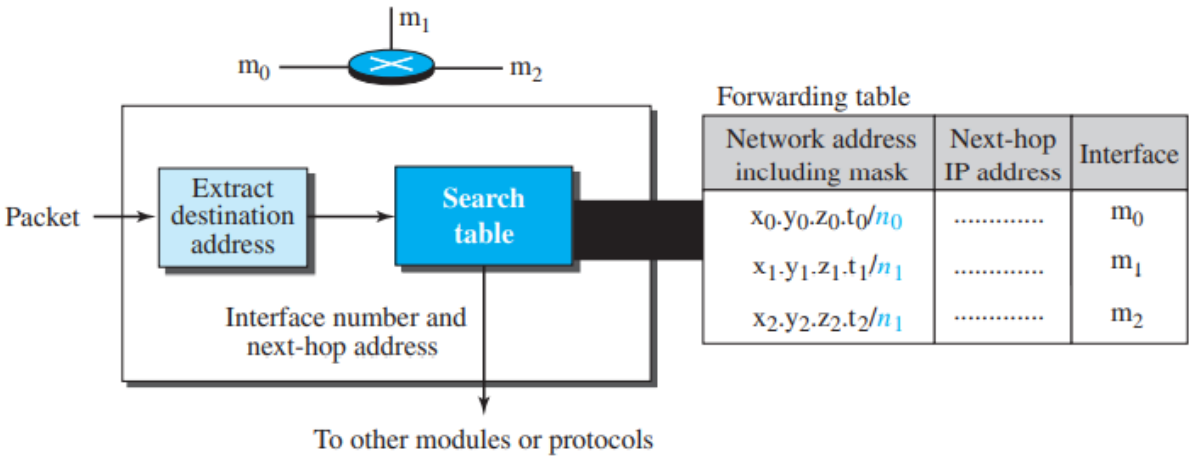
When IP is used as a connectionless protocol, forwarding is based on the destination address of the IP datagram; when the IP is used as a connection-oriented protocol, forwarding is based on the label attached to an IP datagram.

Forwarding Based on Destination Address:

This is a traditional approach, which is prevalent today. In this case, forwarding requires a host or a router to have a forwarding table. When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the next hop to deliver the packet to.

- ✓ In classless addressing, the whole address space is one entity; there are no classes. This means that forwarding requires one row of information for each block involved.
- ✓ The table needs to be searched based on the network address (first address in the block). Unfortunately, the destination address in the packet gives no clue about the network address. To solve the problem, we need to include the mask (/n) in the table.
- ✓ A classless forwarding table needs to include four pieces of information: the mask, the network address, the interface number, and the IP address of the next.
- ✓ For example, if n is 26 and the network address is 180.70.65.192, then one can combine the two as one piece of information: 180.70.65.192/26.

Figure shows a simple forwarding module and forwarding table for a router with only three interfaces.



Simplified forwarding module in classless address

- ✓ The job of the forwarding module is to search the table, row by row. In each row, the n leftmost bits of the destination address (prefix) are kept, and the rest of the bits (suffix) are set to 0s.
- ✓ If the resulting address (which we call the network address), matches with the address in the first column, the information in the next two columns is extracted; otherwise, the search continues.
- ✓ Normally, the last row has a default value in the first column which indicates all destination addresses that did not match the previous rows.

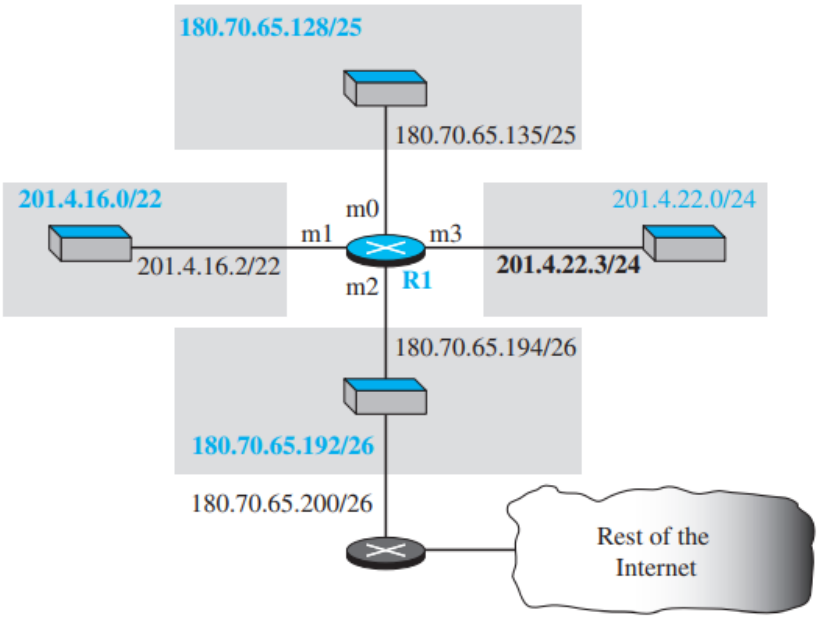
- ✓ Sometimes, the literature explicitly shows the value of the n leftmost bits that should be matched with the n leftmost bits of the destination address.
- ✓ For example, instead of giving the address-mask combination of 180.70.65.192/26, we can give the value of the 26 leftmost bits as shown below.

10110100 01000110 01000001 11

- ✓ Note that we still need to use an algorithm to find the prefix and compare it with the bit pattern. The algorithm is still needed, but the presentation is different. We use this format in our forwarding tables in the exercises when we use smaller address spaces just for practice.

Example

Make a forwarding table for router R1 using the configuration in Figure



Solution

Table shows the corresponding table

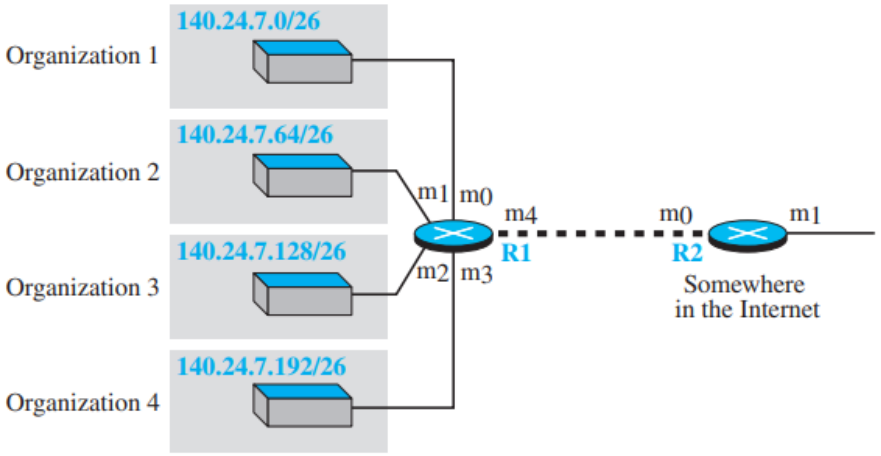
Network address/mask	*Next hop*	*Interface*
180.70.65.192/**26**	—	m2
180.70.65.128/**25**	—	m0
201.4.22.0/**24**	—	m3
201.4.16.0/**22**	—	m1
Default	180.70.65.200	m2

Forwarding table for router R1 in Figure

Address Aggregation:

- ✓ When we use classful addressing, there is only one entry in the forwarding table for each site outside the organization. The entry defines the site even if that site is subnetted.

- ✓ When a packet arrives at the router, the router checks the corresponding entry and forwards the packet accordingly.
- ✓ When we use classless addressing, it is likely that the number of forwarding table entries will increase. This is because the intent of classless addressing is to divide up the whole address space into manageable blocks.
- ✓ The increased size of the table results in an increase in the amount of time needed to search the table. To alleviate the problem, the idea of address aggregation was designed.
- ✓ In figure we have two routers. R1 is connected to networks of four organizations that each use 64 addresses. R2 is somewhere far from R1.
- ✓ R1 has a longer forwarding table because each packet must be correctly routed to the appropriate organization. R2, on the other hand, can have a ridiculously small forwarding table.
- ✓ For R2, any packet with destination 140.24.7.0 to 140.24.7.255 is sent out from interface m0 regardless of the organization number.
- ✓ This is called address aggregation because the blocks of addresses for four organizations are aggregated into one larger block.
- ✓ R2 would have a longer forwarding table if each organization had addresses that could not be aggregated into one block.



Forwarding table for R1

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

Forwarding table for R2

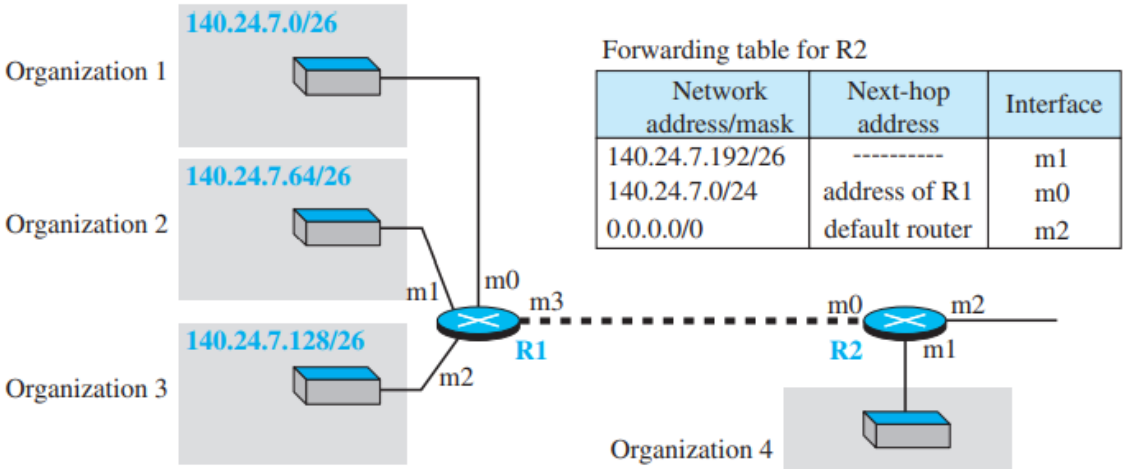
Network address/mask	Next-hop address	Interface
140.24.7.0/24	----------	m0
0.0.0.0/0	default router	m1

Address aggregation

1. Longest Mask Matching:

- ❖ If organization 4 cannot be connected to router R1 for some reason, can we still use the idea of address aggregation and still assign block 140.24.7.192/26 to organization 4? The answer is yes, because routing in classless addressing uses another principle, longest mask matching.
- ❖ This principle states that the forwarding table is sorted from the longest mask to the shortest mask. If there are three masks, /27, /26, and /24, the mask /27 must be the first entry and /24 must be the last.

- Let us see if this principle solves the situation in which organization 4 is separated from the other three organizations. Figure shows the situation.
- Suppose a packet arrives at router R2 for organization 4 with destination address 140.24.7.200. The first mask at router R2 is applied, which gives the network address 140.24.7.192.
- The packet is routed correctly from interface m1 and reaches organization 4. If, however, the forwarding table was not stored with the longest prefix first, applying the /24 mask would result in the incorrect routing of the packet to router R1.



Forwarding table for R2

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

Forwarding table for R1

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

Longest mask matching

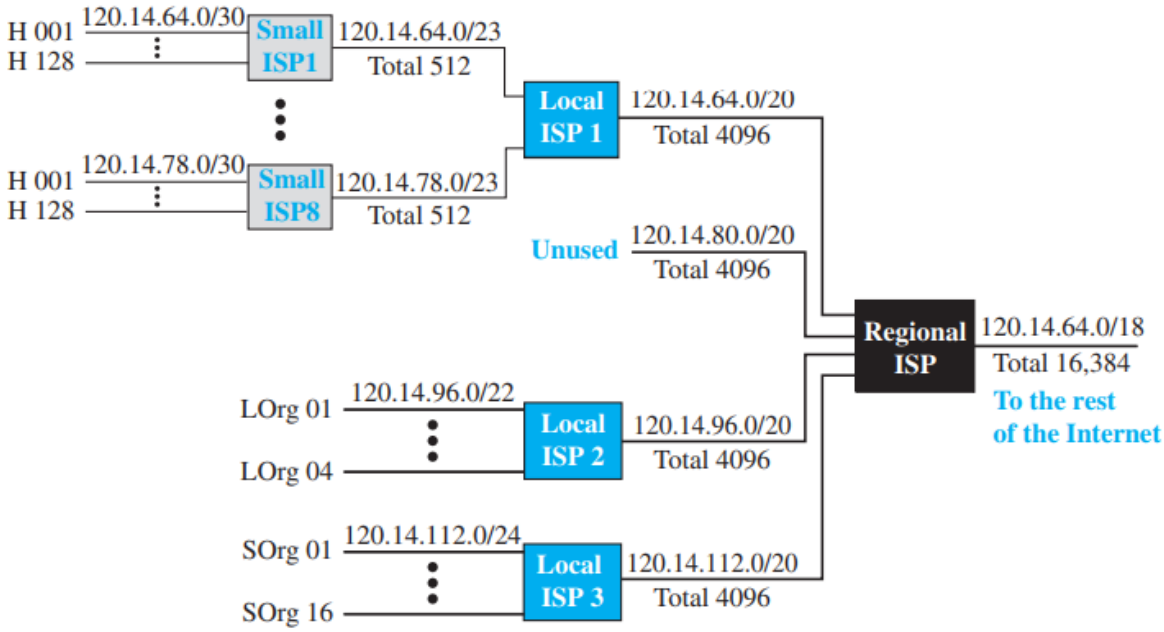
2. Hierarchical Routing

- To solve the problem of gigantic forwarding tables, we can create a sense of hierarchy in the forwarding tables.
- The Internet is divided into backbone and national ISPs. National ISPs are divided into regional ISPs, and regional ISPs are divided into local ISPs.
- If the forwarding table has a sense of hierarchy like the Internet architecture, the forwarding table can decrease in size. A local ISP can be assigned a single, but large, block of addresses with a certain prefix length. The local ISP can divide this block into smaller blocks of different sizes, and assign these to individual users and organizations, both large and small.
- If the block assigned to the local ISP starts with a.b.c.d/n, the ISP can create blocks starting with e.f.g.h/m, where m may vary for each customer and is greater than n.
- All customers of the local ISP are defined as a.b.c.d/n to the rest of the Internet. Every packet destined for one of the addresses in this large block is routed to the local ISP. There is only one entry in every router in the world for all these customers. They all belong to the same group.
- Inside the local ISP, the router must recognize the subblocks and route the packet to the destined customer. If one of the customers is a large organization, it also can create another level of hierarchy

by subnetting and dividing its subblock into smaller subblocks (or sub-subblocks). In classless routing, the levels of hierarchy are unlimited if we follow the rules of classless addressing.

Example

As an example of hierarchical routing, let us consider figure. A regional ISP is granted 16,384 addresses starting from 120.14.64.0. The regional ISP has decided to divide this block into four subblocks, each with 4096 addresses. Three of these subblocks are assigned to three local ISPs; the second subblock is reserved for future use. Note that the mask for each block is /20 because the original block with mask /18 is divided into 4 blocks.



Hierarchical routing with ISPs

- The first local ISP has divided its assigned subblock into 8 smaller blocks and assigned each to a small ISP.
- Each small ISP provides services to 128 households (H001 to H128), each using four addresses. Note that the mask for each small ISP is now /23 because the block is further divided into 8 blocks.
- Each household has a mask of /30 because a household has only 4 addresses (2^{32-30}= 4). The second local ISP has divided its block into 4 blocks and has assigned the addresses to 4 large organizations (LOrg01 to LOrg04).
- Note that each large organization has 1024 addresses and the mask is /22. The third local ISP has divided its block into 16 blocks and assigned each block to a small organization (SOrg01 to SOrg16).
- Each small organization has 256 addresses and the mask is /24. There is a sense of hierarchy in this configuration.
- All routers in the Internet send a packet with destination address 120.14.64.0 to 120.14.127.255 to the regional ISP. The regional ISP sends every packet with destination address 120.14.64.0 to 120.14.79.255 to Local ISP1. Local ISP1 sends every packet with destination address 120.14.64.0 to 120.14.64.3 to H001.

3. Geographical Routing

- ❖ To decrease the size of the forwarding table even further, we need to extend hierarchical routing to include geographical routing.
- ❖ We must divide the entire address space into a few large blocks. We assign a block to America, a block to Europe, a block to Asia, a block to Africa, and so on.
- ❖ The routers of ISPs outside of Europe will have only one entry for packets to Europe in their forwarding tables. The routers of ISPs outside of America will have only one entry for packets to America in their forwarding tables, and so on.

Forwarding Table Search Algorithms

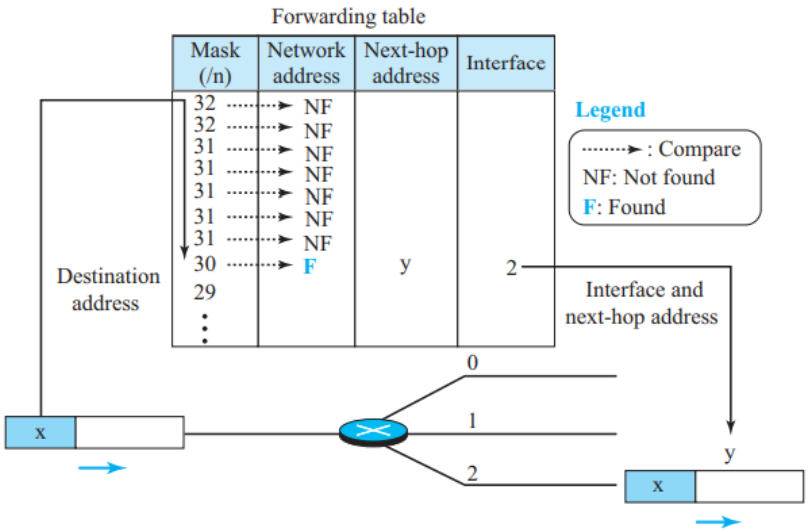
- ✓ In classless addressing, there is no network information in the destination address. The simplest, search method is called the longest prefix match.
- ✓ The forwarding table can be divided into buckets, one for each prefix. The router first tries the longest prefix. If the destination address is found in this bucket, the search is complete.
- ✓ If the address is not found, the next prefix is searched, and so on. It is obvious that this type of search takes a long time. One solution is to change the data structure used for searching.
- ✓ Instead of a list, other data structures (such as a tree or a binary tree) can be used. One candidate is a trie (a special kind of tree).

Forwarding Based on Label

- ✓ In the 1980s, an effort started to somehow change IP to behave like a connection-oriented protocol in which the routing is replaced by switching. In a connectionless network (datagram approach), a router forwards a packet based on the destination address in the header of the packet.
- ✓ In a connection-oriented network (virtual-circuit approach), a switch forwards a packet based on the label attached to the packet. Routing is normally based on searching the contents of a table; switching can be done by accessing a table using an index. In other words, routing involves searching; switching involves accessing.

Example:

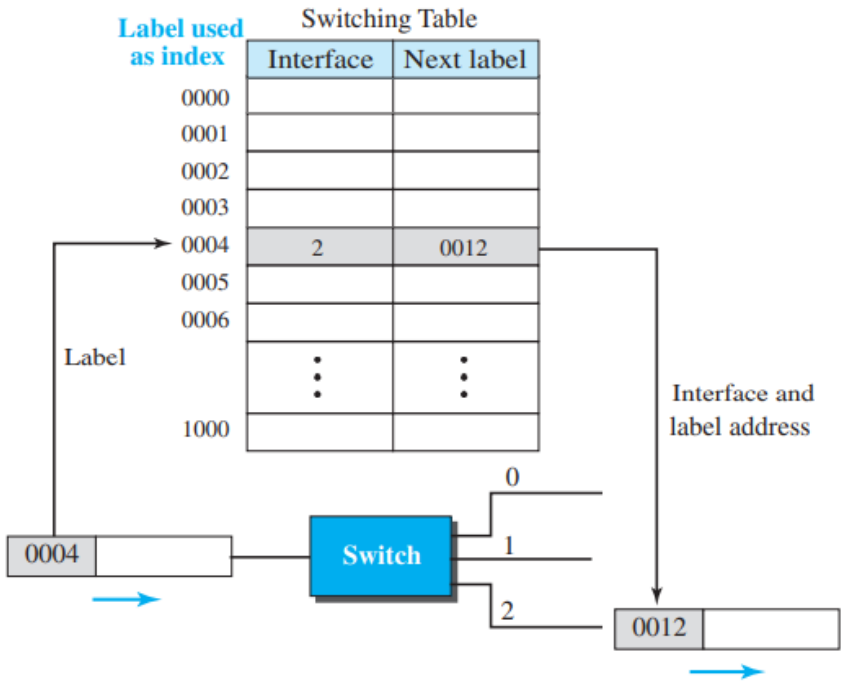
Figure shows a simple example of searching in a forwarding table using the longest mask algorithm.



When the forwarding algorithm gets the destination address of the packet, it needs to delve into the mask column. For each entry, it needs to apply the mask to find the destination network address. It then needs to check the network addresses in the table until it finds the match. The router then extracts the next-hop address and the interface number to be delivered to the data-link layer.

Example:

Figure shows a simple example of using a label to access a switching table. Since the labels are used as the index to the table, finding the information in the table is immediate.

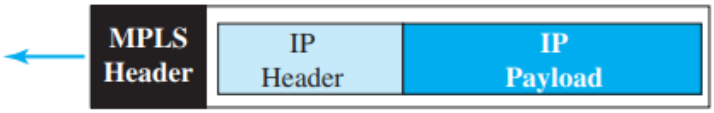


Multi-Protocol Label Switching (MPLS):

- ✓ During the 1980s, several vendors created routers that implement switching technology. Later IETF approved a standard that is called Multi-Protocol Label Switching.
- ✓ In this standard, some conventional routers in the Internet can be replaced by MPLS routers, which can behave like a router and a switch.
- ✓ When behaving like a router, MPLS can forward the packet based on the destination address; when behaving like a switch, it can forward a packet based on the label.

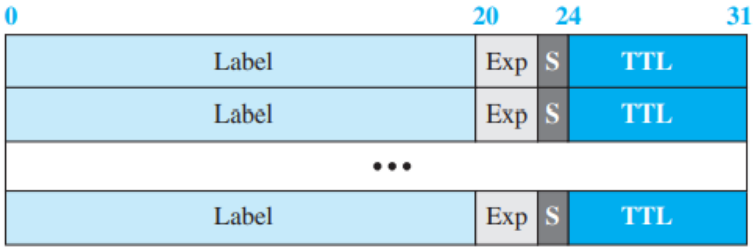
A New Header:

- ✓ To simulate connection-oriented switching using a protocol like IP, the first thing that is needed is to add a field to the packet that carries the label.
- ✓ The IPv4 packet format does not allow this extension (although this field is provided in the IPv6 packet format).
- ✓ The solution is to encapsulate the IPv4 packet in an MPLS packet (as though MPLS were a layer between the data-link layer and the network layer).
- ✓ The whole IP packet is encapsulated as the payload in an MPLS packet and an MPLS header is added. Figure shows the encapsulation.



MPLS header added to an IP packet

- The MPLS header is a stack of subheaders that is used for multilevel hierarchical switching. Figure shows the format of an MPLS header in which each subheader is 32 bits (4 bytes) long.



MPLS header made of a stack of labels

The following is a brief description of each field:

- Label: This 20-bit field defines the label that is used to index the forwarding table in the router.
- Exp: This 3-bit field is reserved for experimental purposes.
- S: The one-bit stack field defines the situation of the subheader in the stack. When the bit is 1, it means that the header is the last one in the stack.
- TTL: This 8-bit field is like the TTL field in the IP datagram. Each visited router decrements the value of this field. When it reaches zero, the packet is discarded to prevent looping

Hierarchical Switching:

- ✓ A stack of labels in MPLS allows hierarchical switching. This is like conventional hierarchical routing.
- ✓ For example, a packet with two labels can use the top label to forward the packet through switches outside an organization; the bottom label can be used to route the packet inside the organization to reach the destination subnet.

NETWORK LAYER PROTOCOLS:

INTERNET PROTOCOL (IP)

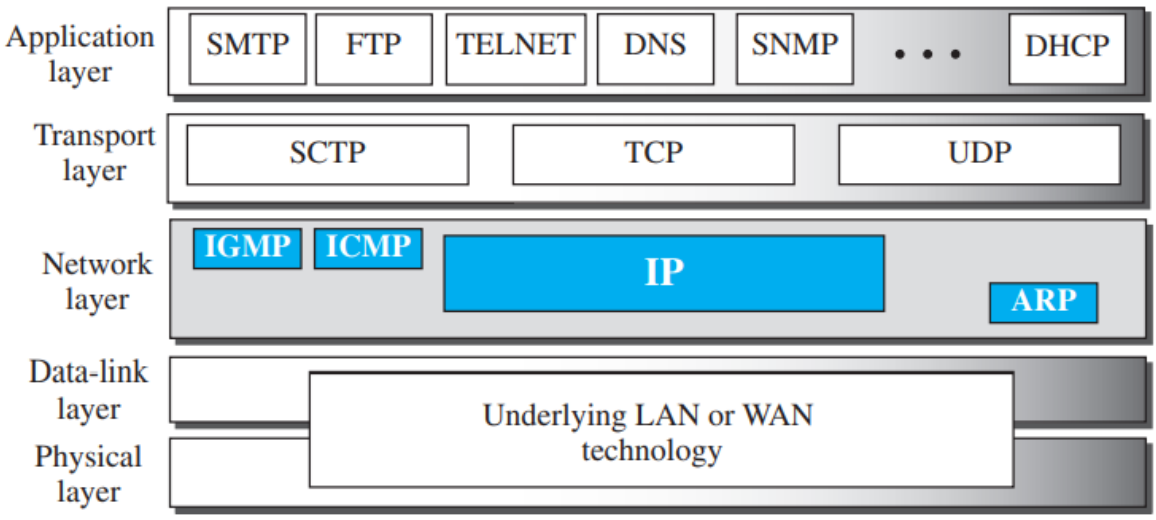
The network layer in version 4 is one main protocol and three auxiliary ones. The main protocol, Internet Protocol version 4 (IPv4), is responsible for packetizing, forwarding, and delivery of a packet at the network layer.

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 shows the positions of these four protocols in the TCP/IP protocol suite.



IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term best-effort means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network. If reliability is important, IPv4 must be paired with a reliable transport-layer protocol such as TCP.

Ex: Best-effort delivery service is the post office

The post office does its best to deliver the regular mail but does not always succeed. If an unregistered letter is lost or damaged, it is up to the sender or would-be recipient to discover this. The post office itself does not keep track of every letter and cannot notify a sender of loss or damage of one.

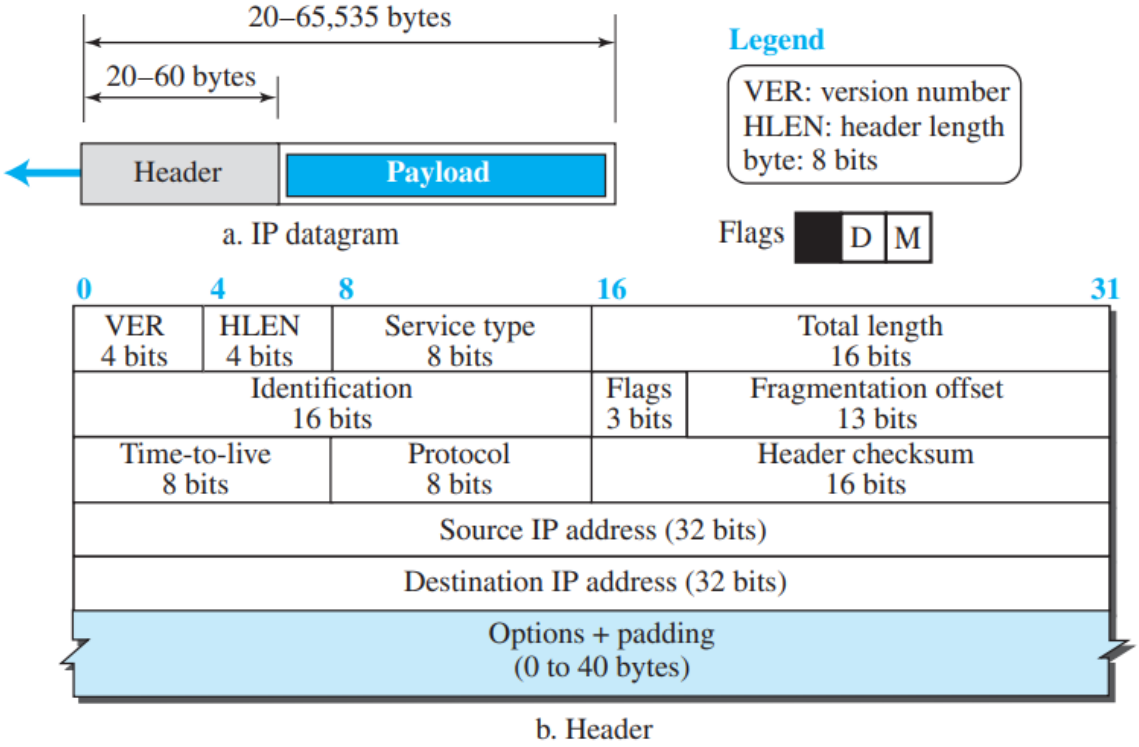
IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route to the destination.

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.

Datagram Format:

Packets used by the IP are called datagrams. Figure shows the IPv4 datagram format.

A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.



i. *Version Number*: The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.

ii. *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. When a device receives a datagram, it needs to know when the header stops and the data, which is encapsulated in the packet, starts. However, to make the value of the header length (number of bytes) fit in a 4-bit header length, the total length of the header is calculated as 4-byte words. The total length is divided by 4 and the value is inserted in the field. The receiver needs to multiply the value of this field by 4 to find the total length.

iii. *Service Type*: In the original design of the IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled. In the late 1990s, IETF redefined the field to provide differentiated services (DiffServ). The use of 4-byte words for the length header is also logical because the IP header always needs to be aligned in 4-byte boundaries.

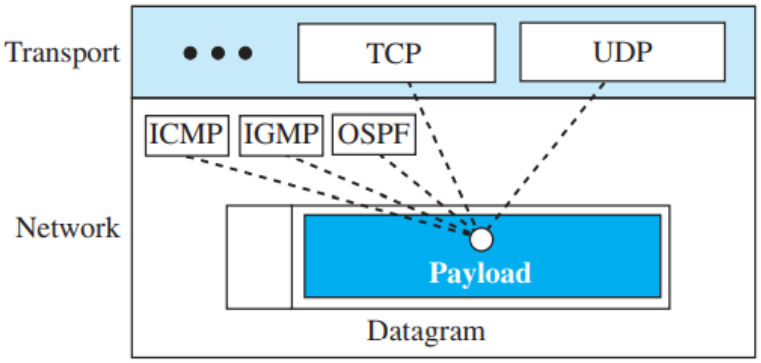
iv. *Total Length*: This 16-bit field defines the total length (header plus data) of the IP datagram in bytes. A 16-bit number can define a total length of up to 65,535 (when all bits are 1s). However, the size of the datagram is normally much less than this. This field helps the receiving device to 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.

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

v. *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.

vi. *Time-to-live:* Due to some malfunctioning of routing protocols a datagram may be circulating in the Internet, visiting some networks over and over without reaching the destination. This may create extra traffic in the Internet. The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.

vii. *Protocol:* In TCP/IP, the data section of a packet, called the payload, carries the whole packet from another protocol. A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP. A datagram can also carry a packet from other protocols that directly use the service of the IP, such as some routing protocols or some auxiliary protocols. The Internet authority has given any protocol that uses the service of IP a unique 8-bit number which is inserted in the protocol field. When the payload is encapsulated in a datagram at the source IP, the corresponding protocol number is inserted in this field; when the datagram arrives at the destination, the value of this field helps to define to which protocol the payload should be delivered. This field provides multiplexing at the source and demultiplexing at the destination, as shown in Figure.



Some protocol values

ICMP	01
IGMP	02
TCP	06
UDP	17
OSPF	89

viii. *Header checksum:* IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission. IP puts the burden of error checking of the payload on the protocol that owns the payload, such as UDP or TCP. The datagram header is added by IP, and its error-checking is the responsibility of IP. Errors in the IP header can be a disaster. For example, if the destination IP address is corrupted, the packet can be delivered to the wrong host. If the protocol field is corrupted, the payload may be delivered to the wrong protocol. If the fields related to the fragmentation are corrupted, the datagram cannot be reassembled correctly at the destination, and so on. For these reasons, IP adds a header checksum field to check the header, but not the payload. Since 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.

ix. *Source and Destination Addresses:* These 32-bit source and destination address fields define the IP address of the source and destination, respectively. The source host should know its IP address. The destination IP address is either known by the protocol that uses the service of IP or is provided by the

DNS. The value of these fields must remain unchanged during the time the IP datagram travels from the source host to the destination host.

x. *Options:* A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging. Although options are not a required part of the IP header, option processing is required of the IP software. This means that all implementations must be able to handle options if they are present in the header. The existence of options in a header creates some burden on the datagram handling; some options can be changed by routers, which forces each router to recalculate the header checksum.

xi. *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. Comparing a datagram to a postal package, payload is the content of the package; the header is only the information written on the package.

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 travelled.
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.

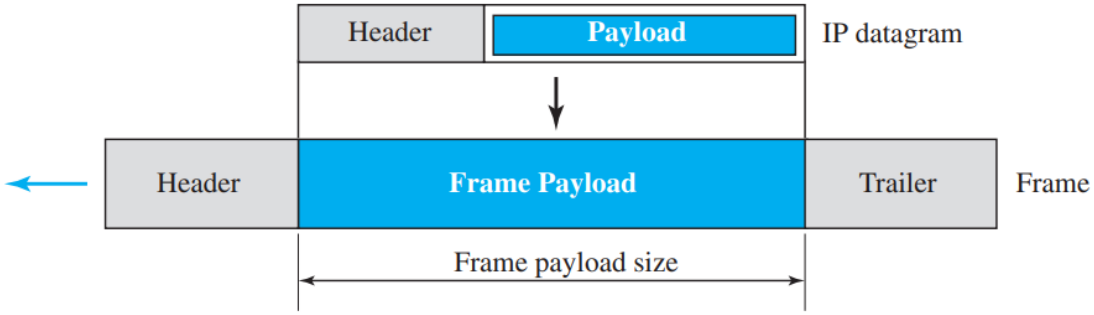
Ex: If a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU): Each link-layer protocol has its own frame format. One of the features of each format is the maximum size of the payload that can be encapsulated.

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.

The value of the MTU differs from one physical network protocol to another.

Ex: The value for a LAN is normally 1500 bytes, but for a WAN it can be larger or smaller.



MTU: Maximum size of frame payload

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.

For other physical networks, 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. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU.

Datagram may be fragmented several times before it reaches the destination. A datagram can be fragmented by the source host or any router in the path. The reassembly of the datagram 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 destination.

Fields Related to Fragmentation: Three fields in an IP datagram are related to fragmentation: identification, flags, and fragmentation offset.

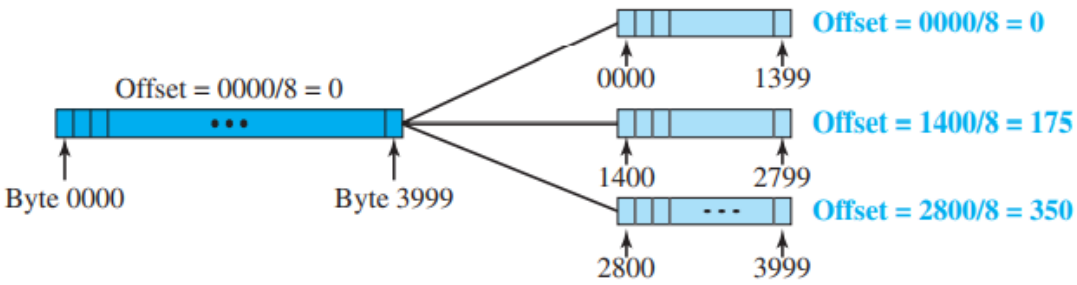
The 16-bit ***identification*** field identifies a datagram originating from the source host. The combination of the identification and source IP address must uniquely define a datagram as it leaves the source host.
To guarantee uniqueness, the IP protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IP protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by one.
If the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied into all fragments. All fragments have the same identification number, which is also the same as the original datagram.
The identification number helps the destination in reassembling the datagram. It knows that all fragments having the same identification value should be assembled into one 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.
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.

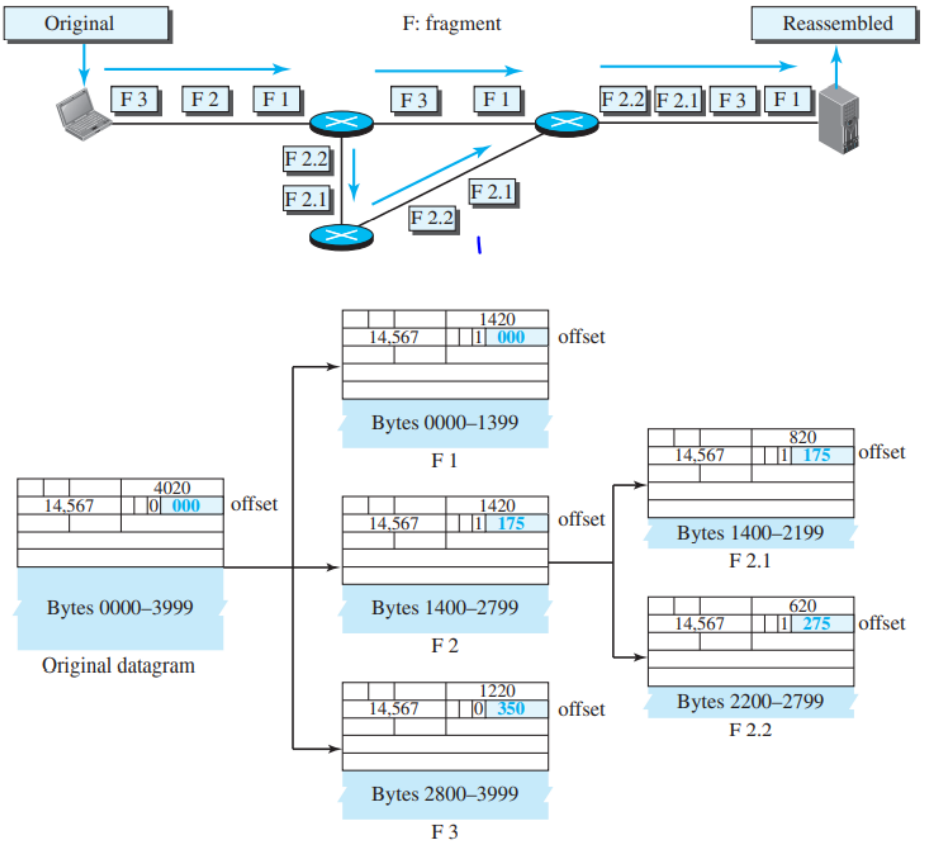
The 13-bit fragmentation offset field shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes.

Figure shows a datagram with a data size of 4000 bytes fragmented into three fragments.



The bytes in the original datagram are numbered 0 to 3999.
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.

The figure shows an expanded view of the fragments in the previous figure.



The original packet starts at the client; the fragments are reassembled at the server. The value of the identification field is the same in all fragments, as is the value of the flags field with the more bit set for all fragments except the last. Also, the value of the offset field for each fragment is shown. Note that although the fragments arrived out of order at the destination, they can be correctly reassembled.
The figure also shows what happens if a fragment itself is fragmented. In this case the value of the offset field is always relative to the original datagram.

It is obvious that even if each fragment follows a different path and arrives out of order, the destination host can reassemble the original datagram from the fragments received (if none of them is lost) 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 a more bit value of 0.

Options:

- ✓ The header of the IPv4 datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long and was discussed in the previous section.
- ✓ The variable part comprises the options that can be a maximum of 40 bytes (in multiples of 4-bytes) to preserve the boundary of the header.
- ✓ Options, as the name implies, are not required for a datagram. They can be used for network testing and debugging. Although options are not a required part of the IPv4 header, option processing is required of the IPv4 software.
- ✓ This means that all implementations must be able to handle options if they are present in the header. Options are divided into two broad categories: single-byte options and multiple-byte options.

Single-Byte Options:
There are two single-byte options.

- ✓ No Operation: A no-operation option is a 1-byte option used as a filler between options.
- ✓ End of Option: An end-of-option option is a 1-byte option used for padding at the end of the option field. It, however, can only be used as the last option.

Multiple-Byte Options:
There are four multiple-byte options.

- ✓ Record Route: A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.

- ✓ Strict Source Route: A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet. Dictation of a route by the source can be useful for several purposes. The sender can choose a route with a specific type of service, such as minimum delay or maximum throughput.
  If a datagram specifies a strict source route, all the routers defined in the option must be visited by the datagram. A router must not be visited if its IPv4 address is not listed in the datagram. If the datagram visits a router that is not on the list, the datagram is discarded, and an error message is issued. If the datagram arrives at the destination and some of the entries were not visited, it will also be discarded, and an error message issued.

- ✓ Loose Source Route: A loose source route option is like the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.

- ✓ Timestamp: A timestamp option is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time. Knowing the time, a datagram is processed can help users and managers track the behaviour of the routers in the Internet. We can estimate the time it takes for a datagram to go from one router to another. We say estimate because, although all routers may use Universal time, their local clocks may not be synchronized.

Security of IPv4 Datagrams

The IPv4 protocol, as well as the whole Internet, was started when the Internet users trusted each other. No security was provided for the IPv4 protocol. Today, the situation is different; the Internet is not secure anymore. There are three security issues that are particularly applicable to the IP protocol: packet sniffing, packet modification, and IP spoofing.

✓ Packet Sniffing:

An intruder may intercept an IP packet and make a copy of it. Packet sniffing is a passive attack, in which the attacker does not change the contents of the packet. This type of attack is exceedingly difficult to detect because the sender and the receiver may never know that the packet has been copied. Although packet sniffing cannot be stopped, encryption of the packet can make the attacker's effort useless. The attacker may still sniff the packet, but the content is not detectable.

✓ Packet Modification:

The second type of attack is to modify the packet. The attacker intercepts the packet, changes its contents, and sends the new packet to the receiver. The receiver believes that the packet is coming from the original sender. This type of attack can be detected using a data integrity mechanism. The receiver, before opening and using the contents of the message, can use this mechanism to make sure that the packet has not been changed during the transmission.

✓ IP Spoofing:

An attacker can masquerade as somebody else and create an IP packet that carries the source address of another computer. An attacker can send an IP packet to a bank pretending that it is coming from one of the customers. This type of attack can be prevented using an origin authentication mechanism.

✓ IPSec:

The IP packets today can be protected from the previously mentioned attacks using a protocol called IPSec (IP Security). This protocol, which is used in conjunction with the IP protocol, creates a connection-oriented service between two entities in which they can exchange IP packets without worrying about the three attacks discussed above.

- Defining Algorithms and Keys: The two entities that want to create a secure channel between themselves can agree on some available algorithms and keys to be used for security purposes.
- Packet Encryption: The packets exchanged between two parties can be encrypted for privacy using one of the encryption algorithms and a shared key agreed upon in the first step. This makes the packet sniffing attack useless.
- Data Integrity: Data integrity guarantees that the packet is not modified during the transmission. If the received packet does not pass the data integrity test, it is discarded. This prevents the second attack, packet modification, described above.
- Origin Authentication: IPSec can authenticate the origin of the packet to be sure that the packet is not created by an imposter. This can prevent IP spoofing attacks as described above.

UNICAST ROUTING:

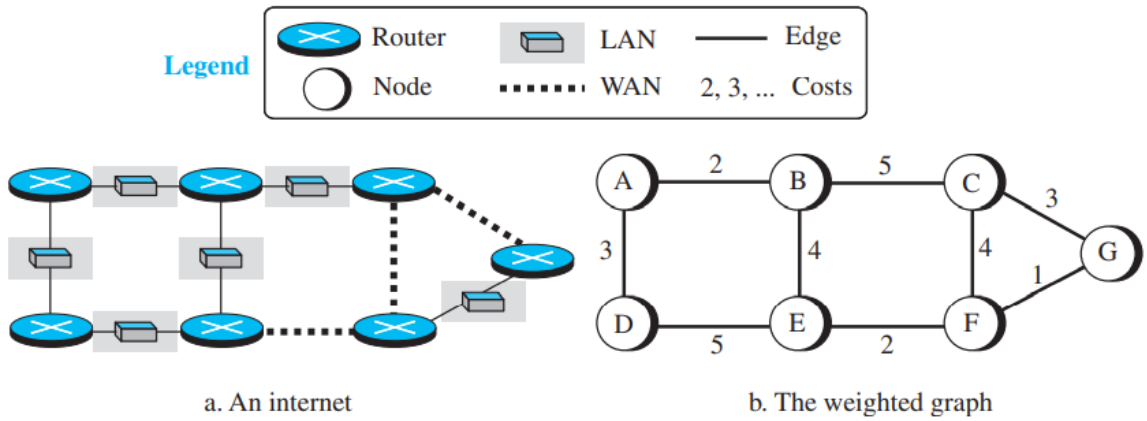
Unicast routing in the Internet, with many routers and a huge number of hosts, can be done only by using hierarchical routing: routing in several steps using different routing algorithms.

General Idea:

- ✓ In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
- ✓ The source host needs no forwarding table because it delivers its packet to the default router in its local network.
- ✓ The destination host needs no forwarding table either because it receives the packet from its default router in its local network.
- ✓ This means that only the routers that glue together the networks in the internet need forwarding tables.
- ✓ Routing a packet from its source to its destination means routing the packet from a source router (the default router of the source host) to a destination router (the router connected to the destination network).

An Internet as a Graph:

- ✓ To find the best route, an internet can be modelled as a graph. A graph in computer science is a set of nodes and edges (lines) that connect the nodes.
- ✓ To model an internet as a graph, we can think of each router as a node and each network between a pair of routers as an edge.
- ✓ An internet is modelled as a weighted graph, in which each edge is associated with a cost. If a weighted graph is used to represent a geographical area, the nodes can be cities and the edges can be roads connecting the cities; the weights are distances between cities.
- ✓ In routing the cost of an edge has a different interpretation in different routing protocols. If there is no edge between the nodes, the cost is infinity. Figure shows how an internet can be modelled as a graph.



An internet and its graphical representation

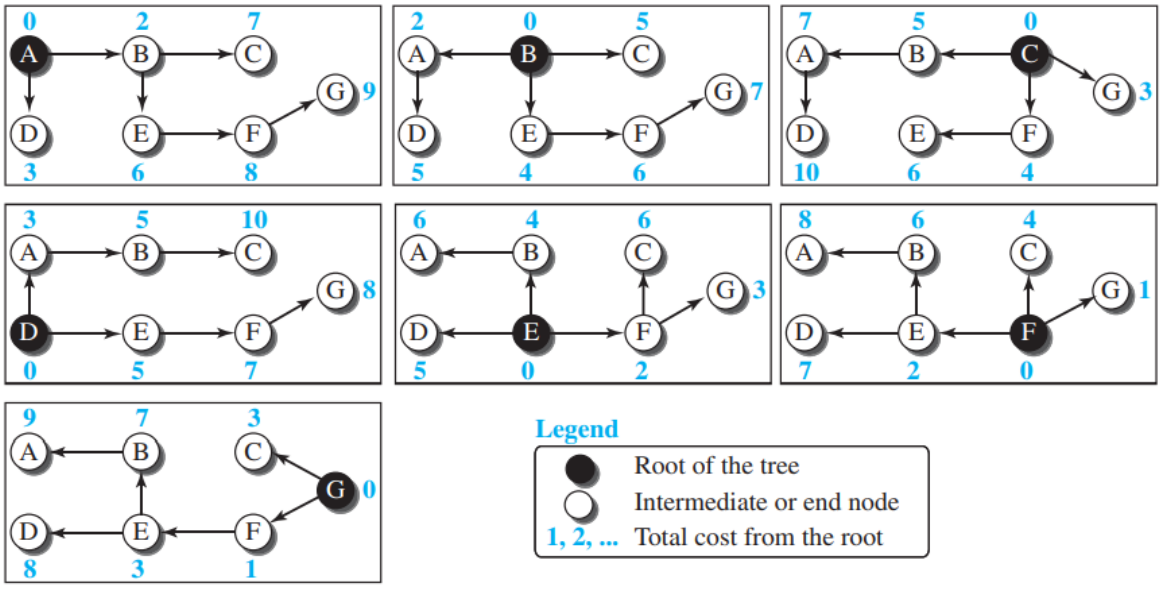
Least-Cost Routing:

- ✓ When an internet is modelled as a weighted graph, one of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two.

- ✓ The source router chooses a route to the destination router in such a way that the total cost for the route is the least cost among all possible routes.
- ✓ In the above figure, the best route between A and E is A-B-E, with the cost of 6. This means that each router needs to find the least-cost route between itself and all the other routers to be able to route a packet using this criterion.

Least-Cost Trees:

- ✓ If there are N routers in an internet, there are (N − 1) least-cost paths from each router to any other router. This means we need N × (N − 1) least-cost paths for the whole internet.
- ✓ If we have only 10 routers in an internet, we need 90 least-cost paths. A better way to see all these paths is to combine them in a least-cost tree.
- ✓ A least-cost tree is a tree with the source router as the root that spans the whole graph (visits all other nodes) and in which the path between the root and any other node is the shortest.
- ✓ In this way, we can have only one shortest-path tree for each node; we have N least-cost trees for the whole internet. Figure shows the seven least-cost trees for the internet in above figure.



Least-cost trees for nodes in the internet of above figure

The least-cost trees for a weighted graph can have several properties if they are created using consistent criteria.

1. The least-cost route from X to Y in X's tree is the inverse of the least-cost route from Y to X in Y's tree; the cost in both directions is the same.

- ✓ For example, in Figure, the route from A to F in A's tree is (A → B → E → F), but the route from F to A in F's tree is (F → E → B → A), which is the inverse of the first route. The cost is 8 in each case.

2. Instead of travelling from X to Z using X's tree, we can travel from X to Y using X's tree and continue from Y to Z using Y's tree.

- ✓ For example, in Figure, we can go from A to G in A's tree using the route (A → B → E → F → G). We can also go from A to E in A's tree (A → B → E) and then continue in E's tree using the route (E → F → G). The combination of the two routes in the second case is the same route as in the first case. The cost in the first case is 9; the cost in the second case is also 9 (6 + 3).

ROUTING ALGORITHMS

Distance-Vector Routing:

In distance-vector routing, the first thing each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbours. The incomplete trees are exchanged between immediate neighbours to make the trees more and more complete and to represent the whole internet.

Bellman-Ford Equation:

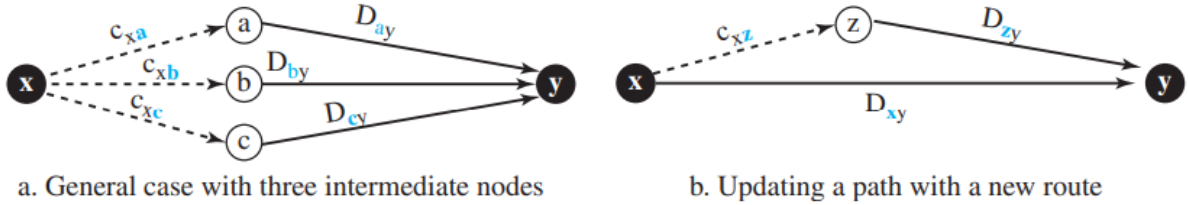
The heart of distance-vector routing is the famous Bellman-Ford equation. This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, . . .) when the costs between the source and the intermediary nodes and the least costs between the intermediary nodes and the destination are given.
The following shows the general case in which D_{ij} is the shortest distance and c_{ij} is the cost between nodes i and j.

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

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, as shown below:

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

The figure below shows the idea graphically for both cases.



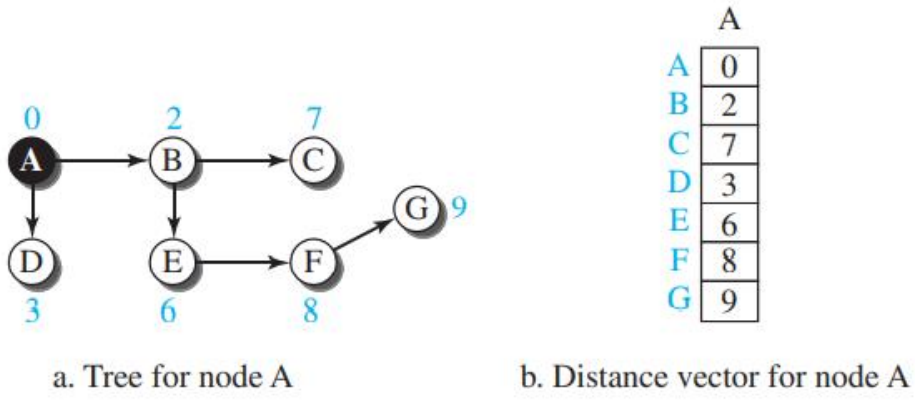
a. General case with three intermediate nodes b. Updating a path with a new route

Graphical idea behind Bellman-Ford equation

Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths. In the figure, 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 even think of this equation as the builder of a new least-cost tree from previously established least-cost trees if we use the equation repeatedly. The use of this equation in distance-vector routing is a witness that this method also uses least-cost trees, but this use may be in the background.

Distance Vectors:

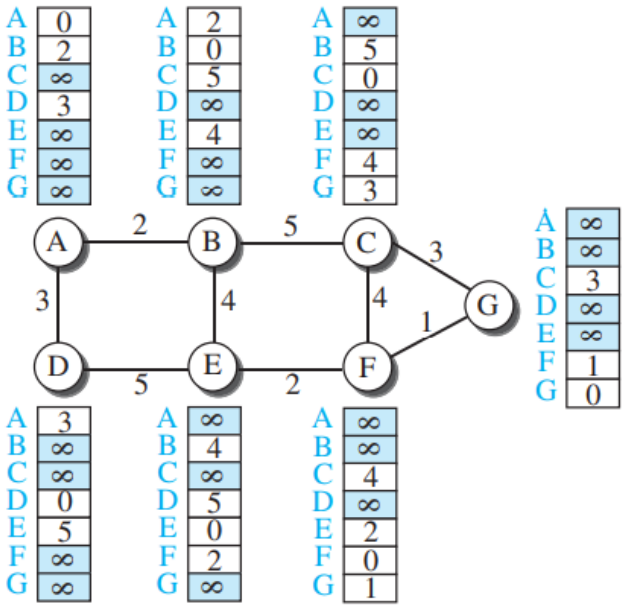
- ✓ The concept of a distance vector is the rationale for the name distance-vector routing. A least-cost tree is a combination of least-cost paths from the root of the tree to all destinations. These paths are graphically glued together to form the tree.
- ✓ Distance-vector routing unglues these paths and creates a distance vector, a one-dimensional array to represent the tree. Figure shows the tree for node A in the internet in first figure and the corresponding distance vector



a. Tree for node A b. Distance vector for node A

The distance vector corresponding to a tree

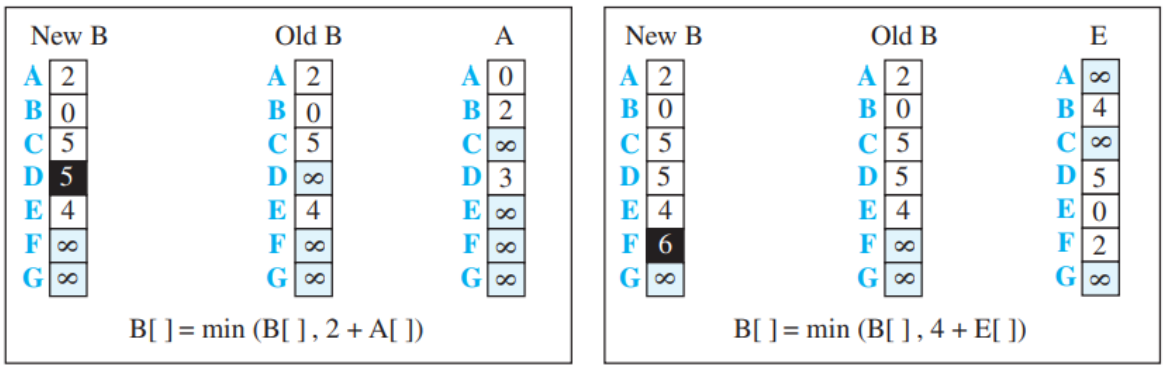
- ✓ The name of the distance vector defines the root, the indexes define the destinations, and the value of each cell defines the least cost from the root to the destination.
- ✓ A distance vector does not give the path to the destinations as the least-cost tree does; it gives only the least costs to the destinations.
- ✓ 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 neighbourhood.
- ✓ The node sends some greeting messages out of its interfaces and discovers the identity of the immediate neighbours and the distance between itself and each neighbour.
- ✓ 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.
- ✓ Figure shows all distance vectors for our internet. However, we need to mention that 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.



The first distance vector for an internet

- ✓ 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.
- ✓ To improve these vectors, the nodes in the internet need to help each other by exchanging information. After each node has created its vector, it sends a copy of the vector to all its immediate neighbours.
- ✓ After a node receives a distance vector from a neighbour, it updates its distance vector using the Bellman-Ford equation (second case).
- ✓ However, we need to understand that we need to update, not only one least cost, but N of them in which N is the number of the nodes in the internet.
- ✓ If we are using a program, we can do this using a loop; if we are showing the concept on paper, we can show the whole vector instead of the N separate equations.

We show the whole vector instead of seven equations for each update in Figure.



a. First event: B receives a copy of A's vector.

b. Second event: B receives a copy of E's vector.

Note:
X[ ]: the whole vector

Updating distance vectors

- ✓ The figure shows two asynchronous events, happening one after another with some time in between. 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.
- ✓ 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.
- ✓ We need to remember that after updating a node, it immediately sends its updated vector to all neighbours. Even if its neighbours have received the previous vector, the updated one may help more.

Distance-Vector Routing Algorithm:

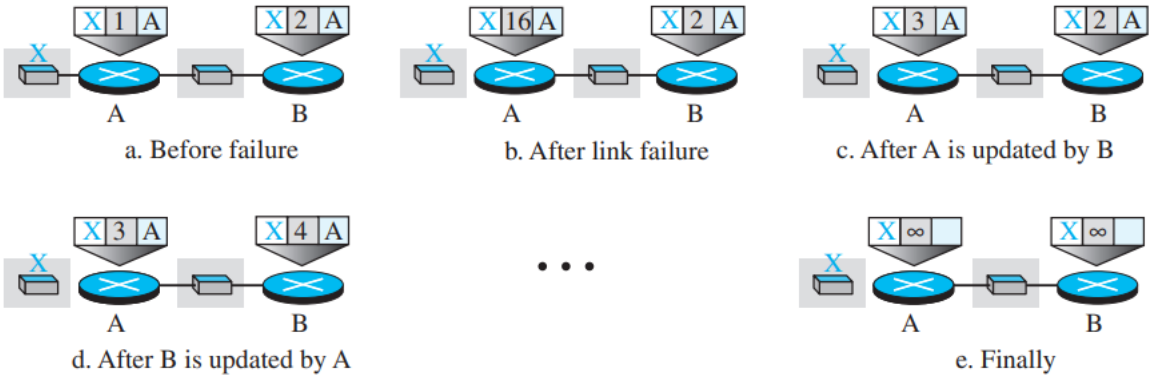
Now we can give a simplified pseudocode for the distance-vector routing algorithm, as shown. The algorithm is run by its node independently and asynchronously.

```
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 neighbour)
8.              D[y] = c[myself][y]
9.     else
10.             D[y] = ∞
11. }
12.  send vector {D[1], D[2], …, D[N]} to all neighbours
13. // Update (improve the vector with the vector received from a neighbour)
14. repeat (forever)
15. {
16. wait (for a vector D_w from a neighbour 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 neighbours
23. }
24. }// End of Distance Vector
```

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 neighbour. The for loop in lines 17 to 20 allows all entries (cells) in the vector to be updated after receiving a new vector. Note that the node sends its vector in line 12, after being initialized, and in line 22, after it is updated.

- Count to Infinity
  - A problem with distance-vector routing is that 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.
  - The problem is referred to as count to infinity. It sometimes takes several updates before the cost for a broken link is recorded as infinity by all routers.

- Two-Node Loop
  - One example of count to infinity is the two-node loop problem. To understand the problem, let us look at the scenario depicted in figure.



Two-node instability

  - The figure shows a system with three nodes. At the beginning, both nodes A and B know how to reach node X. But suddenly, the 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, if B has found a way to reach X, immediately updates its forwarding table.
  - 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. However, 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. A few solutions have been proposed for instability of this kind.

- Split Horizon
  - One solution to instability is called split horizon. In this strategy, 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.
  - In our scenario, node B eliminates the last line of its forwarding table before it sends it to A. In this case, 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 node A and node B know that X is not reachable.

➢ Poison Reverse
- ❖ Using the split-horizon strategy has one drawback. Normally, the corresponding protocol uses a timer, and 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.
- ❖ In the poison 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."

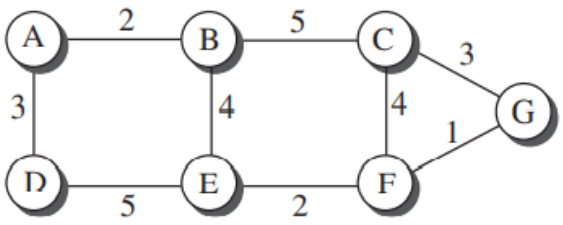
➢ Three-Node Instability
- ❖ The two-node instability can be avoided using split horizon combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed.

Link-State Routing

The link-state (LS) routing uses the term link-state to define the characteristic of a link (an edge) that represents a network in the internet. 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):

- ✓ To create a least-cost tree with this method, 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 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 figure shows an example of an LSDB for the graph in first figure. 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.



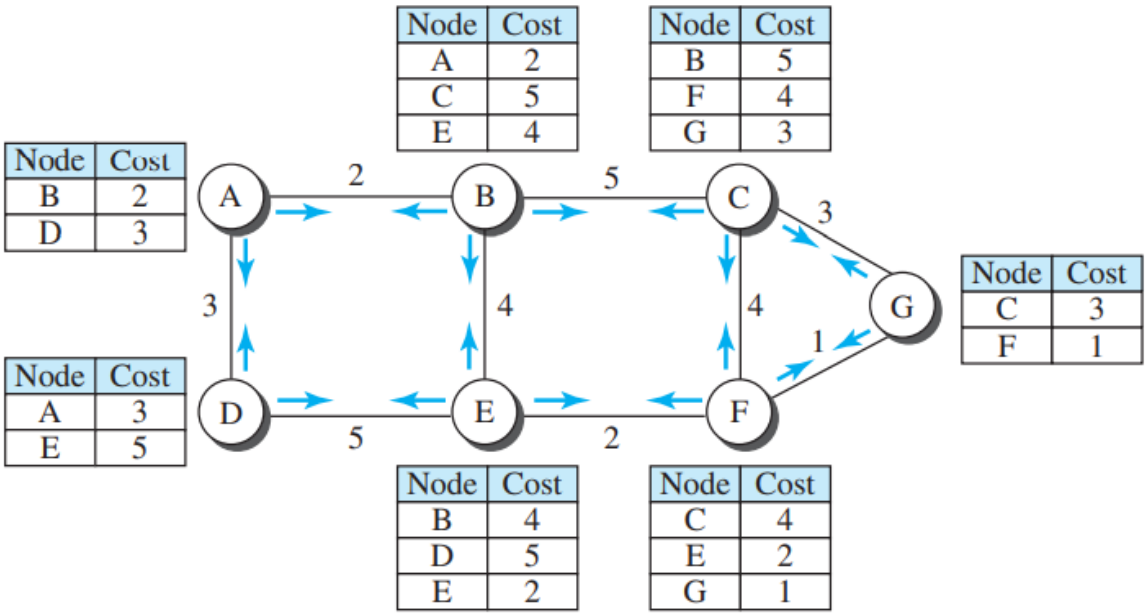
a. The weighted graph

	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

Example of a link-state database

- ✓ Each node can create this LSDB that contains information about the whole internet by a process called flooding. Each node can send some greeting messages to all its immediate neighbours (those nodes to which it is connected directly) to collect two pieces of information for each neighbouring node: the identity of the node and the cost of the link.
- ✓ The combination of these two pieces of information is called the LS packet (LSP); the LSP is sent out of each interface, as shown in Figure for our internet in first figure.
- ✓ 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). We need to convince ourselves that, after receiving all new LSPs, each node creates the comprehensive LSDB as shown in Figure. This LSDB is the same for each node and shows the whole map of the internet. In other words, a node can make the whole map if it needs to, using this LSDB.



LSPs created and sent out by each node to build LSDB

- ✓ We can compare the link-state routing algorithm with the distance-vector routing algorithm. In the distance-vector routing algorithm, each router tells its neighbours what it knows about the whole internet; in the link-state routing algorithm, each router tells the whole internet what it knows about its neighbours.

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 Algorithm. This iterative algorithm uses the following steps:

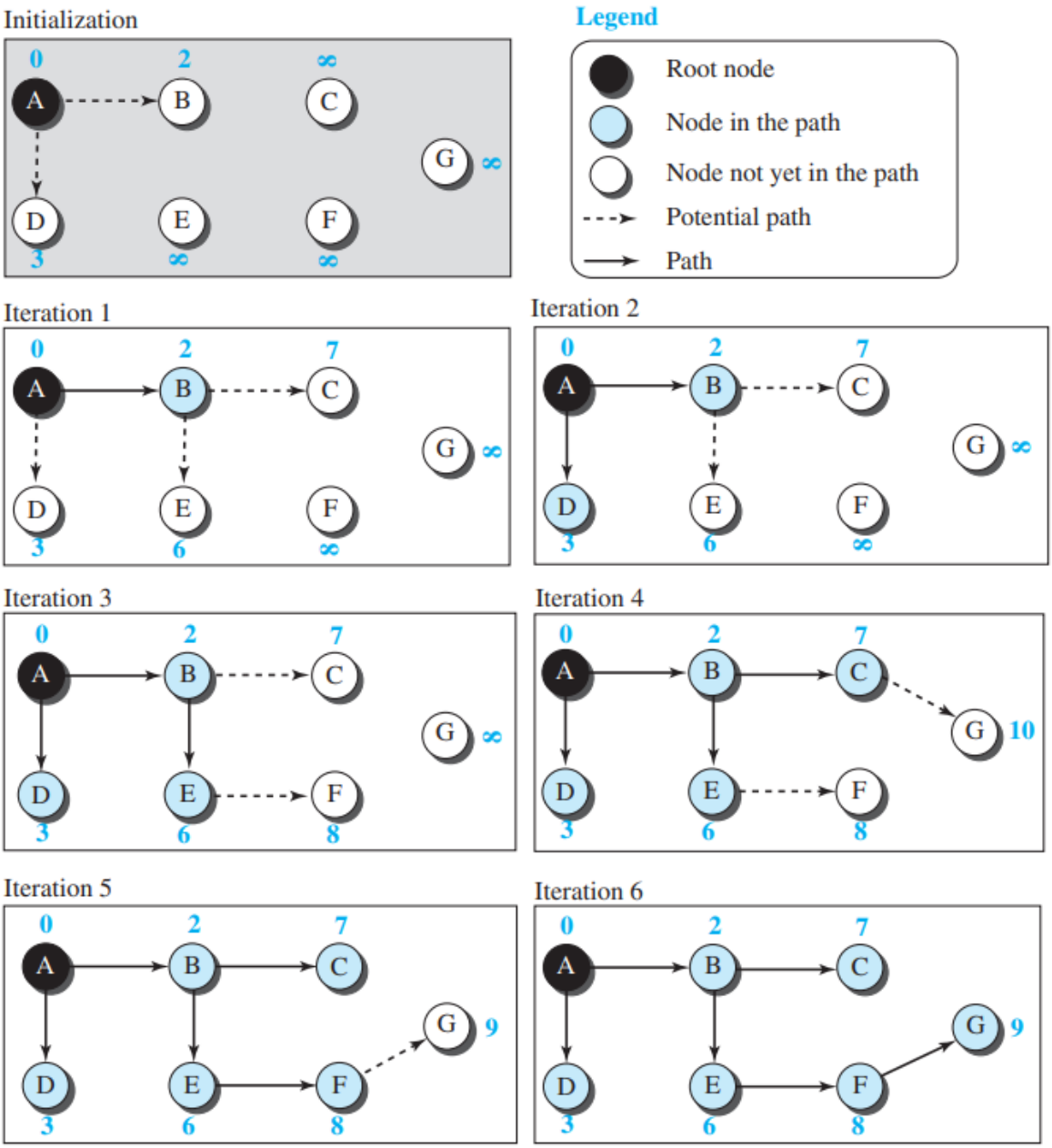
1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.

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 neighbour)
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 neighbours of w
20.         for (every node x, which is a neighbour 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.  } // End of Dijkstra
```

Lines 4 to 13 implement step 1 in the algorithm. Lines 16 to 23 implement step 2 in the algorithm. Step 2 is repeated until all nodes are added to the tree. Figure shows the formation of the least-cost tree for the graph in Figure: *Example of a link-state database* using Dijkstra's algorithm. We need to go through an initialization step and six iterations to find the least-cost tree.



Least-cost tree

Path-Vector Routing:

Both link-state and distance-vector routing are based on the least-cost goal. However, there are instances where this goal is not the priority.

For example, assume that there are some routers in the internet that a sender wants to prevent its packets from going through or a router may belong to an organization that does not provide enough security, or it may belong to a commercial rival of the sender which might inspect the packets for obtaining information.

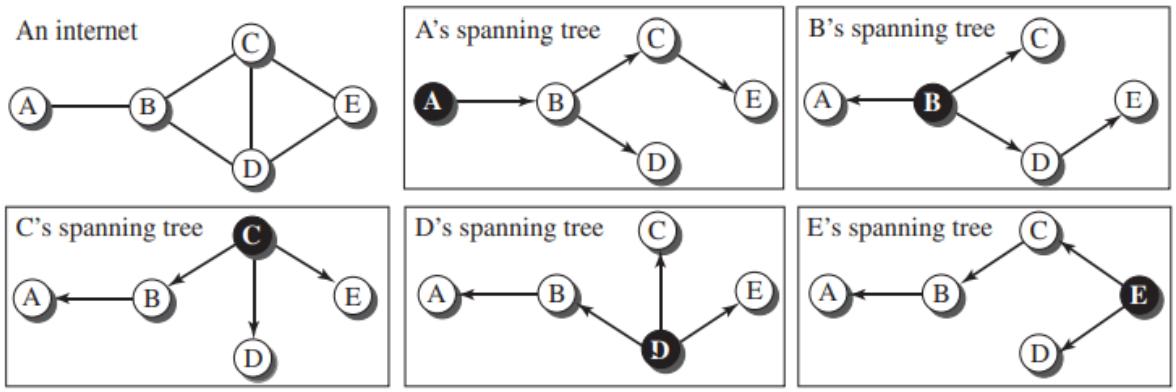
Least-cost routing does not prevent a packet from passing through an area when that area is in the least-cost path. The least-cost goal, applied by LS or DV routing, does not allow a sender to apply specific policies to the route a packet may take.

Aside from safety and security, there are occasions in which the goal of routing is merely reachability: to allow the packet to reach its destination more efficiently without assigning costs to the route.

Path-vector routing does not have the drawbacks of LS or DV routing as described above because it is not based on least-cost routing. The best route is determined by the source using the policy it imposes on the route. The source can control the path. Although path-vector routing is not actually used in an internet and is mostly designed to route a packet between ISPs.

Spanning Trees:

- ✓ In path-vector routing, the path from a source to all destinations is also determined by the best spanning tree.
- ✓ The best spanning tree is not the least-cost tree; it is the tree determined by the source when it imposes its own policy.
- ✓ If there is more than one route to a destination, the source can choose the route that meets its policy best. A source may apply several policies at the same time.
- ✓ One of the common policies uses the minimum number of nodes to be visited . Another common policy is to avoid some nodes as the middle node in a route.
- ✓ Figure shows a small internet with only five nodes. Each source has created its own spanning tree that meets its policy.
- ✓ The policy imposed by all sources is to use the minimum number of nodes to reach a destination. The spanning tree selected by A and E is such that the communication does not pass-through D as a middle node. Similarly, the spanning tree selected by B is such that the communication does not pass-through C as a middle node.



Spanning trees in path-vector routing

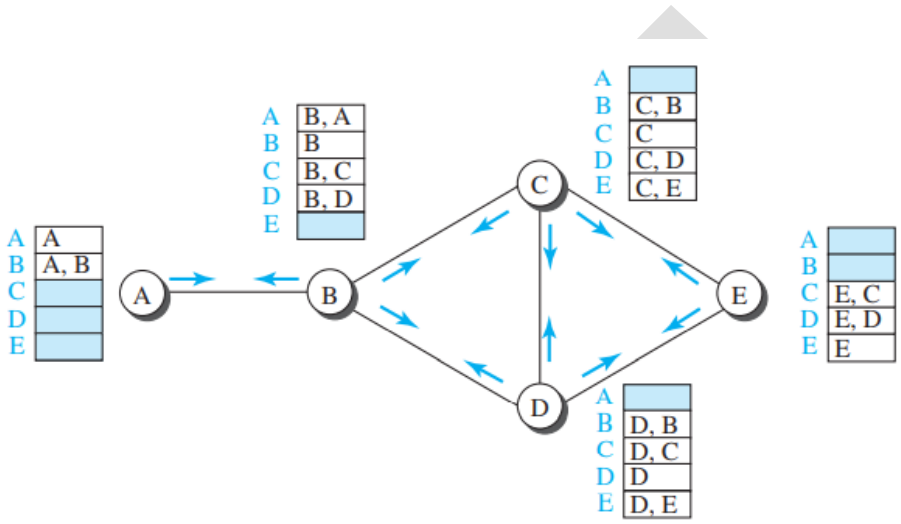
Creation of Spanning Trees:

- ✓ Path-vector routing, like distance-vector routing, is an asynchronous and distributed routing algorithm. The spanning trees are made, gradually and asynchronously, by each node.
- ✓ When a node is booted, it creates a path vector based on the information it can obtain about its immediate neighbour. A node sends greeting messages to its immediate neighbours to collect these pieces of information.
- ✓ Figure shows all these path vectors for our internet in Figure above.
- ✓ Note that we do not mean that all these tables are created simultaneously; they are created when each node is booted.

- ✓ The figure also shows how these path vectors are sent to immediate neighbours after they have been created (arrows). Each node, after the creation of the initial path vector, sends it to all its immediate neighbours.
- ✓ Each node, when it receives a path vector from a neighbour, updates its path vector using an equation like the Bellman-Ford, but applying its own policy instead of looking for the least cost. We can define this equation as

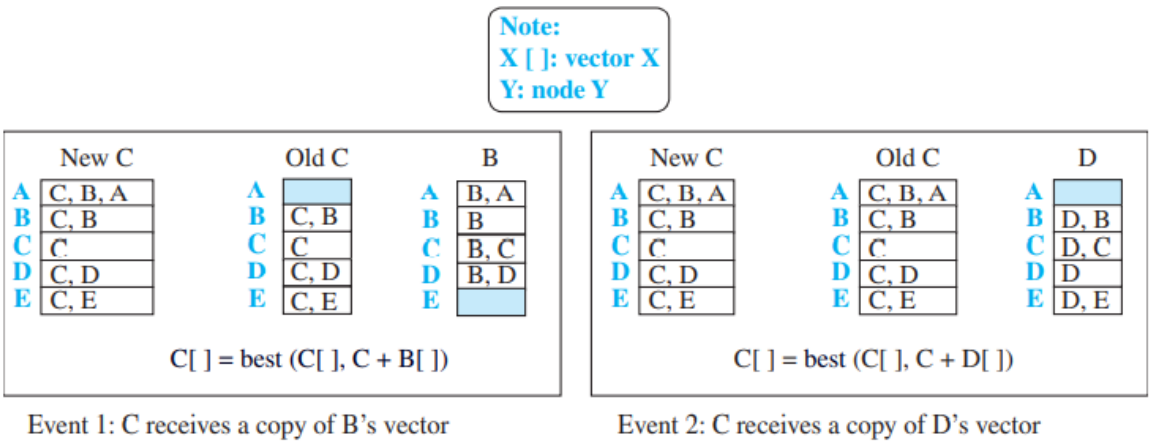
$$Path(x, y) = best\ \{Path(x, y), [(x + Path(v, y))]\} \qquad \text{for all v's in the internet}$$

- ✓ In this equation, the operator (+) means to add x to the beginning of the path. We also need to be cautious to avoid adding a node to an empty path because an empty path means one that does not exist.



Path vectors made at booting time

- ✓ The policy is defined by selecting the best of multiple paths. Path-vector routing also imposes one more condition on this equation: If Path (v, y) includes x, that path is discarded to avoid a loop in the path. In other words, x does not want to visit itself when it selects a path to y.
- ✓ Figure shows the path vector of node C after two events. In the first event, node C receives a copy of B's vector, which improves its vector: now it knows how to reach node A. In the second event, node C receives a copy of D's vector, which does not change its vector. As a matter of fact, the vector for node C after the first event is stabilized and serves as its forwarding table.



Event 1: C receives a copy of B's vector

Event 2: C receives a copy of D's vector

Updating path vectors

Path-Vector Algorithm:

```
1.  Path_Vector_Routing ( )
2.  {
3.  // Initialization
4.      for (y = 1 to N)
5.      {
6.          if (y is myself)
7.          Path[y] = myself
8.          else if (y is a neighbour)
9.          Path[y] = myself + neighbour node
10.         else
11.         Path[y] = empty
12.     }
13. Send vector {Path[1], Path[2], …, Path[y]} to all neighbours
14. // Update
15. repeat (forever)
16.     {
17.     wait (for a vector Pathw from a neighbour w)
18.     for (y = 1 to N)
19.         {
20.             if (Pathw includes myself)
21.             discard the path            // Avoid any loop
22.             else
23.             Path[y] = best {Path[y], (myself + Pathw[y])}
24.         }
25.         If (there is a change in the vector)
26.         Send vector {Path[1], Path[2], …, Path[y]} to all neighbours
27.     }
28.  } // End of Path Vector
```

Lines 4 to 12 show the initialization for the node. Lines 17 to 24 show how the node updates its vector after receiving a vector from the neighbour. The update process is repeated forever.

****