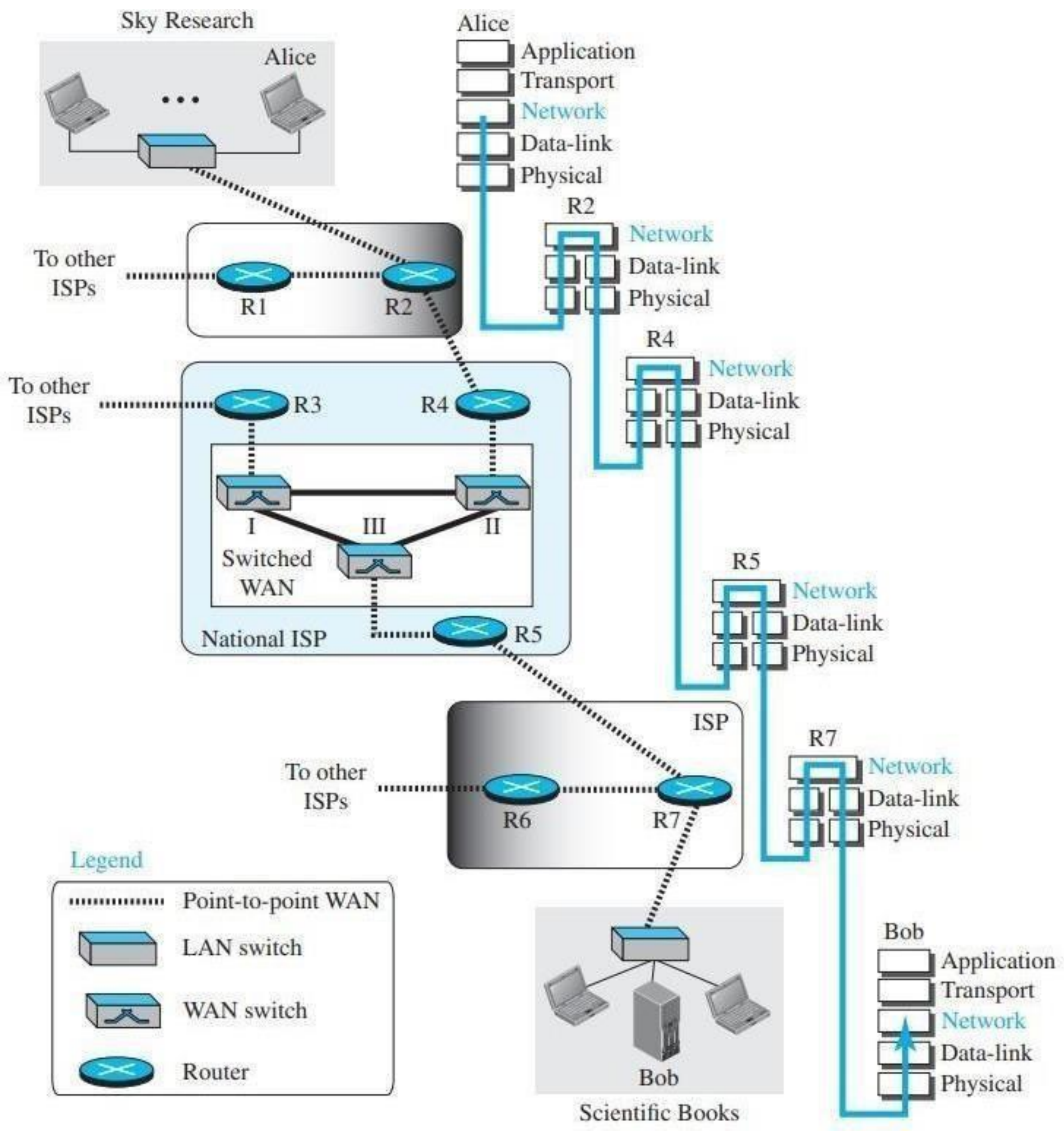
MODULE 3

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.

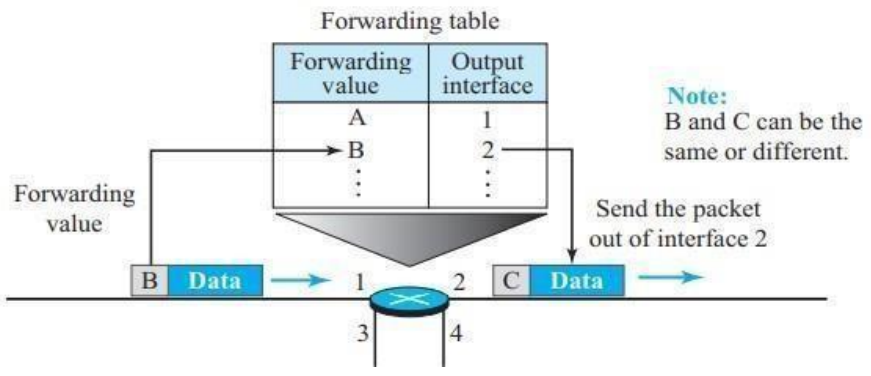
The network layer is involved at the source host, destination host, and all routers in the path (R2, R4, R5, and R7). At the source host (Alice), the network layer accepts a packet from a transport layer, encapsulates the packet in a datagram, and delivers the packet to the data-link layer. At the destination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

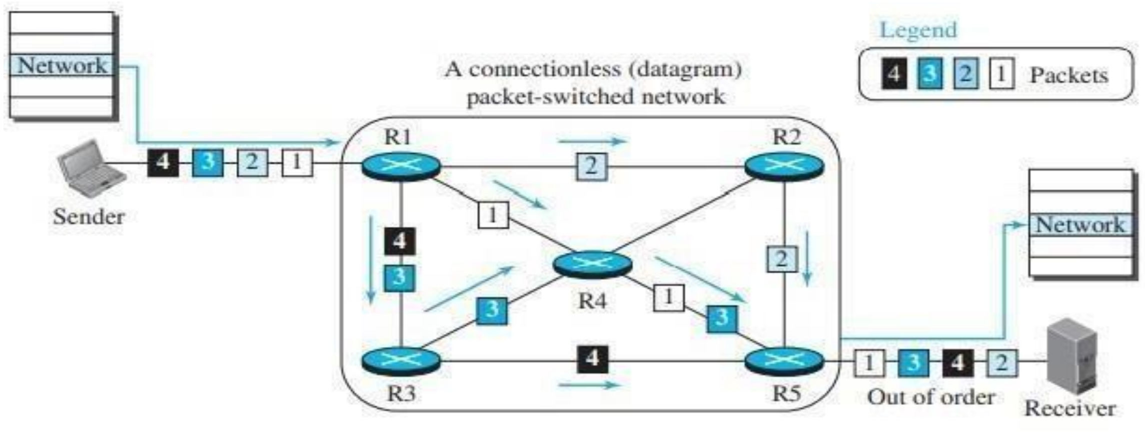
Security was not a concern when the Internet was originally designed because it was used by a small number of users at universities for research activities; other people had no access to the Internet. The network layer was designed with no security provision. Today, security is a big concern. To provide 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

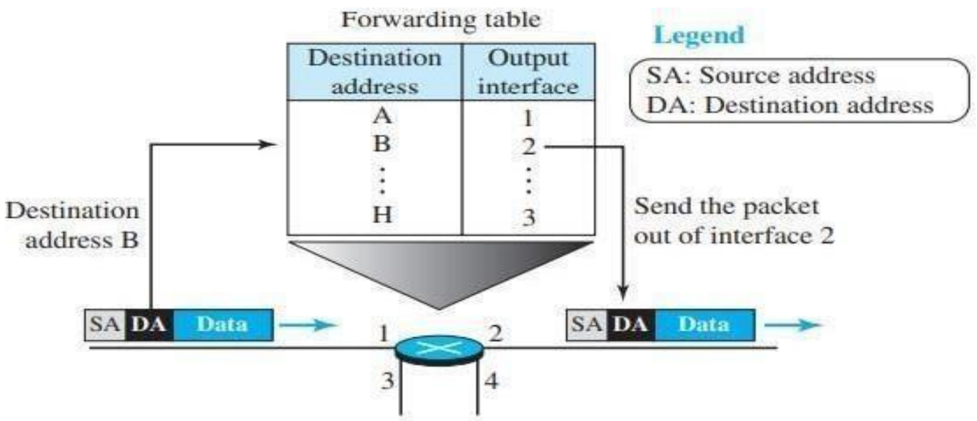
When the Internet started, the network layer was designed to provide a connectionless service in which the network-layer protocol treats each packet independently, with each packet having no relationship to any other packet.
The idea was that the network layer is only responsible for delivery of packets 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.



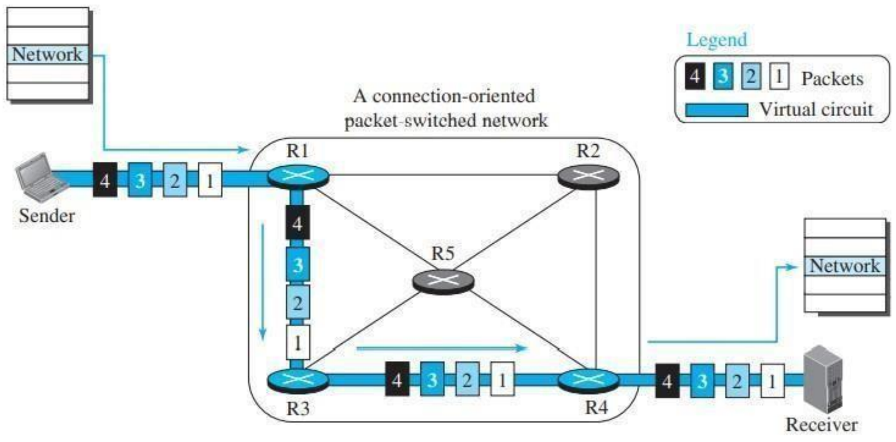
Forwarding process in a router when used in a connectionless network

Virtual-CircuitApproach: Connection-Oriented Service

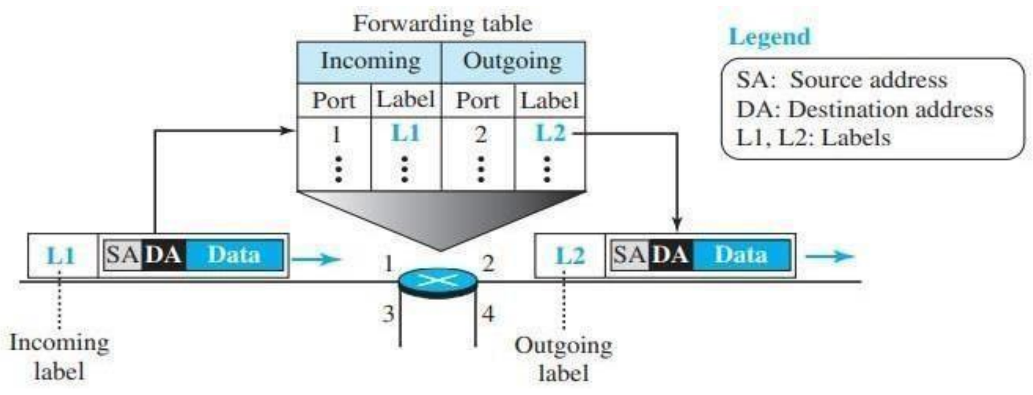
In a connection-oriented service (also called virtual-circuit approach), there is a relationship between all packets belonging to a message. Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams. After connection setup, the datagrams 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

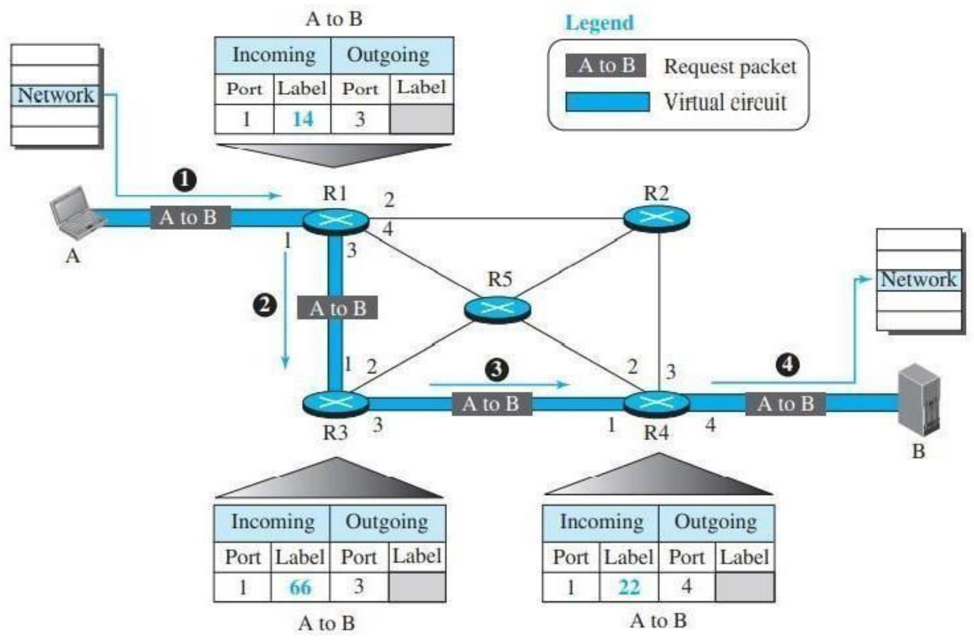
In this case, the forwarding decision is based on the value of the label, or virtual circuit identifier, as it is sometimes called.

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

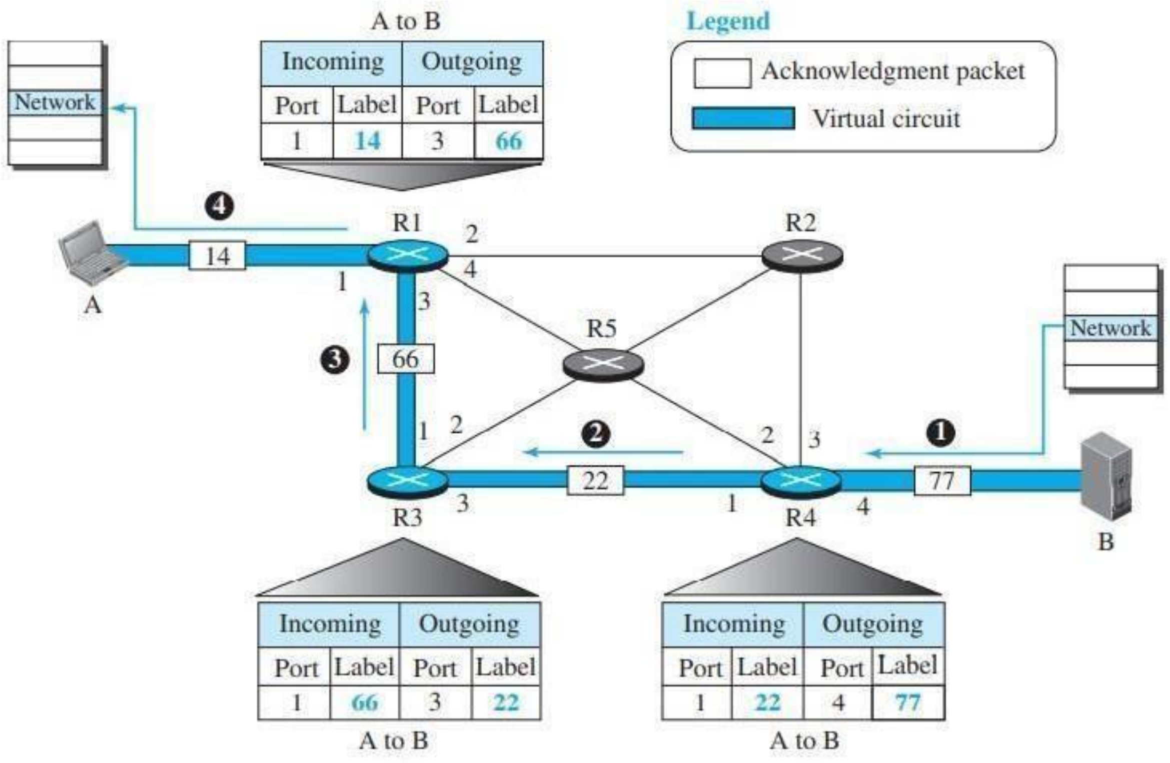
In the setup phase, a router creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to destination B. Two auxiliary packets need to be exchanged between the sender and the receiver: the request packet and the acknowledgment packet.

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.



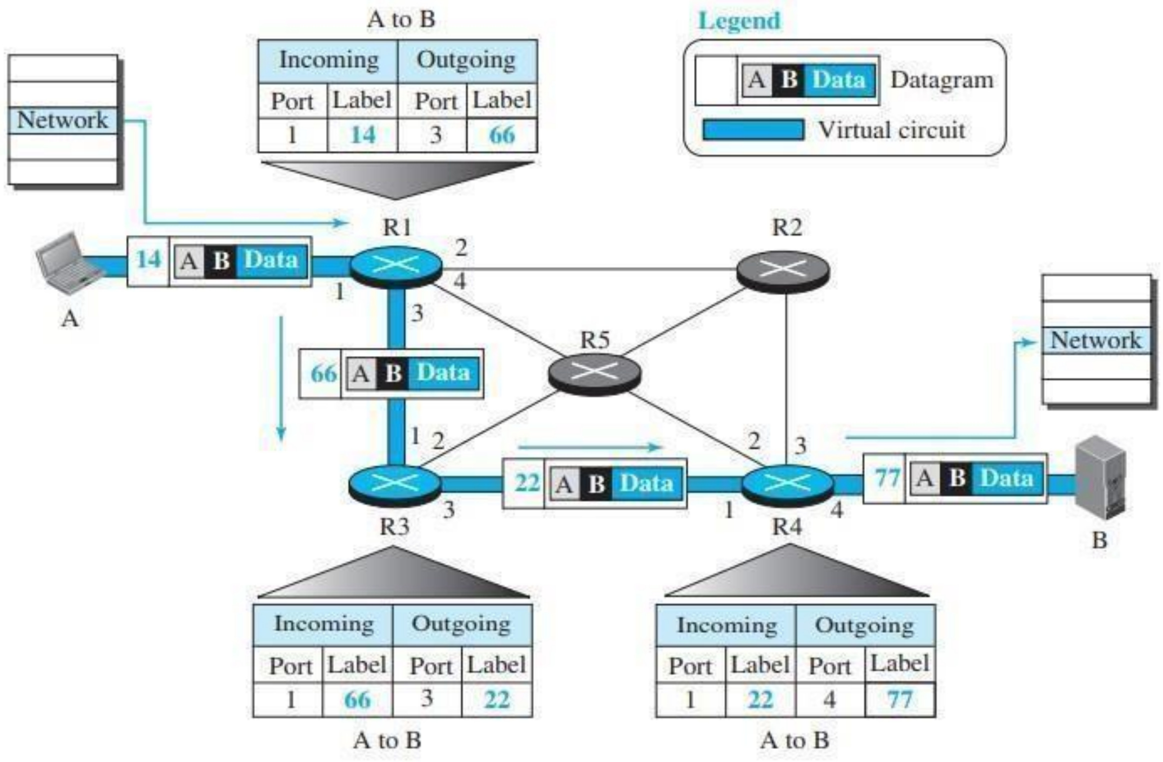
Sending acknowledgments in a virtual-circuit network

- 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.

<u>IPV4 Addresses</u>

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.

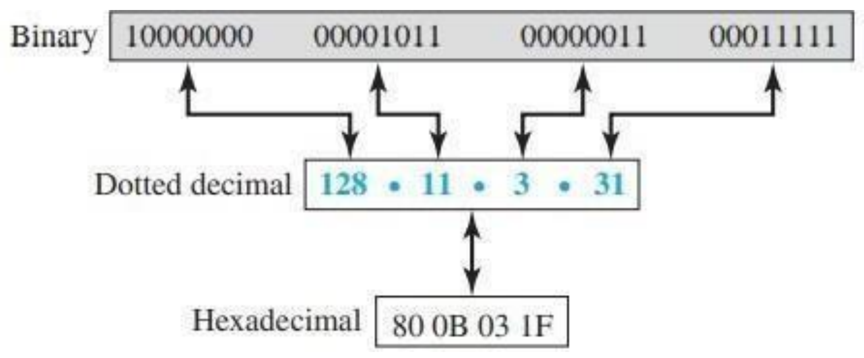
If a protocol uses b bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1).

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.

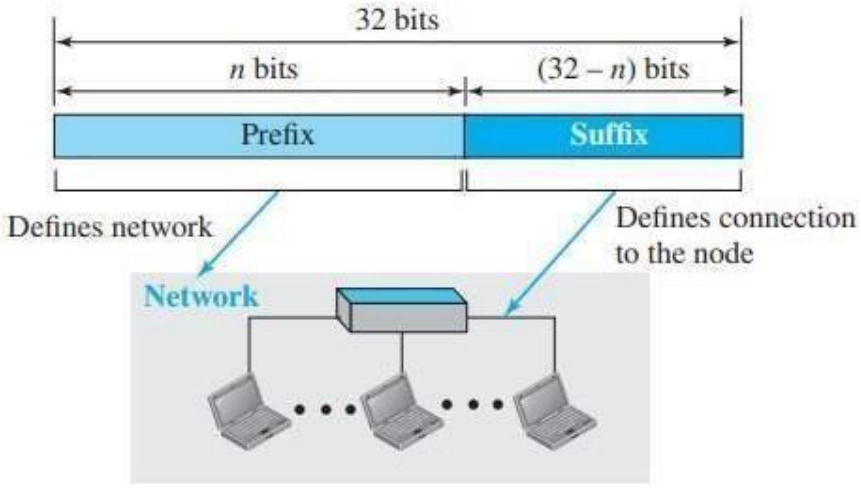


Three different notations in IPv4 addressing

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

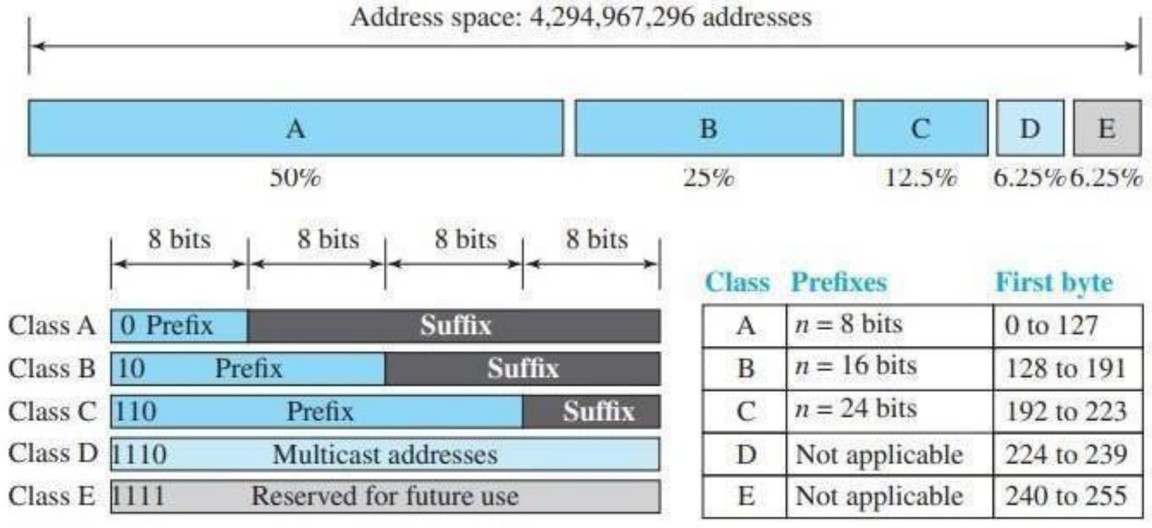
IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one (n = 8, n = 16, and n = 24).

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.

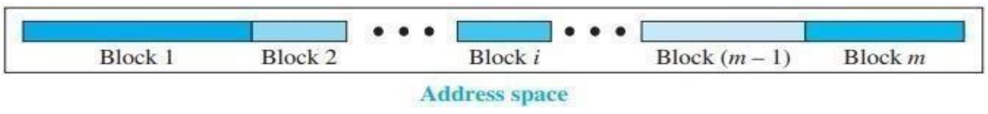
Although the long-range solution has already been devised and is called IPv6, a short-term solution was also devised to use the same address space but to change the distribution of addresses to provide a fair share to each organization.

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.

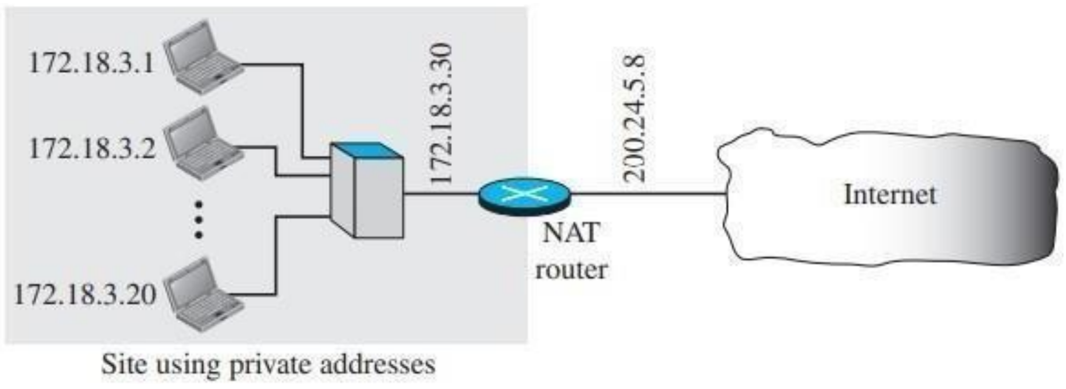
An address in class B can be thought of as a classless address in which the prefix is 16, and so on. In other words, classful addressing is a special case of classless addressing.

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.



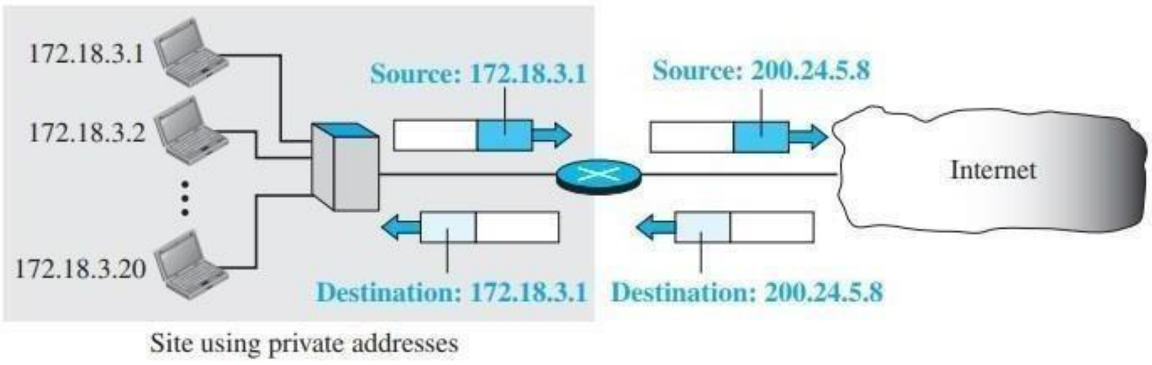
NAT

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.

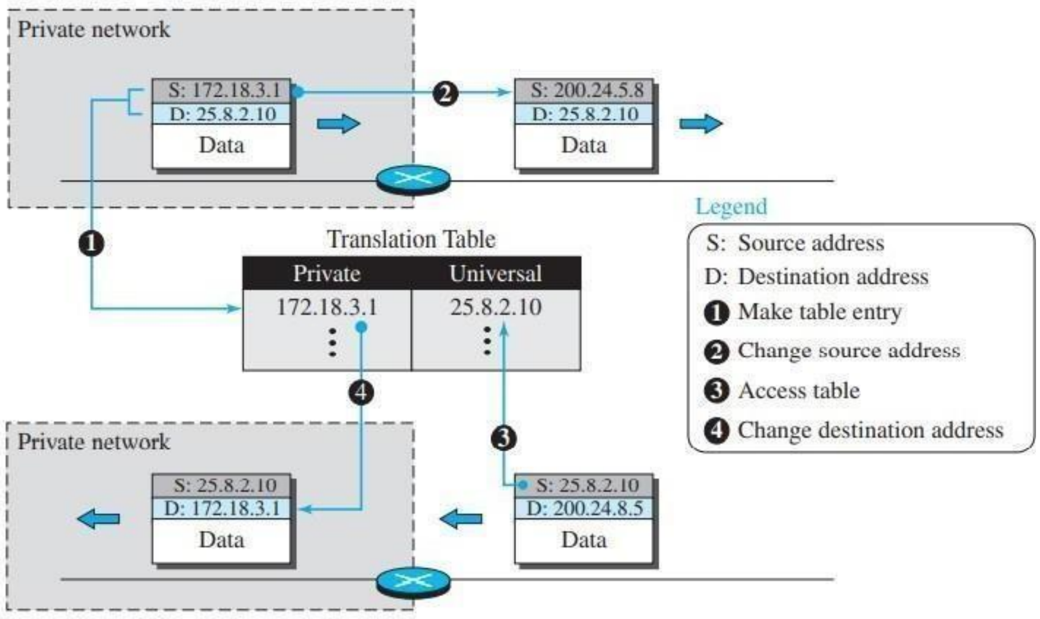


Address translation

Table: Translation

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.

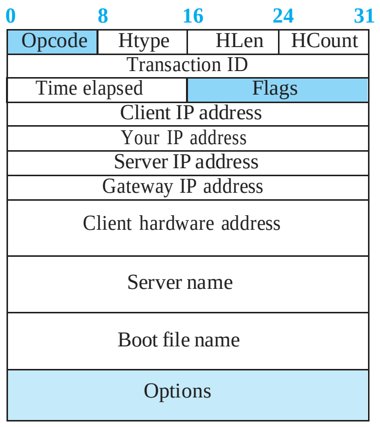
Dynamic Host Configuration Protocol (DHCP)

- 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. However, 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 actually 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-forth of customers use the Internet at the same time.

DHCP Message Format

- DHCP is a client-server protocol in which the client sends a request message and the server returns a response message. Before we discuss the operation of DHCP, let us show the general format of the DHCP message in Figure 18.25.
- Most of the fields are explained in the figure, but we need to discuss the option field, which plays a very 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.

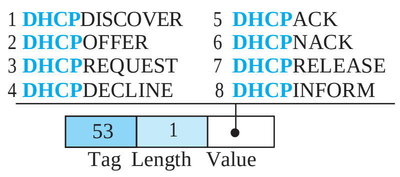
Figure 18.25 *DHCP message format*



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

- An option is composed of three fields: a 1-byte tag field, a 1-byte length field, and a variable-length value field.
- 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 18.26.

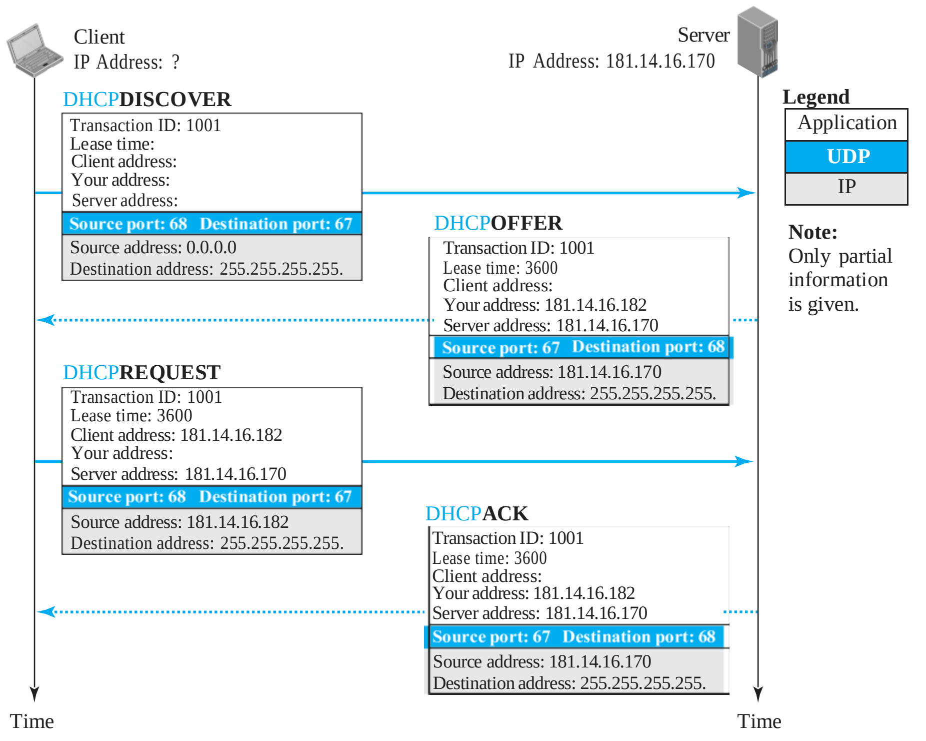
Figure 18.26 *Option format*



DHCP Operation

Figure 18.27 shows a simple scenario.

Figure 18.27 *Operation of DHCP*


Client
IP Address: ?
Server
IP Address: 181.14.16.170
DHCPDISCOVER
Transaction ID: 1001
Lease time:
Client address:
Your address:
Server address:
Source port: 68 Destination port: 67
Source address: 0.0.0.0
Destination address: 255.255.255.255.
DHCPOFFER
Transaction ID: 1001
Lease time: 3600
Client address:
Your address: 181.14.16.182
Server address: 181.14.16.170
Source port: 67 Destination port: 68
Source address: 181.14.16.170
Destination address: 255.255.255.255.
DHCPREQUEST
Transaction ID: 1001
Lease time: 3600
Client address: 181.14.16.182
Your address:
Server address: 181.14.16.170
Source port: 68 Destination port: 67
Source address: 181.14.16.182
Destination address: 255.255.255.255.
DHCPACK
Transaction ID: 1001
Lease time: 3600
Client address:
Your address: 181.14.16.182
Server address: 181.14.16.170
Source port: 67 Destination port: 68
Source address: 181.14.16.170
Destination address: 255.255.255.255.
Legend
Application
UDP
IP
Note: Only partial information is given.
Time
Time

1. The joining host creates a **DHCP**DISCOVER 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. We will discuss the reason for using two well-known port numbers later. 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.
2. The DHCP server or servers (if more than one) responds with a **DHCP**OFFER 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.

3. The joining host receives one or more offers and selects the best of them. The joining host then sends a **DHCP**REQUEST 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.
4. Finally, the selected server responds with a **DHCP**ACK 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 **DHCP**NACK 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

- We said that 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 cli- ents. 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 of the information that a client may need for joining the network. In the **DHCP**ACK 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.

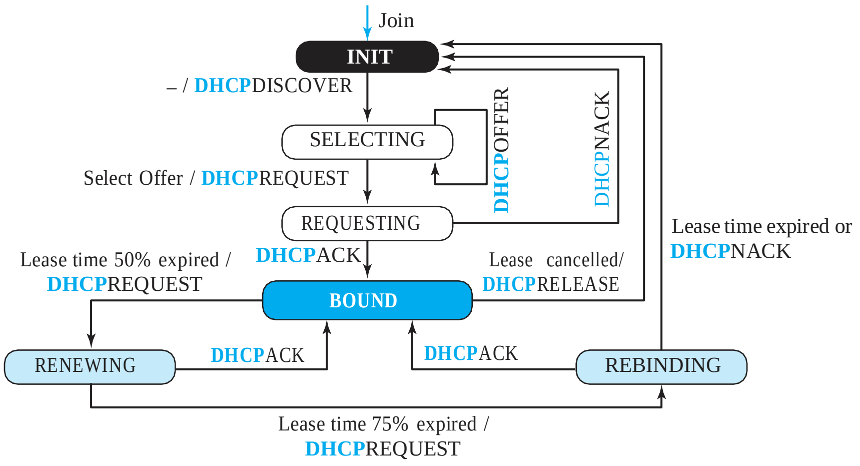
Error Control

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. As we will see in Chapter 24, the use of the checksum in UDP is optional. Second, the DHCP client uses timers and a retransmission policy if it does not receive the DHCP reply to a request. However, 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

The previous scenarios we discussed for the operation of the DHCP were very simple. 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. Figure 18.28 shows the transition diagram with the main states.

Figure 18.28 *FSM for the DHCP client*


Join
INIT
– / DHCPDISCOVER
SELECTING
DHCPOFFER
DHCPNACK
Select Offer / DHCPREQUEST
REQUESTING
Lease time expired or DHCPNACK
Lease time 50% expired / DHCPREQUEST
DHCPACK
Lease cancelled/ DHCPRELEASE
BOUND
DHCPACK
DHCPACK
RENEWING
REBINDING
Lease time 75% expired / DHCPREQUEST

- 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 mayreceive 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), *rebind- ing timer* (set to 75 percent of the lease time), and *expiration timer* (set to the lease time).

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).

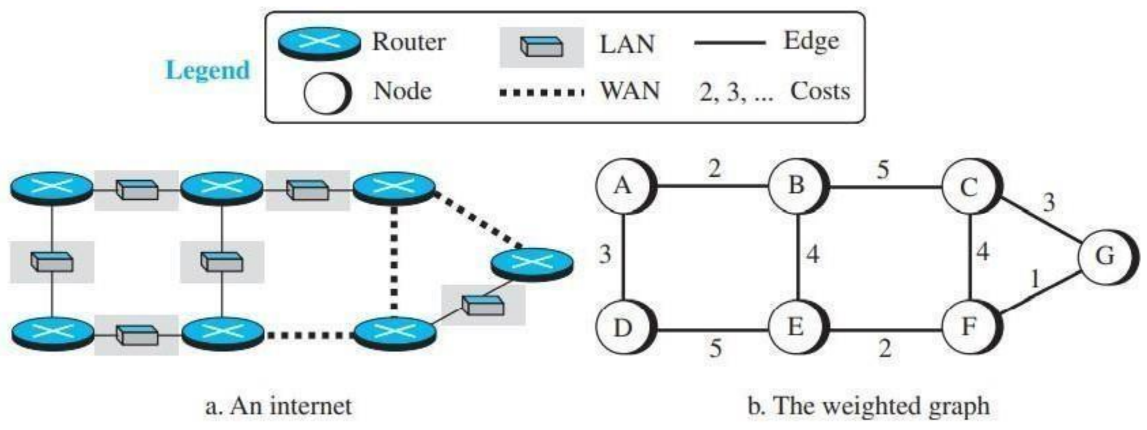
An Internet as a Graph:

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:

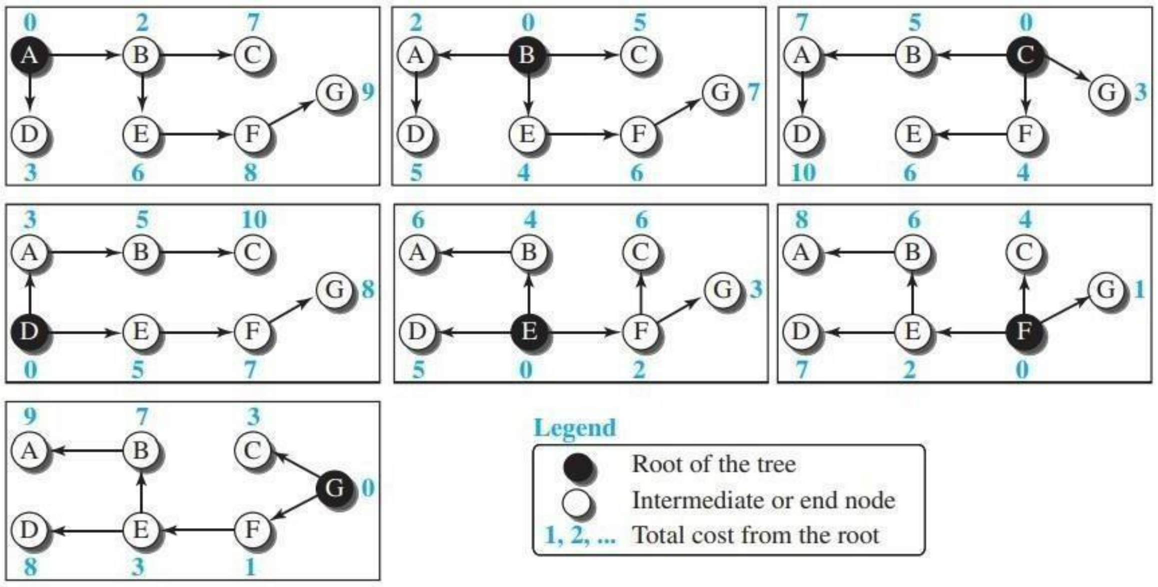
When an internet is modelled as a weighted graph, one of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two.

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.


Legend
● Root of the tree
○ Intermediate or end node
1, 2, ... Total cost from the root

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 continuefrom 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.

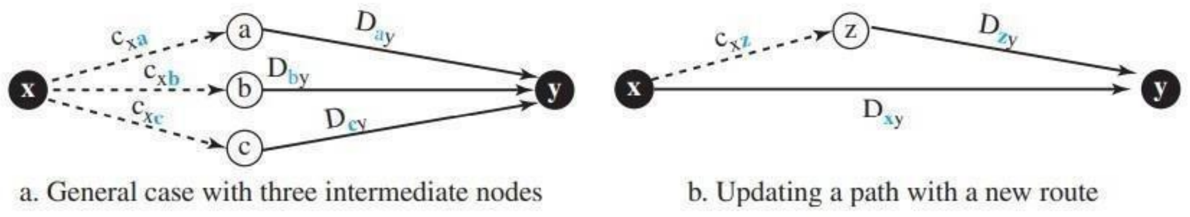
Bellman-Ford Equation:

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.

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

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.



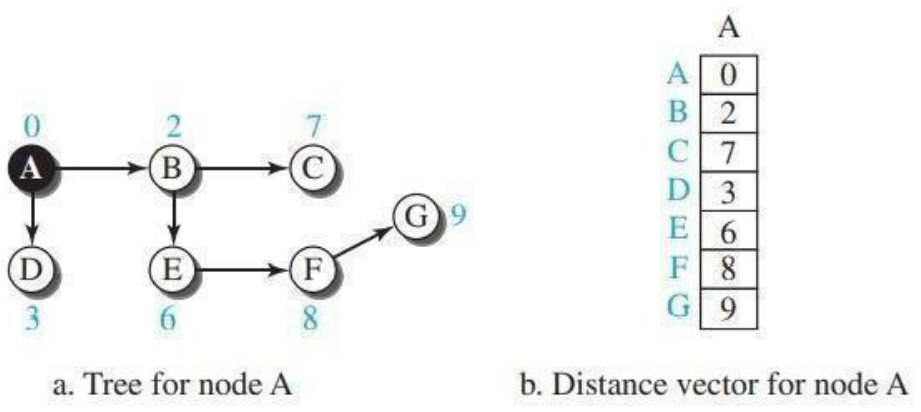
a. General case with three intermediate nodes
b. Updating a path with a new route

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 leastcost 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.

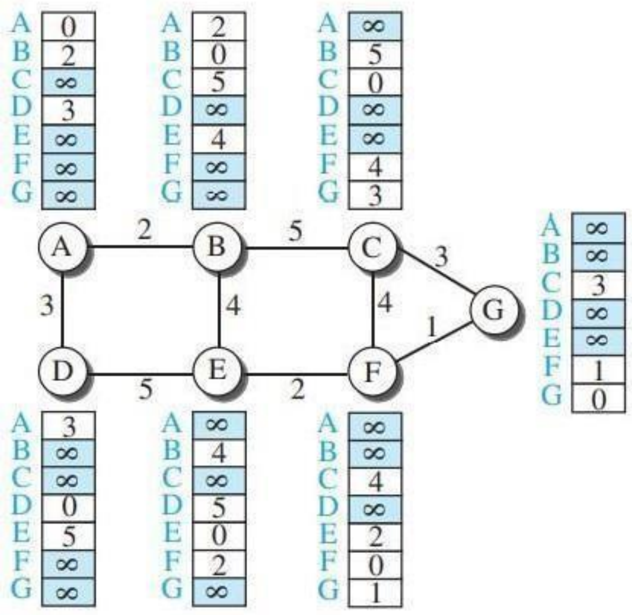
Distance Vectors:

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



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.


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

The first distance vector for an internet

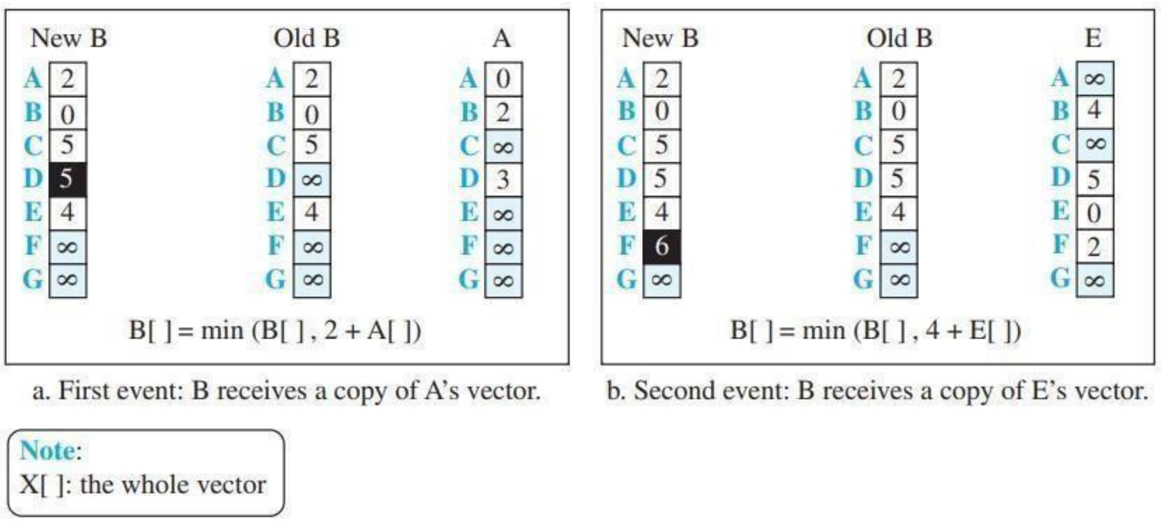
These rudimentary vectors cannot help the internet to effectively forward a packet. For example, node A thinks that it is not connected to node G because the corresponding cell shows the least cost of infinity.

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.


New B
Old B
A
B[ ] = min (B[ ], 2 + A[ ])
a. First event: B receives a copy of A's vector.
New B
Old B
E
B[ ] = min (B[ ], 4 + E[ ])
b. Second event: B receives a copy of E's vector.
Note:
X[ ]: the whole vector

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

	New B	Old B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	6	∞	2
G	∞	∞	∞

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).

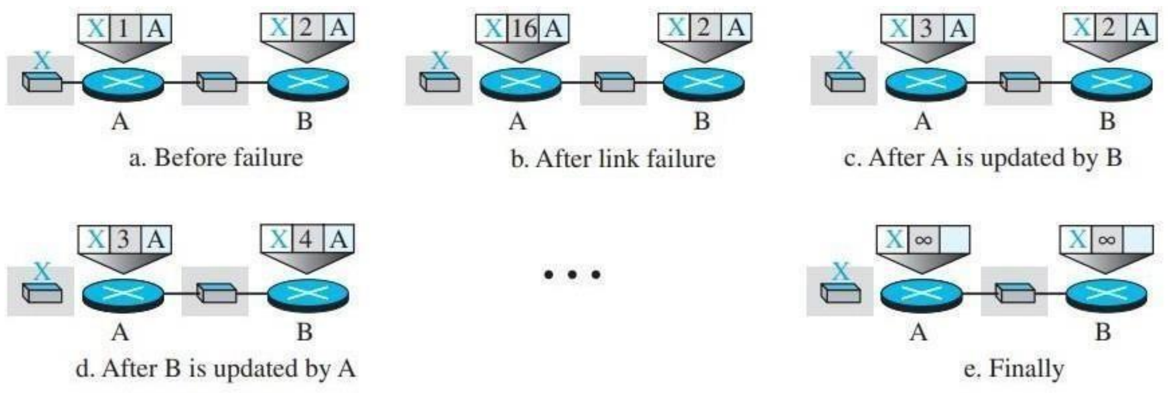
Exchanging vectors eventually stabilizes the system and allows all nodes to find the ultimate least cost between themselves and any other node.

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.

- 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."

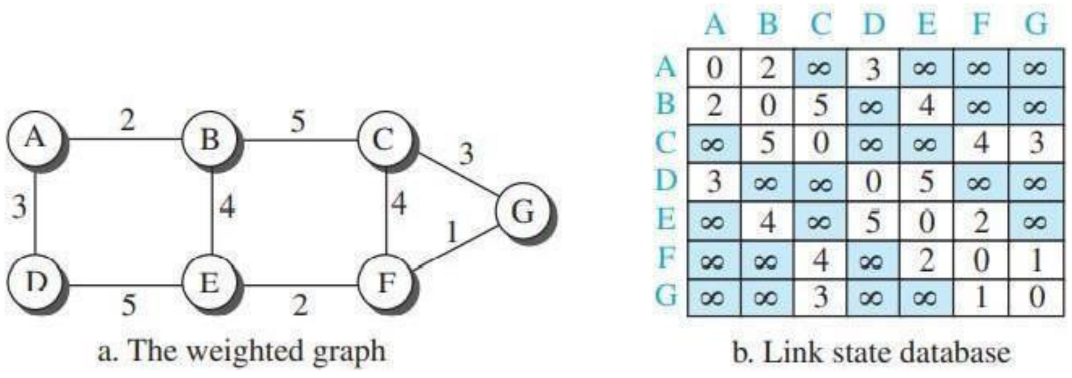
- Three-Node Instability
  - The two-node instability can be avoided using split horizon combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed.

<u>Link-State Routing</u>

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	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

Example of a link-state database

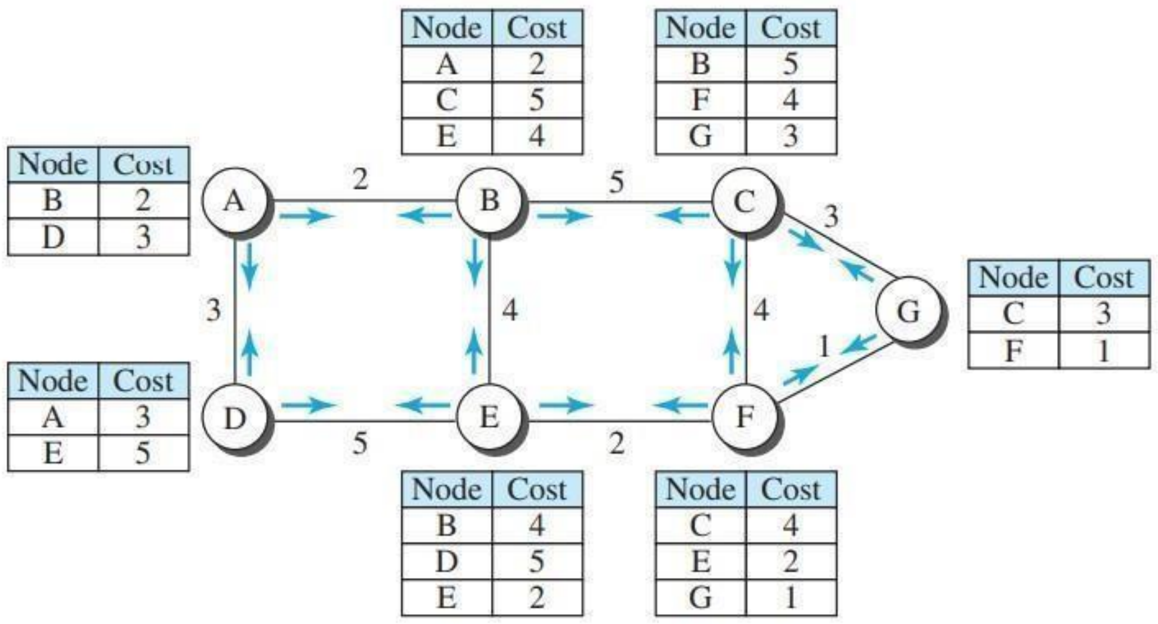
Each node can create this LSDB that contains information about the whole internet by a process called flooding. Each node can send some greeting messages to all its immediate neighbours (those nodes to which it is connected directly) to collect two pieces of information for each neighbouring node: the 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:

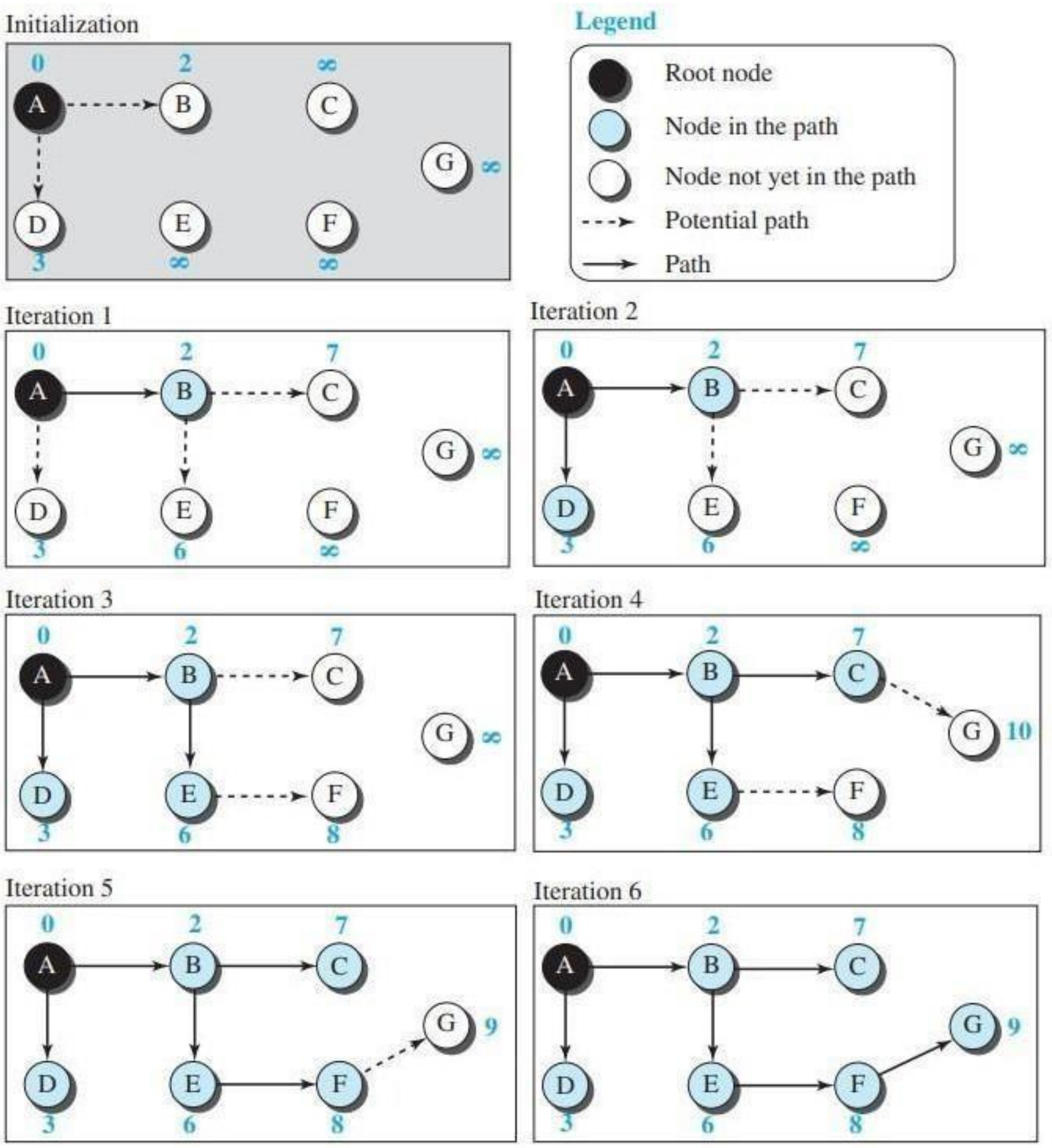
1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.

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 leastcost 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:

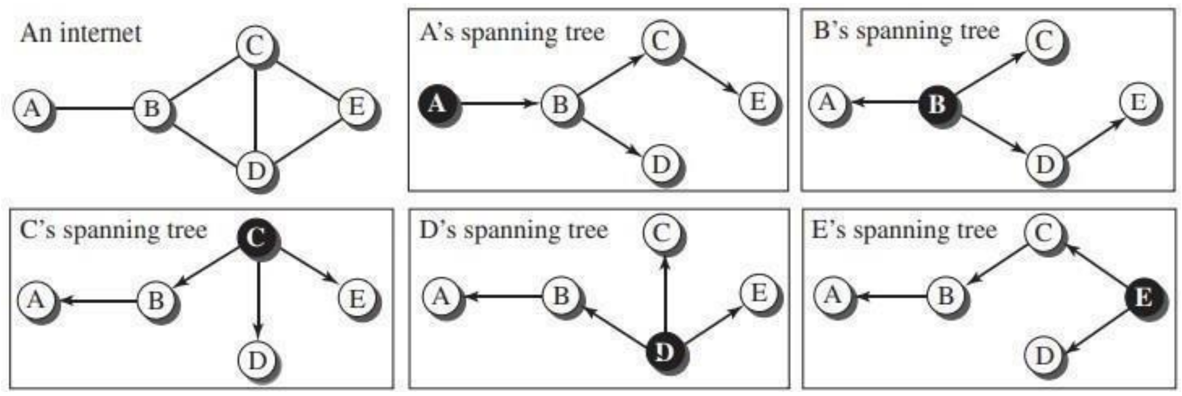
In path-vector routing, the path from a source to all destinations is also determined by the best spanning tree. The best spanning tree is not the least-cost tree; it is the tree determined by the source when it imposes its own policy.

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.

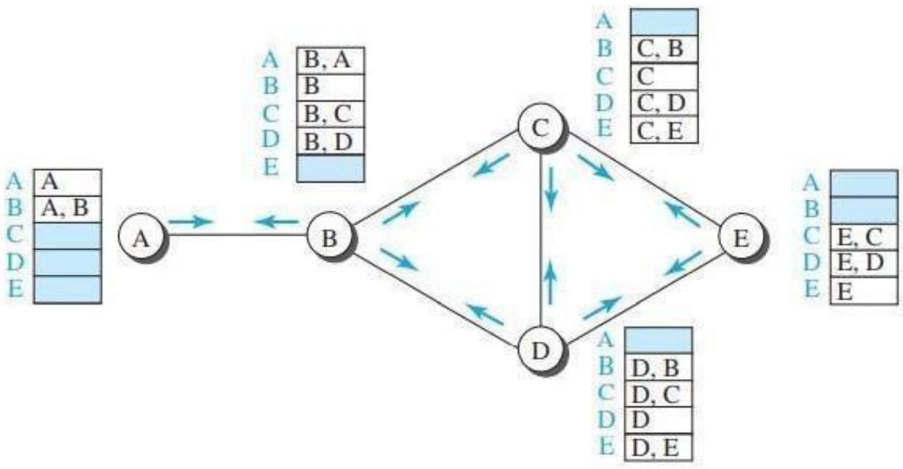
Note that we do not mean that all these tables are created simultaneously; they are created when each node is booted.

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

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

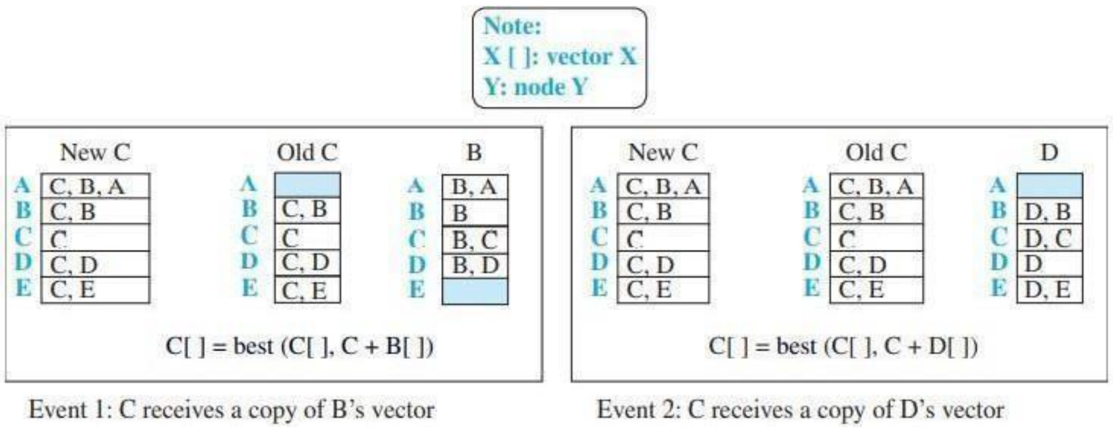
In this equation, the operator (+) means to add x to the beginning of the path. We also need to be cautious to avoid adding a node to an empty path because an empty path means one that does not exist.



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.



Note:
X [ ]: vector X
Y: node Y
New C: C, B, A; C, B; C; C, D; C, E
Old C: C, B; C; C, D; C, E
B: B, A; B; B, C; B, D
C[ ] = best (C[ ], C + B[ ])
Event 1: C receives a copy of B's vector
New C: C, B, A; C, B; C; C, D; C, E
Old C: C, B, A; C, B; C; C, D; C, E
D: D, B; D, C; D; D, E
C[ ] = best (C[ ], C + D[ ])
Event 2: C receives a copy of D's vector

UNICAST ROUTING PROTOCOLS

Hierarchical Routing in Internet

The Internet today is made of a huge number of networks and routers that connect them. Routing in the Internet cannot be done using a single protocol for two reasons:
A scalability problem and an administrative issue.

Scalability problem means that the size of the forwarding tables becomes huge, searching for a destination in a forwarding table becomes time-consuming, and updating creates a huge amount of traffic.

Hierarchical routing means considering each ISP as an autonomous system (AS).Each AS can run a routing protocol that meets its needs, but the global Internet runs a global protocol to join and connect all ASs together.The routing protocol run in each AS is referred to as intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP);The global routing protocol is referred to as inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP).

Routing Information Protocol (RIP)

The Routing Information Protocol (RIP) is one of the most widely used intradomain routing protocols based on the distance-vector routing algorithm.

Hop Count

A router in this protocol implements the distance-vector routing algorithm. First, since a router in an AS needs to know how to forward a packet to different networks(subnets) in an AS, RIP routers advertise the cost of reaching different networks instead of reaching other nodes in a theoretical graph.

The cost is defined between a router and the network in which the destination host is located.
Second, to make the implementation of the cost simpler (independent from performance factors of the routers and links, such as delay, bandwidth, and so on), the cost is defined as the number of hops, which means the number of networks (subnets)a packet needs to travel through from the source router to the final destination host.

Note that the network in which the source host is connected is not counted in this calculation because the source host does not use a forwarding table; the packet is delivered to the default router.

Figure 3.3.1 shows the concept of hop count advertised by three routers from a source host to a destination host. In RIP, the maximum cost of a path can be 15, which means 16 is considered as infinity (no connection).For this reason, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops.

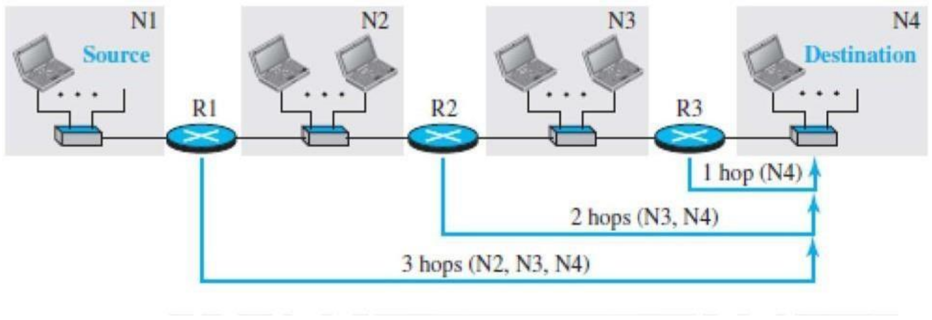
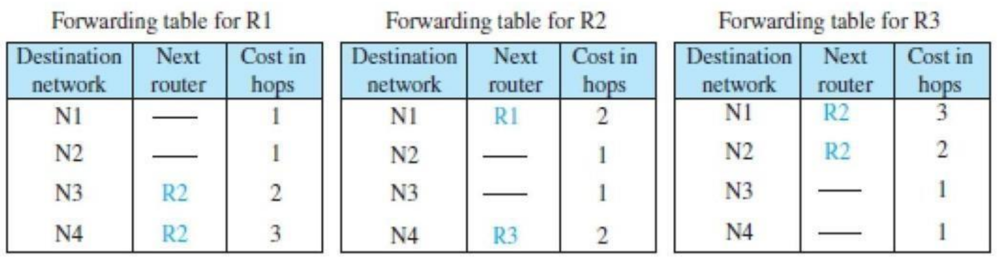


Fig3.3.1: Hop counts in RIP.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-613]

Forwarding Table

A forwarding table in RIP is a three-column table in which the first column is the address of the destination network, the second column is the address of the next router to which the packet should be forwarded, and the third column is the cost (the number of hops) to reach the destination network.Figure3.3.2 shows the three forwarding tables for the routers in Figure (above).

Note that the first and the third columns together convey the same information as does a distance vector, but the cost shows the number of hops to the destination networks.



Forwarding table for R1

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

Forwarding table for R2

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

Forwarding table for R3

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

Fig3.3.2:Forwarding tables in RIP.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-614]

For example**,** R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 isR3; R3 defines that there is no next router for this path. The tree is then R1 -R2-R3-N4.

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

RIP Implementation

RIP is implemented as a process that uses the service of UDP on the port number 520. RIP is a routing protocol to help IP route its datagrams through the AS,the RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams.

That is, RIP runs at the application layer, but creates forwarding tables for IP at the network layer.

RIP Messages

Two RIP processes, a client and a server, need to exchange messages. RIP-2 defines the format of the message, as shown in figure3.3.3.The message Entry, can be repeated as needed in a message. Each entry carries the information related to one line in the forwarding table of the router that sends the message.

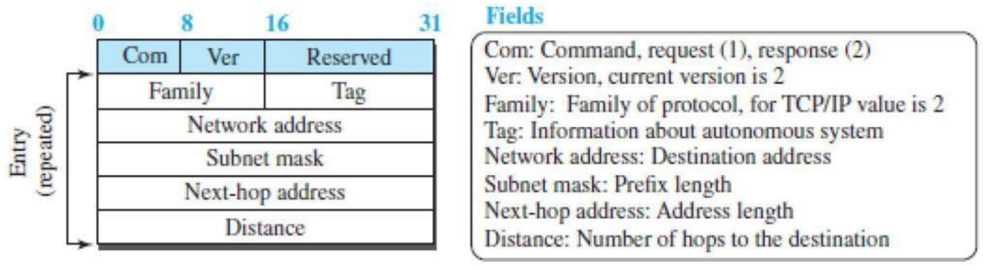

0
8
16
31
Com
Ver
Reserved
Family
Tag
Network address
Subnet mask
Next-hop address
Distance
Entry (repeated)
Fields
Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

Fig3.3.3: RIP message format.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-615]

RIP has two types of messages:

Request and response. A request message is sent by a router that has just come up or by a router that has some time-out entries.

A request message can ask about specific entries or all entries.

A response (or update)message can be either solicited or unsolicited. A solicited response message is sent only in answer to a request message. It contains information about the destination specified in the corresponding request message.

RIP Algorithm

RIP implements the same algorithm as the distance-vector routing algorithm.

- Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message.
- The receiver adds one hop to each cost and changes the next router field to the address of the sending router.
- The received router selects the old routes as the new ones except in the following three cases:

  1. If the received route does not exist in the old forwarding table, it should be added to the route.
  2. If the cost of the received route is lower than the cost of the old one, the received route should be selected as the new one.
  3. If the cost of the received route is higher than the cost of the old one, but the value of the next router is the same in both routes, the received route should be selected as the new one.

Timers in RIP

RIP uses three timers to support its operation.

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

The expiration timer governs the validity of a route. When a router receives update information for a route, the expiration timer is set to 180 seconds for that particular route. Every time a new update for the route is received, the timer is reset.

If there is a problem on an internet and no update is received within the allotted 180 seconds, the route is considered expired and the hop count of the route is set to 16, which means the destination is unreachable. Every route has its own expiration timer. The garbage collection timer is used to purge a route from the forwarding table.

The garbage collection timer is used to purge a route from the forwarding table. When the information about a route becomes invalid, the router does not immediately purge that route from its table. Instead, it continues to advertise the route with a metric value of 16. At the same time, a garbage collection timer is set to 120 seconds for that route. When the count reaches zero, the route is purged from the table.

Open Shortest Path First (OSPF)

Open Shortest Path First (**OSPF**) is an intradomain routing protocol like RIP.It is based on the link-state routing protocol.

Metric

In OSPF, like RIP, the cost of reaching a destination from the host is calculated from the source router to the destination network.

However, each link (network) can be assigned a weight based on the throughput, round-trip time, reliability, and so on

Figure 3.3.4 shows the idea of the cost from a router to the destination host network.

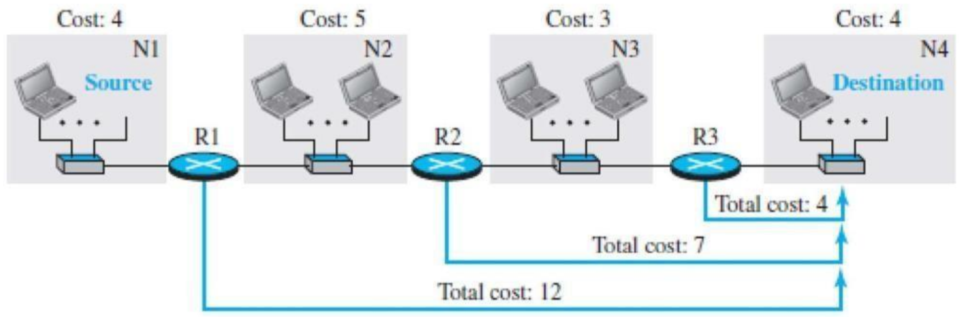
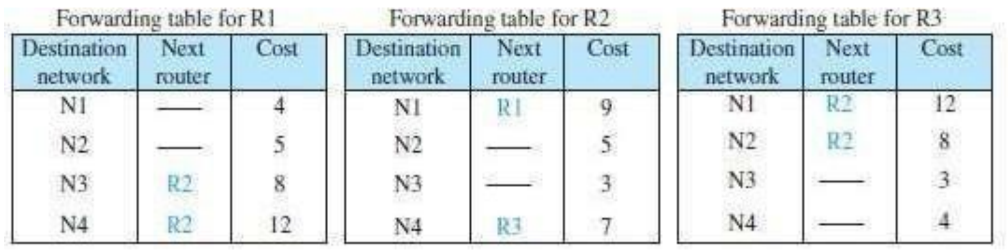


Fig3.3.4:Metric in OSPF.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-618]

Forwarding Tables

Each OSPF router can create a forwarding table as in figure 3.3.5, after finding the shortest-path tree between itself and the destination using Dijkstra's algorithm.



Forwarding table for R1

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

Forwarding table for R2

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

Forwarding table for R3

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

Fig3.3.5:Forwading table in OSPF.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-619]

Areas

OSPF was designed to handle routing in a small or large autonomous system.

The formation of shortest-path trees in OSPF requires that all routers flood the whole AS with their LSPs to create the global LSDB.This may not create a problem in a small AS, but create traffic in large AS.To prevent this, the AS needs to be divided into small sections called areas as shown in figure 3.3.6 .

Each area acts as a small independent domain for flooding .Each router in an area needs to know the information about the link states not only in its area but also in other areas.

For this reason, one of the areas in the AS is designated as the backbone area, responsible for gluing the areas together.

The routers in the backbone area are responsible for passing the information collected by each area to all other areas. In this way, a router in an area can receive all LSPs generated in other areas. For the purpose of communication, each area has an area identification.

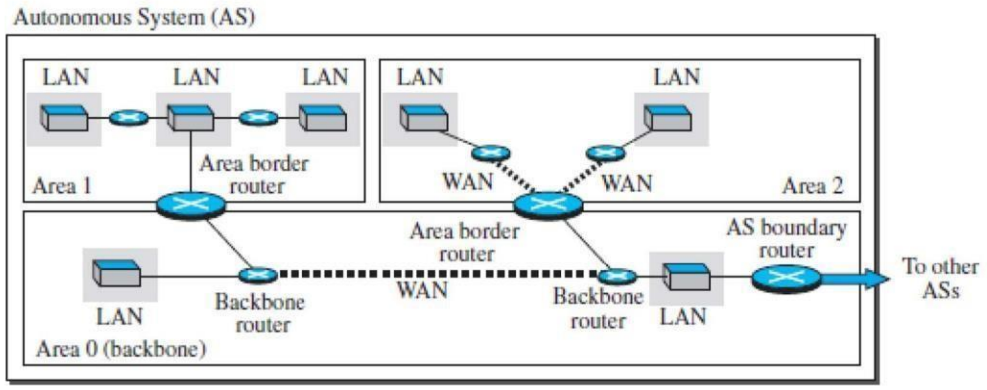


Fig3.3.6:Areas in AS.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-619]

OSPF Implementation

OSPF is implemented as a program in the network layer, using the service of the IP for propagation. An IP datagram that carries a message from OSPF sets the value of the protocol field to 89. This means that, the OSPF messages are encapsulated inside datagrams.

OSPF has two versions: version 1 and version 2.

OSPF Messages

OSPF is a very complex protocol; it has five different types of messages as shown in figure 3.3.7 .

.The hello message (type 1) is used by a router to introduce itself to the neighbors.

The database description message (type 2) is sent in response to the hello message to allow a newly joined router to acquire the full LSDB.The link state request message (type 3) is sent by a router that needs information about a specific LS.The link-state update message (type 4) is the main OSPF message used for building the LSDB. This message, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link).The link-state acknowledgment message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.

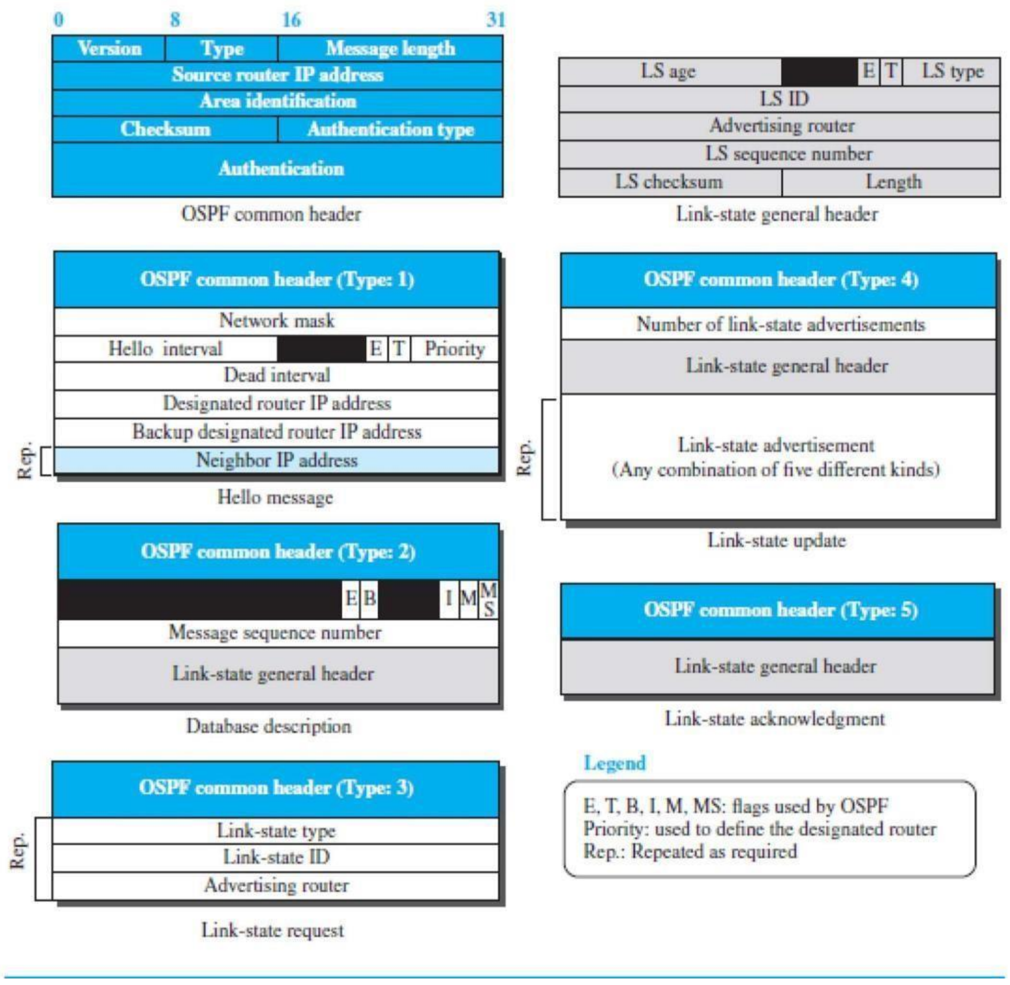

0
8
16
31
Version
Type
Message length
Source router IP address
Area identification
Checksum
Authentication type
Authentication
OSPF common header
LS age
E T
LS type
LS ID
Advertising router
LS sequence number
LS checksum
Length
Link-state general header
OSPF common header (Type: 1)
Network mask
Hello interval
E T
Priority
Dead interval
Designated router IP address
Backup designated router IP address
Neighbor IP address
Rep.
Hello message
OSPF common header (Type: 4)
Number of link-state advertisements
Link-state general header
Link-state advertisement
(Any combination of five different kinds)
Rep.
Link-state update
OSPF common header (Type: 2)
E B
I M MS
Message sequence number
Link-state general header
Database description
OSPF common header (Type: 5)
Link-state general header
Link-state acknowledgment
OSPF common header (Type: 3)
Link-state type
Link-state ID
Advertising router
Rep.
Link-state request
Legend
E, T, B, I, M, MS: flags used by OSPF
Priority: used to define the designated router
Rep.: Repeated as required

Fig3.3.7: OSPF message format.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-622]

OSPF Algorithm

OSPF implements the link-state routing algorithm .

After each router has created the shortest-path tree, the algorithm needs to use it to create the corresponding routing algorithm.The algorithm needs to be augmented to handle sending and receiving all five types of messages.

Border Gateway Protocol Version 4 (BGP4)

The Border Gateway Protocol version 4 (BGP4) is the only inter domain routing protocol used in the Internet today.

Consider an example of an internet with four autonomous systems. AS2, AS3, and AS4 are stub autonomous systems; AS1 is a transient one as shown in figure 3.3.8 .

. Here,data exchange between AS2, AS3, and AS4 should pass through AS1.

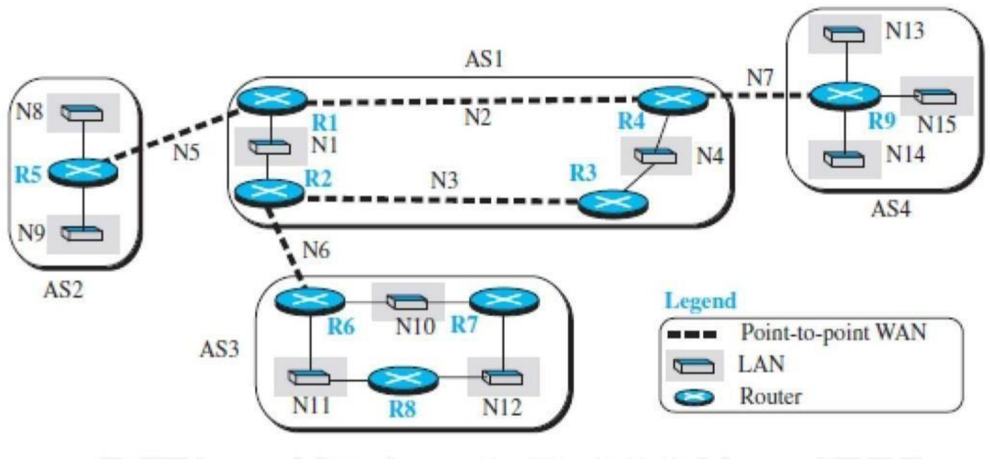


Fig3.3.8:Sample internet with four AS.
[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-623]

Each router in each AS knows how to reach a network that is in its own AS, but it does not know how to reach a network in another AS.

To enable each router to route a packet to any network in the internet, we first install a variation of BGP4, called external BGP (eBGP), on each border router (the one at the edge of each AS which is connected to a router at another AS).We then install the second variation of BGP, called internal BGP (iBGP), on all routers.

The border routers will be running three routing protocols (intradomain, eBGP, and iBGP), but other routers are running two protocols (intradomain and iBGP).

Operation of External BGP (eBGP)

BGP is a point-to-point protocol. When the software is installed on two routers, they try to create a TCP connection using the well-known port 179.

The two routers that run the BGP processes are called BGP peers or BGP speakers.The eBGP variation of BGP allows two physically connected border routers in two different ASs to form pairs of eBGP speakers and exchange messages.

The routers that we use has three pairs: R1-R5, R2-R6, and R4-R9. The connection between these pairs is established over three physical WANs (N5,N6, and N7). There is a need for a logical TCP connection to be created over the physical connection to make the exchange of information possible. Each logical connection in BGP is referred to as a session. This means that we need three sessions, as shown in Figure 3.3.9.

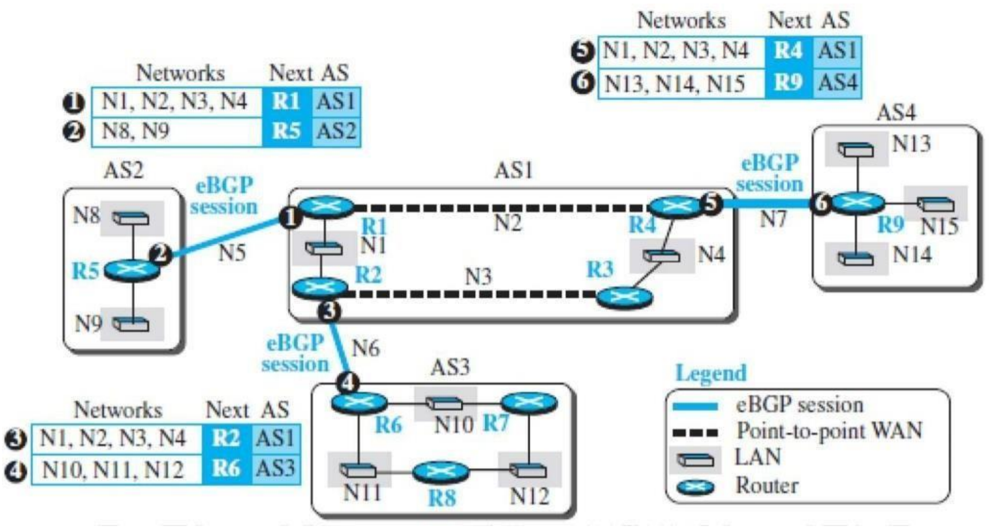


Fig3.3.9: EBGP operation.

[Source : "Data Communications and Networking" by Behrouz A. Forouzan,Page-624]

The circled number defines the sending router in each case.

For example, message number 1 is sent by router R1 and tells router R5 that N1, N2, N3,and N4 can be reached through router R1 (R1 gets this information from the corresponding intradomain forwarding table).

Router R5 can now add these pieces of information at the end of its forwarding table. When R5 receives any packet destined for these four networks, it can use its forwarding table and find that the next router is R1.

Messages

BGP four types of messages for communication between the BGP speakers across the ASs and inside an AS:

Four messages are open, update, keep alive, and notification .

All BGP packets share the same common header.

Open Message. To create a neighborhood relationship, a router running BGP opens a TCP connection with a neighbor and sends an open message.

Update Message.

The update message is used by a router to withdraw destinations that have been advertised previously, to announce a route to a new destination, or both.

Note that BGP can withdraw several destinations that were advertised before, but it can only advertise one new destination in a single update message.

Keep alive Message. The BGP peers that are running exchange keep alive messages regularly (before their hold time expires) to tell each other that they are alive.

Notification. A notification message is sent by a router whenever an error condition is detected or a router wants to close the session.

Performance

BGP performance can be compared with RIP. BGP speakers exchange a lot of messages to create forwarding tables, but BGP is free from loops and count-to-infinity.

THE IPv6 PROTOCOL

1.1 THE IPv6 PROTOCOL

Changes from IPv4 to IPv6 (Advantages of IPv6)

1) Header Format

- ➢ IPv6 uses a new header format.
- ➢ Options are
  - → separated from the base-header and
  - → inserted between the base-header and the data.
- ➢ This speeds up the routing process (because most of the options do not need to be checkedby routers).

2) New Options

- ➢ IPv6 has new options to allow for additional functionalities.

3) Extension

- ➢ IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.

4) Resource Allocation

- ➢ In IPv6,
  - → type-of-service (TOS) field has been removed
  - → two new fields: 1) traffic class and 2) flow label, are added to enable the source to request special handling of the packet.
- ➢ This mechanism can be used to support real-time audio and video.

5) Security

- ➢ The encryption option provides confidentiality of the packet.
- ➢ The authentication option provides integrity of the packet.

1.1.1 Packet Format

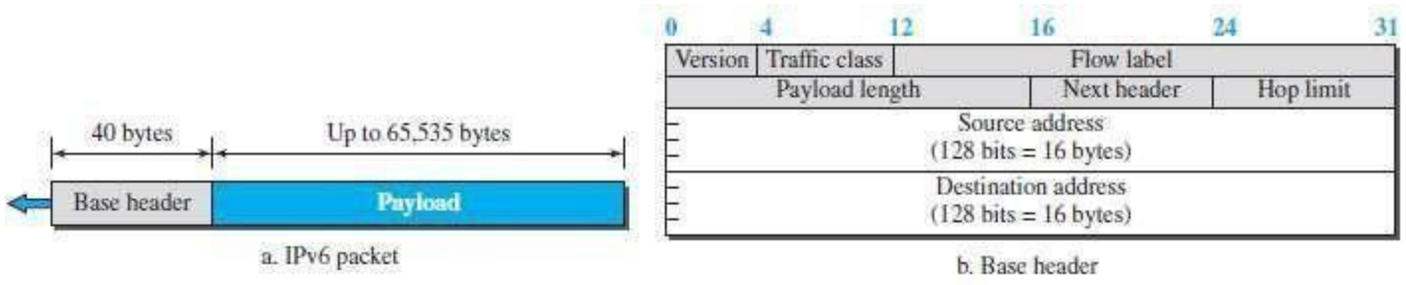


Figure 22.6 *IPv6 datagram*

- IP header contains following fields (Figure 22.6):

Version

- ➢ This specifies version number of protocol. For IPv6, version=6.

Traffic Class

- ➢ This field is used to distinguish different payloads with different delivery requirements.
  (Traffic class replaces the type-of-service field in IPv4).

Flow Label

- This field is designed to provide special handling for a particular flow of data.

Payload Length

- This indicates length of data (excluding header). Maximum length=65535 bytes.
- The length of the base-header is fixed (40 bytes); only the length of the payload needs to be defined.

Next Header

- This identifies type of extension header that follows the basic header.

Hop Limit

- This specifies number of hops the packet can travel before being dropped by a router.(Hop limit serves the same purpose as the TTL field in IPv4).

Source and Destination Addresses

- These identify source host and destination host respectively.

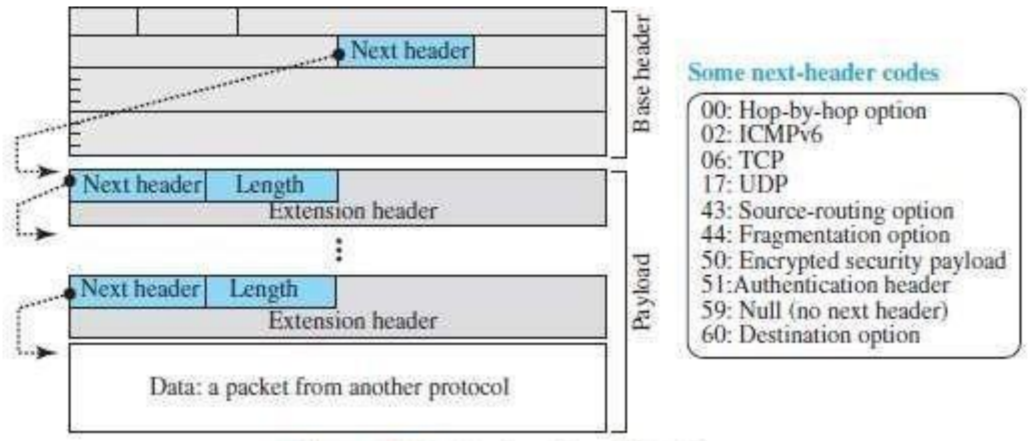


Figure 22.7 *Payload in an IPv6 datagram*

Payload

- The payload contains zero or more extension headers (options) followed by the data fromother protocols (UDP, TCP, and so on).
- The payload can have many extension headers as required by the situation.
- Each extension header has 2 mandatory fields (Figure 22.7):

  Next header and Length

- Two mandatory fields are followed by information related to the particular option.

Concept of Flow & Priority in IPv6

- To a router, a flow is a sequence of packets that share the same characteristics such as
  - → traveling the same path
  - → using the same resources or
  - → having the same kind of security
- A router that supports the handling of flow labels has a flow label table.
- The table has an entry for each active flow label.
  Each entry defines the services required by the corresponding flow label.
- When a router receives a packet, the router consults its flow label table.
- Then, the router provides the packet with the services mentioned in the entry.
- A flow label can be used to support the transmission of real-time audio/video.
- Real-time audio/video requires resources such as
  - → high bandwidth
  - → large buffers or
  - → long processing time
- Resource reservation guarantees that real-time data will not be delayed due to a lack of resources.

Fragmentation & Reassembly

- Fragmentation of the packet is done only by the source, but not by the routers.The reassembling is done by the destination.
- At routers, the fragmentation is not allowed to speed up the processing in the router.
- Normally, the fragmentation of a packet in a router needs a lot of processing. This is because
  1) The packets need to be fragmented.
  2) All fields related to the fragmentation need to be recalculated.
- The source will
  - → check the size of the packet and
  - → make the decision to fragment the packet or not.
- If packet-size is greater than the MTU of the network, the router will drop the packet.
- Then, the router sends an error message to inform the source.

1.1.2 <u>Extension Header</u>

- An IP packet is made of
  - → base-header &
  - → some extension headers.
- Length of base header = 40 bytes.
- To support extra functionalities, extension headers can be placed b/w base header and payload.
- Extension headers act like options in IPv4.
- Six types of extension headers (Figure 22.8):

1.1.2.1 Hop-by-hop option 2) Source routing 3) Fragmentation
4) Authentication 5) Encrypted security payload 5) Destination option.

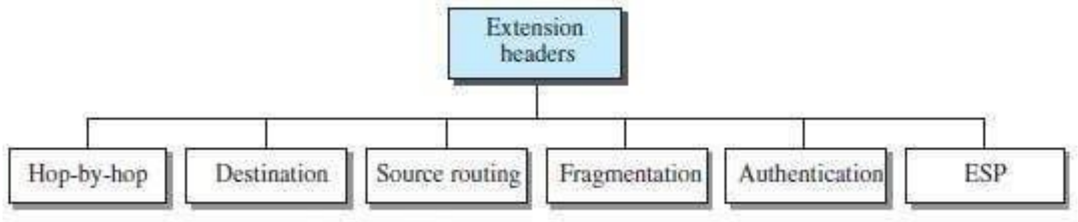


Figure 22.8 *Extension header types*

1) Hop-by-Hop Option

- This option is used when the source needs to pass information to all routers visited by the datagram.
- Three options are defined: i) Pad1, ii) PadN, and iii) Jumbo payload.

i) Pad1

- ➢ This option is designed for alignment purposes.
- ➢ Some options need to start at a specific bit of the 32-bit word.
- ➢ Pad1 is added, if one byte is needed for alignment.

ii) PadN

- ➢ PadN is similar in concept to Pad1.
- ➢ The difference is that PadN is used when 2 or more bytes are needed for alignment.

iii) Jumbo Payload

- ➢ This option is used when larger packet has to be sent. (> 65,535 bytes)
- ➢ Large packets are referred to as jumbo packets.
- ➢ Maximum length of payload = 65,535 bytes.

2) Destination Option

- This option is used when the source needs to pass information to the destination only.
- Intermediate routers are not allowed to access this information.
- Two options are defined: i) Pad1 & ii) PadN

3) Source Routing

- This option combines the concepts of
  - → strict source routing and
  - → loose source routing.

4) Fragmentation

- In IPv6, only the original source can fragment.
- A source must use a "Path MTU Discovery technique" to find the smallest MTU along the path from the source to the destination.

- Minimum size of MTU = 1280 bytes. This value is required for each network connected to the Internet.
- If a source does not use a Path MTU Discovery technique, the source fragments the datagram to a size of 1280 bytes.

5) Authentication

- This option has a dual purpose:

  i) Validates the message sender: This is needed so the receiver can be sure that a message isfrom the genuine sender and not from an attacker.

  ii) Ensures the integrity of data: This is needed to check that the data is not altered in transitionby some attacker.

6) Encrypted Security Payload (ESP)

- This option provides confidentiality and guards against attacker.

5.9.3.1 Comparison of Options between IPv4 and IPv6

1) The no-operation and end-of-option options in IPv4 are replaced by Pad1 and PadN optionsin IPv6.
2) The record route option is not implemented in IPv6 because it was not used.
3) The timestamp option is not implemented because it was not used.
4) The source route option is called the source route extension header in IPv6.
5) The fragmentation fields in the base-header section of IPv4 have moved to the fragmentation extension header in IPv6.
6) The authentication extension header is new in IPv6.
7) The encrypted security payload extension header is new in IPv6.