UNIT III
NETWORK LAYER

Network Layer Services – Packet switching – Performance – IPV4 Addresses – Forwarding of IP Packets - Network Layer Protocols: IP, ICMP v4 – Unicast Routing Algorithms – Protocols – Multicasting Basics – IPV6 Addressing – IPV6 Protocol.

NETWORK LAYER SERVICES

Packetizing

The first duty of the network layer is definitely 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. In other words, one duty of the network layer is to carry a payload from the source to he destination without changing it or using it. The network layer is doing the service of a carrier such as the postal office, which is responsible for delivery of packages from a sender to a receiver without changing or using the contents.

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 (as discussed later) and delivers the packet to the data-link layer.

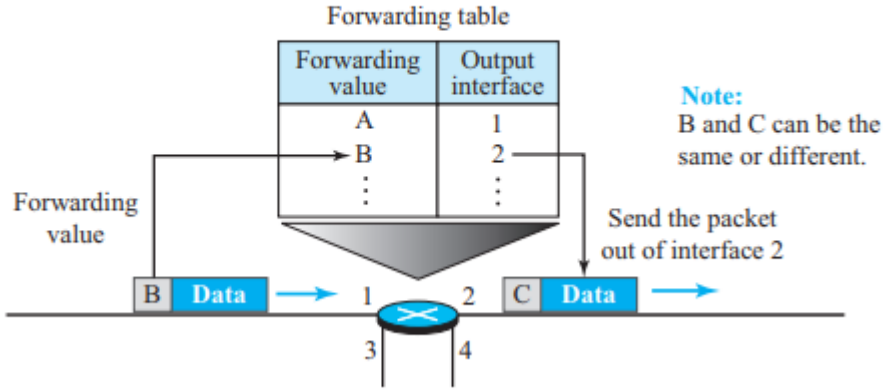
The destination host receives the network-layer packet from its data-link layer, decapsulates the packet, and delivers the payload to the corresponding upper-layer protocol. If the packet is fragmented at the source or at routers along the path, the network layer is responsible for waiting until all fragments arrive, reassembling them, and delivering them to the upper-layer protocol.

Routing

The network layer is responsible for routing the packet from its source to the destination. This means that there is more than one route from the source to the destination. 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.

Forwarding

Figure 18.2 *Forwarding process*



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).

Other Services

Error Control: The designers of the network layer, however, 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.

Congestion Control: 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 (in particular 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. However, 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, however, 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

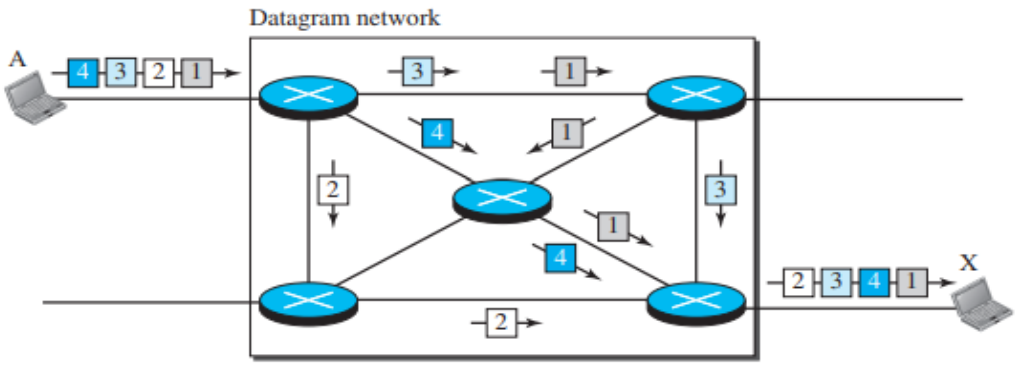
In data communications, we need to send messages from one end system to another. If the message is going to pass through a packet-switched network, it needs to be divided into packets of fixed or variable size. The size of the packet is determined by the network and the governing protocol. In packet switching, there is no resource allocation for a packet. This means that there is no reserved bandwidth on the links, and there is no scheduled processing time for each packet. Resources are allocated on demand. The allocation is done on a first come, first-served basis. When a switch receives a packet, no matter what the source or destination is, the packet must wait if there are other packets being processed. As with other systems in our daily life, this lack of reservation may create delay. For example, if we do not have a reservation at a restaurant, we might have to wait.

We can have two types of packet-switched networks: **datagram networks and virtual circuit networks.**

Datagram Networks In a datagram network, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as datagrams. Datagram switching is normally done at the network layer. We briefly discuss datagram networks here as a comparison with circuit-switched and virtual-circuitswitched networks.

Figure shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers.

Figure 8.7 *A datagram network with four switches (routers)*

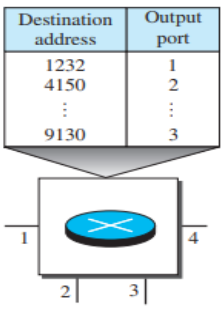


In this example, all four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination. This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X. This approach can cause the datagrams of a transmission to arrive at their destination out of order with different delays between the packets. Packets may also be lost or dropped because of a lack of resources. In most protocols, it is the responsibility of an upper-layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application. The datagram networks are sometimes referred to as connectionless networks. The term connectionless here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

The destination addresses and the corresponding forwarding output ports are recorded in the tables.

Figure 8.8 *Routing table in a datagram network*

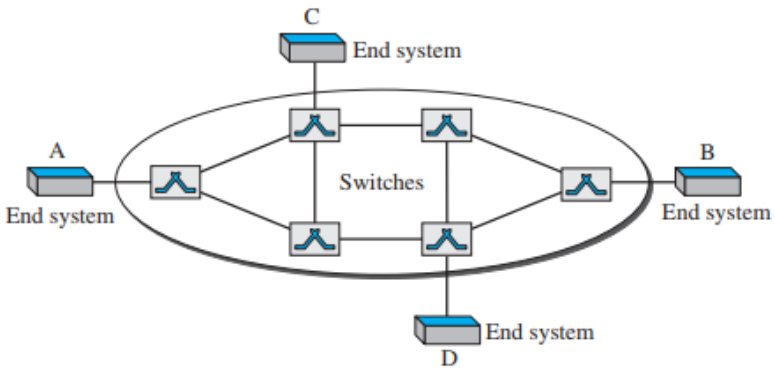


Virtual-Circuit Networks A virtual-circuit network is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what the next switch should be and the channel on which the packet is being carried), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual-circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data-link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future.

Below figure is an example of a virtual-circuit network. The network has switches that allow traffic from sources to destinations. A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

Figure 8.10 *Virtual-circuit network*



Addressing In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

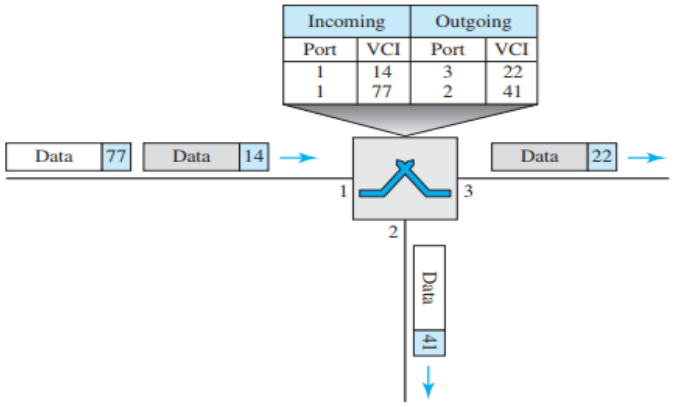
Global Addressing A source or a destination needs to have a global address—an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier The identifier that is actually used for data transfer is called the virtual-circuit identifier (VCI) or the label. A VCI, unlike a global address, is a small number that

has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCI.

Data-Transfer Phase To transfer a frame from a source to its destination, all switches need to have a table entry for this virtual circuit. The table, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits.

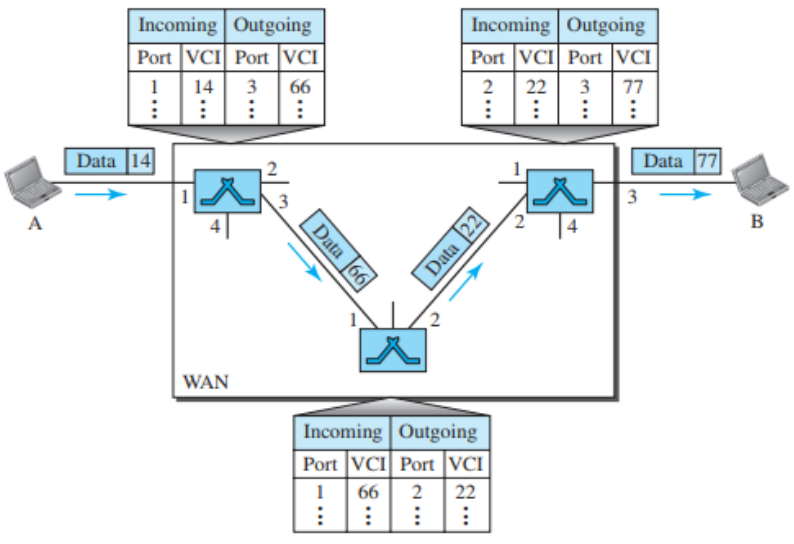
Figure 8.12 *Switch and tables in a virtual-circuit network*



Above figure shows the frame arriving at port 1 with a VCI of 14. When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14. When it is found, the switch knows to change the VCI to 22 and send out the frame from port 3.

Below figure shows how a frame from source A reaches destination B and how its VCI changes during the trip. Each switch changes the VCI and routes the frame. The data-transfer phase is active until the source sends all its frames to the destination. The procedure at the switch is the same for each frame of a message. The process creates a virtual circuit, not a real circuit, between the source and destination.

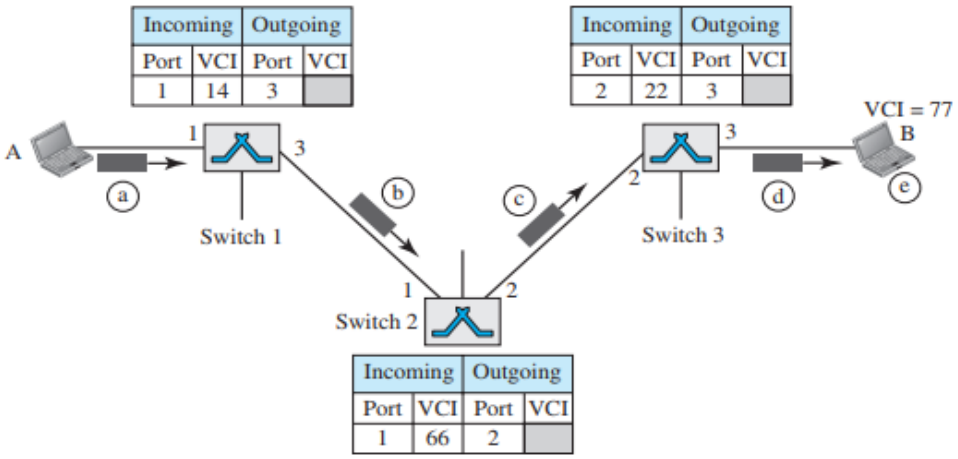
Figure 8.13 *Source-to-destination data transfer in a virtual-circuit network*



Setup Phase In the setup phase, a switch creates an entry for a virtual circuit. For example, suppose source A needs to create a virtual circuit to B. Two steps are required: the setup request and the acknowledgment.

Setup Request A setup request frame is sent from the source to the destination.

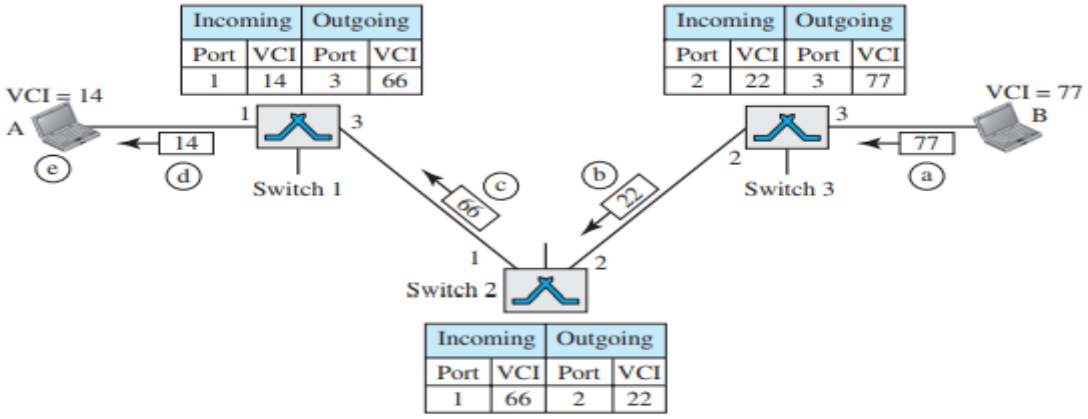
Figure 8.14 *Setup request in a virtual-circuit network*



a. Source A sends a setup frame to switch 1.
b. Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; it has a routing table which is different from the switching table. For the moment, assume that it knows the output port. The switch creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the outgoing port (3). It does not yet know the outgoing VCI, which will be found during the acknowledgment step. The switch then forwards the frame through port 3 to switch 2.
c. Switch 2 receives the setup request frame. The same events happen here as at switch 1; three columns of the table are completed: in this case, incoming port (1), incoming VCI (66), and outgoing port (2).
d. Switch 3 receives the setup request frame. Again, three columns are completed: incoming port (2), incoming VCI (22), and outgoing port (3).
e. Destination B receives the setup frame, and if it is ready to receive frames from A, it assigns a VCI to the incoming frames that come from A, in this case 77. This VCI lets the destination know that the frames come from A, and not other sources.

Acknowledgment A special frame, called the acknowledgment frame, completes the entries in the switching tables.

Figure 8.15 *Setup acknowledgment in a virtual-circuit network*



a. The destination sends an acknowledgment to switch 3. The acknowledgment carries the global source and destination addresses so the switch knows which entry in the table is to be completed. The frame also carries VCI 77, chosen by the destination as the incoming VCI for frames from A. Switch 3 uses this VCI to complete the outgoing VCI column for this entry. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.
b. Switch 3 sends an acknowledgment to switch 2 that contains its incoming VCI in the table, chosen in the previous step. Switch 2 uses this as the outgoing VCI in the table.
c. Switch 2 sends an acknowledgment to switch 1 that contains its incoming VCI in the table, chosen in the previous step. Switch 1 uses this as the outgoing VCI in the table.
d. Finally switch 1 sends an acknowledgment to source A that contains its incoming VCI in the table, chosen in the previous step.
e. The source uses this as the outgoing VCI for the data frames to be sent to destination B.

Teardown Phase In this phase, source A, after sending all frames to B, sends a special frame called a teardown request. Destination B responds with a teardown confirmation frame. All switches delete the corresponding entry from their tables.

PERFORMANCE

The performance of a network can be measured in terms of **delay, throughput, and packet loss.**

Delay

All of us expect instantaneous response from a network, but a packet, from its source to its destination, encounters delays. The delays in a network can be divided into four types: **transmission delay, propagation delay, processing delay, and queuing delay**.

Transmission Delay:

A source host or a router cannot send a packet instantaneously. A sender needs to put the bits in a packet on the line one by one. If the first bit of the packet is put on the line at time t_1 and the last bit is put on the line at time t_2, transmission delay of the packet is $(t_2 - t_1)$.

Definitely, the transmission delay is longer for a longer packet and shorter if the sender can transmit faster.

$$\textbf{Delay}_{tr} = \textbf{(Packet length) / (Transmission rate).}$$

Propagation Delay:

Propagation delay is the time it takes for a bit to travel from point A to point B in the transmission media.

$$\textbf{Delay}_{pg} = \textbf{(Distance) / (Propagation speed).}$$

Processing Delay:

The processing delay is the time required for a router or a destination host to receive a packet from its input port, remove the header, perform an error detection procedure, and deliver the packet to the output port (in the case of a router) or deliver the packet to the upper-layer protocol (in the case of the destination host).

$$\textbf{Delay}_{pr} = \textbf{Time required to process a packet in a router or a destination host.}$$

Queuing Delay:

Queuing delay can normally happen in a router. A router has an input queue connected to each of its input ports to store packets waiting to be processed; the router also has an output queue connected to each of its output ports to store packets waiting to be transmitted. The queuing delay for a packet in a router is measured as the time a packet waits in the input queue and output queue of a router.

$$\textbf{Delay}_{qu} = \textbf{The time a packet waits in input and output queues in a router}$$

Total Delay:

Assuming equal delays for the sender, routers, and receiver, the total delay (source-todestination delay) a packet encounters can be calculated if we know the number of routers, n, in the whole path.

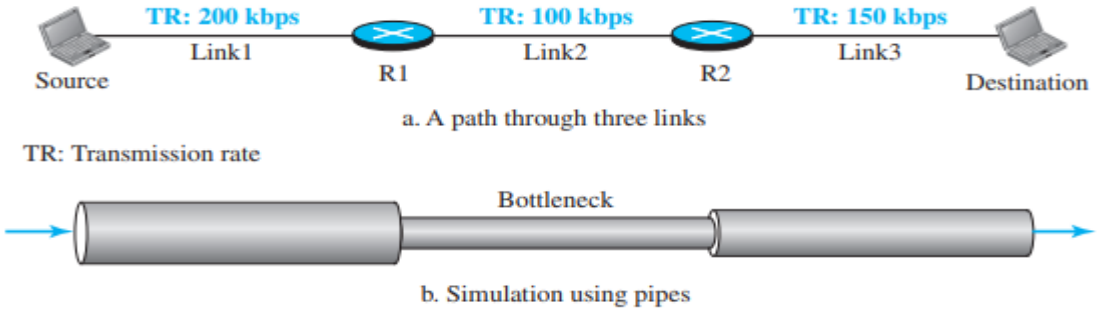
$$\textbf{Total delay} = \textbf{(n + 1) (Delay}_{tr} \textbf{ + Delay}_{pg} \textbf{ + Delay}_{pr}\textbf{) + (n) (Delay}_{qu}\textbf{)}$$

Note that if we have n routers, we have (n + 1) links. Therefore, we have (n + 1) transmission delays related to n routers and the source, (n + 1) propagation delays related to (n + 1) links, (n + 1) processing delays related to n routers and the destination, and only n queuing delays related to n routers.

Throughput

Throughput at any point in a network is defined as the number of bits passing through the point in a second, which is actually the transmission rate of data at that point.

Figure 18.10 *Throughput in a path with three links in a series*



Assume that we have three links, each with a different transmission rate, then

$$\textbf{Throughput} = \textbf{minimum } \{TR_1, TR_2, \ldots TR_n\}.$$

Packet Loss

Another issue that severely affects the performance of communication is the number of packets lost during transmission. When a router receives a packet while processing another packet, the received packet needs to be stored in the input buffer waiting for its turn. A router, however, has an input buffer with a limited size. A time may come when the buffer is full and the next packet needs to be dropped. The effect of packet loss on the Internet network layer is that the packet needs to be resent, which in turn may create overflow and cause more packet loss.

Congestion Control

Congestion control is a mechanism for improving performance. In general, we can divide congestion control mechanisms into two broad categories: **open-loop congestion control (prevention) and closed-loop congestion control (removal).**

Open-Loop Congestion Control: In open-loop congestion control, policies are applied to prevent congestion before it happens. The list of policies that can prevent congestion are:

- Retransmission Policy
- Window Policy
- Acknowledgment Policy
- Discarding Policy
- Admission Policy

Closed-Loop Congestion Control: Closed-loop congestion control mechanisms try to alleviate congestion after it happens. Several mechanisms have been used by different protocols.

- Backpressure
- Choke Packet

IPv4 ADDRESSES

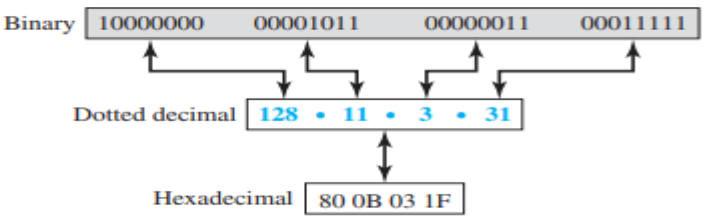
The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the **Internet address or IP address.** An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet.

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

Figure 18.16 *Three different notations in IPv4 addressing*

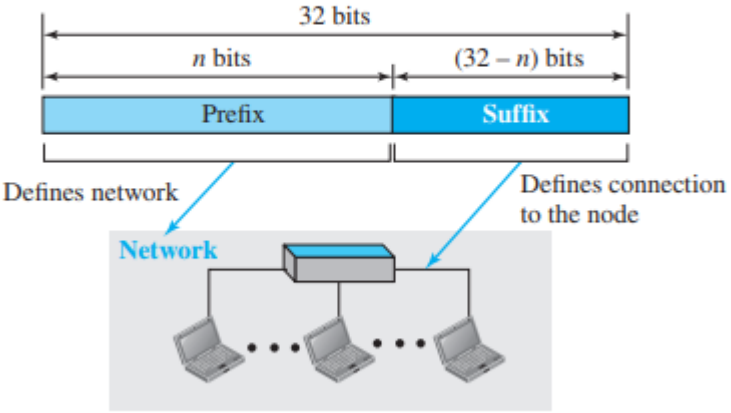


There are three common notations to show an IPv4 address: **binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16).**

Hierarchy in Addressing

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). The prefix length is n bits and the suffix length is (32 − n) bits.

Hierarchy in addressing

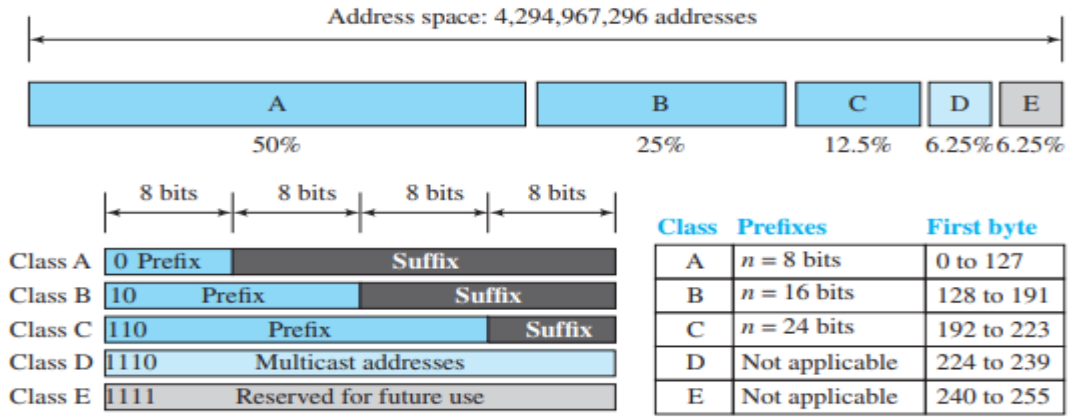


A prefix can be fixed length or variable length. The network identifier in the IPv4 was first designed as a fixed-length prefix. This scheme, which is now obsolete, is referred to as classful addressing. The new scheme, which is referred to as classless addressing, uses a variable-length network prefix.

Classful Addressing

The whole address space was divided into **five classes (class A, B, C, D, and E).** This scheme is referred to as classful addressing.

Figure 18.18 *Occupation of the address space in classful addressing*



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

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 $\mathbf{2^7 = 128}$ 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 $\mathbf{2^{14} = 16{,}384}$ 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 $\mathbf{2^{21} = 2{,}097{,}152}$ networks in the world that can have a class C address.

Class D is not divided into prefix and suffix. It is used for **multicast addresses**. All addresses that start with 1111 in binary belong to class E. As in Class D, **Class E** is not divided into prefix and suffix and is used as **reserve**.

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 nsub = 10. At the same time, if all of 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.

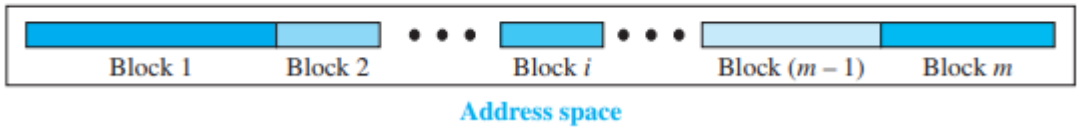
Classless Addressing

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, however, requires that the length of IP addresses also be increased, which means the format of the IP packets needs to be changed. The short-term solution still uses IPv4 addresses, but it is called classless addressing.

There was another motivation for classless addressing. During the 1990s, Internet Service Providers (ISPs) came into prominence. An ISP is an organization that provides Internet access for individuals, small businesses, and midsize organizations that do not want to create an Internet site and become involved in providing Internet services (such as electronic mail) for their employees.

In 1996, the Internet authorities announced a new architecture called classless addressing. In classless addressing, variable-length blocks are used that belong to no classes. We can have a block of 1 address, 2 addresses, 4 addresses, 128 addresses, and so on. 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 18.19 *Variable-length blocks in classless addressing*



Address Mask

Another way to find the first and last addresses in the block is to use the address mask. The address mask is a 32-bit number in which the n leftmost bits are set to 1s and the rest of the bits (32 − n) are set to 0s.

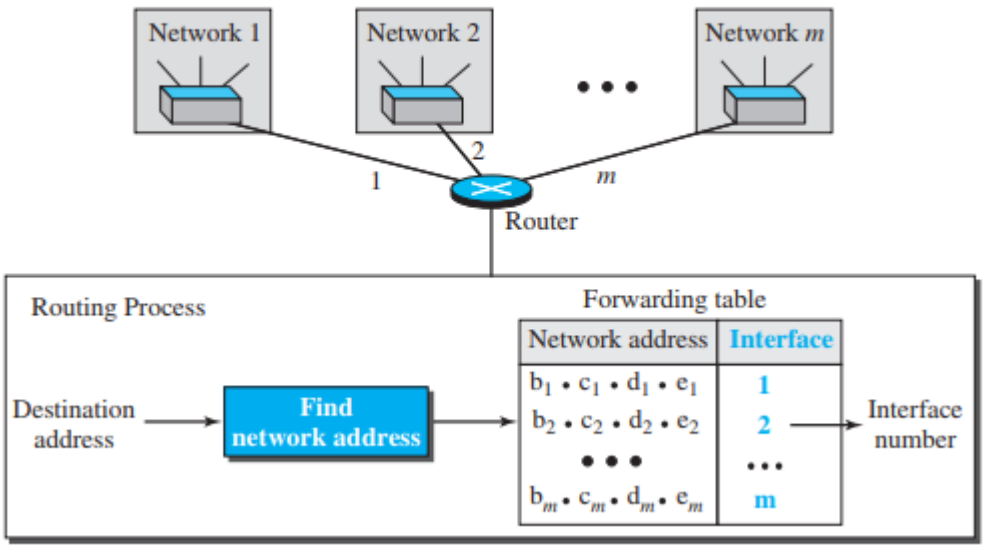
A computer can easily find the address mask because it is the complement of $(2^{32-n} - 1)$. The reason for defining a mask in this way is that it can be used by a computer program to extract the information in a block, using the three bit-wise operations NOT, AND, and OR.

1. The number of addresses in the block N = NOT (mask) + 1.
2. The first address in the block = (Any address in the block) AND (mask).
3. The last address in the block = (Any address in the block) OR [(NOT (mask)].

Network Address

The first address, the network address, is particularly important because it is used in routing a packet to its destination network. For the moment, let us assume that an internet is made of m networks and a router with m interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet should be sent: from which interface the packet should be sent out.

Figure 18.22 *Network address*



Special Addresses

Before finishing the topic of addresses in IPv4, we need to mention five special addresses that are used for special purposes:

- **this-host address:** 0.0.0.0/32
- **limited-broadcast address:** 255.255.255.255/32
- **loopback address:** 127.0.0.0/8
- **private addresses:** Four blocks are assigned as private addresses: 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, and 169.254.0.0/16
- **multicast addresses:** The block 224.0.0.0/4 is reserved for multicast addresses.

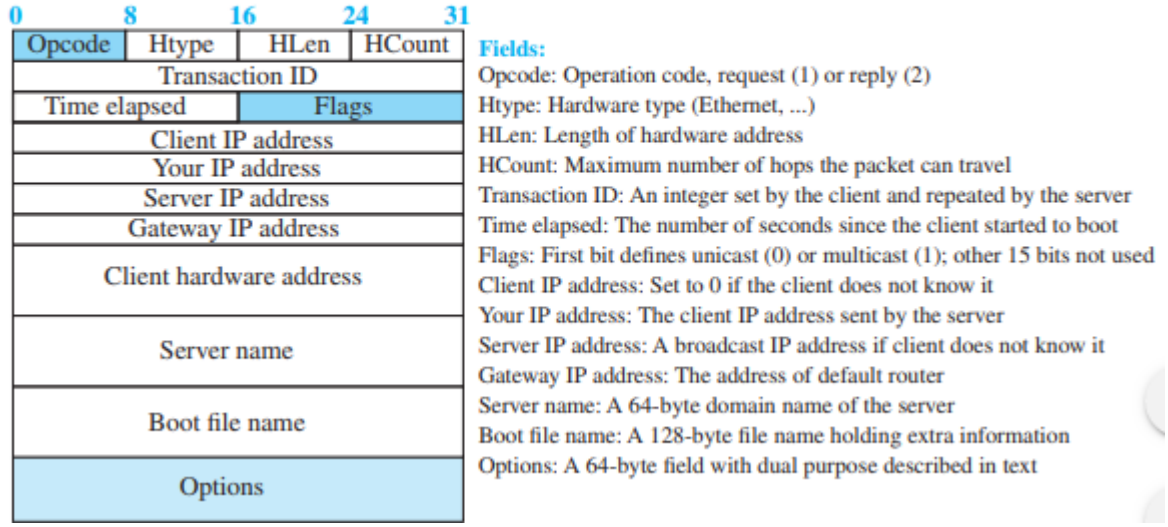
Dynamic Host Configuration Protocol (DHCP)

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. 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.

DHCP Message Format

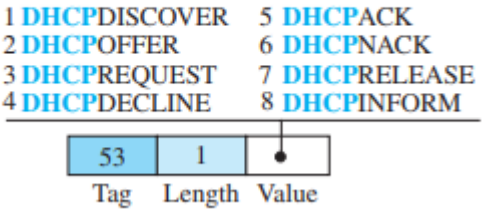
DHCP is a client-server protocol in which the client sends a request message and the server returns a response message.

Figure 18.25 *DHCP message format*



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.

Option format



FORWARDING OF IP PACKETS

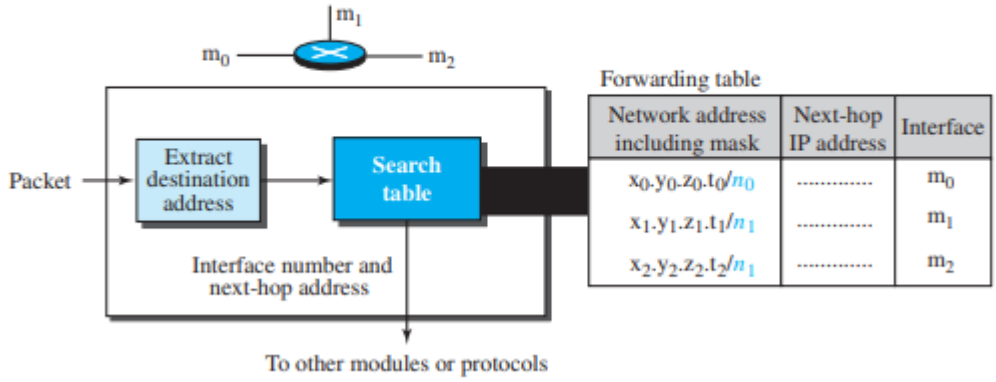
Forwarding means to place the packet in its route to its destination. Since the Internet today is made of a combination of links (networks), forwarding means to deliver the packet to the next hop (which can be the final destination or the intermediate connecting device). Although the IP protocol was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

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

Forwarding Based on Destination Address:

When a host has a packet to send or when a router has received a packet to be forwarded, it looks at this table to find the next hop to deliver the packet to. In classless addressing, the whole address space is one entity; there are no classes. This means that forwarding requires one row of information for each block involved. The table needs to be searched based on the network address (first address in the block). Unfortunately, the destination address in the packet gives no clue about the network address. To solve the problem, we need to include the mask (/n) in the table. In other words, a classless forwarding table needs to include four pieces of information: the mask, the network address, the interface number, and the IP address of the next router (needed to find the link-layer address of the next hop.

Figure 18.32 *Simplified forwarding module in classless address*

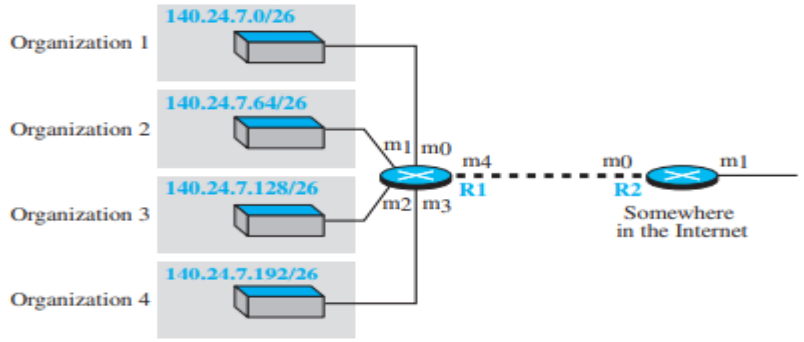


The job of the forwarding module is to search the table, row by row. In each row, the n leftmost bits of the destination address (prefix) are kept and the rest of the bits (suffix) are set to 0s. If the resulting address (which we call the network address), matches with the address in the first column,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

Address Aggregation

When we use classful addressing, there is only one entry in the forwarding table for each site outside the organization. The entry defines the site even if that site is subnetted. When a packet arrives at the router, the router checks the corresponding entry and forwards the packet accordingly. When we use classless addressing, it is likely that the number of forwarding table entries will increase. This is because the intent of classless addressing is to divide up the whole address space into manageable blocks. The increased size of the table results in an increase in the amount of time needed to search the table. To alleviate the problem, the idea of address aggregation was designed.

Figure 18.34 *Address aggregation*



Forwarding table for R1

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

Forwarding table for R2

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

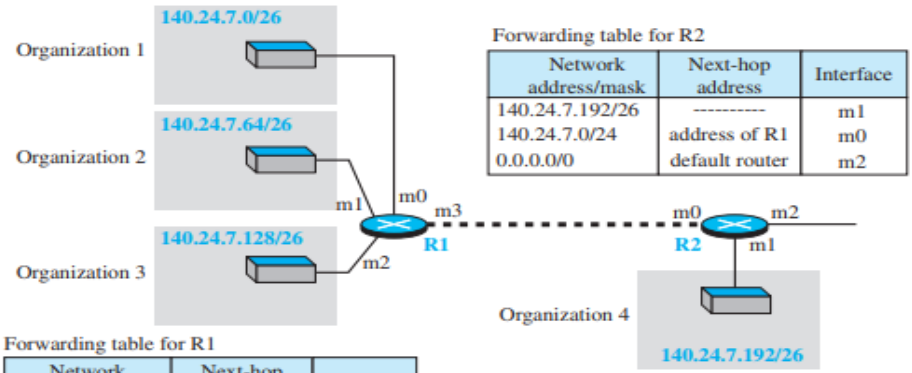
R1 is connected to networks of four organizations that each use 64 addresses. R2 is somewhere far from R1. R1 has a longer forwarding table because each packet must be correctly routed to the appropriate organization. R2, on the other hand, can have a very small forwarding table. For R2, any packet with destination 140.24.7.0 to 140.24.7.255 is sent out from interface m0 regardless of the organization number. This is called address aggregation because the blocks of addresses for four organizations are aggregated into one larger block. R2 would have a longer forwarding table if each organization had addresses that could not be aggregated into one block.

Longest Mask Matching

What happens if one of the organizations in the previous figure is not geographically close to the other three?

The answer is yes, because routing in classless addressing uses another principle, longest mask matching. This principle states that the forwarding table is sorted from the longest mask to the shortest mask.

Figure 18.35 *Longest mask matching*



Forwarding table for R2

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

Forwarding table for R1

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

Suppose a packet arrives at router R2 for organization 4 with destination address 140.24.7.200. The first mask at router R2 is applied, which gives the network address 140.24.7.192. The packet is routed correctly from interface m1 and reaches organization 4. If, however, the forwarding table was not stored with the longest prefix first, applying the /24 mask would result in the incorrect routing of the packet to router R1.

Hierarchical Routing

To solve the problem of gigantic forwarding tables, we can create a sense of hierarchy in the forwarding tables. The Internet today has a sense of hierarchy. We said that the Internet is divided into backbone and national ISPs. National ISPs are divided into regional ISPs, and regional ISPs are divided into local ISPs. If the forwarding table has a sense of hierarchy like the Internet architecture, the forwarding table can decrease in size.

Geographical Routing

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

Forwarding Table Search Algorithms

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

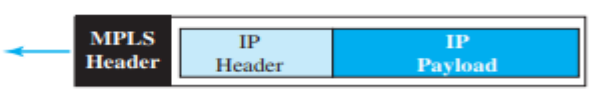
Forwarding Based on Label

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

Multi-Protocol Label Switching (MPLS)

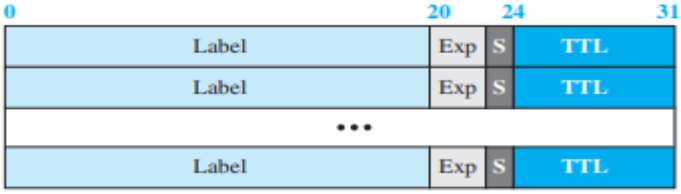
During the 1980s, several vendors created routers that implement switching technology. Later IETF approved a standard that is called Multi-Protocol Label Switching. In this standard, some conventional routers in the Internet can be replaced by MPLS routers, which can behave like a router and a switch.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

Figure 18.39 *MPLS header added to an IP packet*



The MPLS header is actually a stack of subheaders that is used for multilevel hierarchical switching.

Figure 18.40 *MPLS header made of a stack of labels*



The following is a brief description of each field:

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

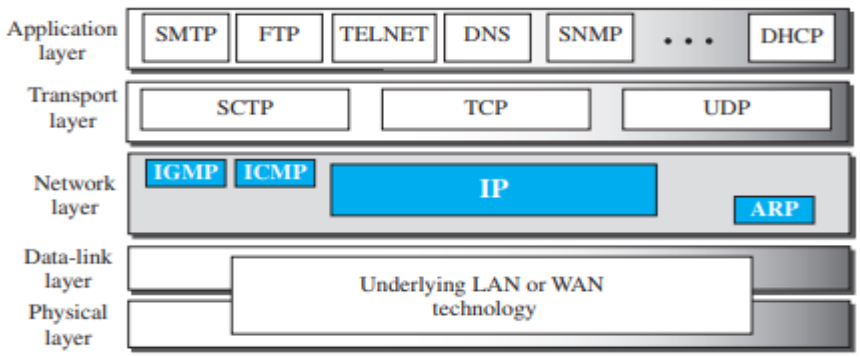
Hierarchical Switching

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

NETWORK LAYER PROTOCOLS

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

Figure 19.1 *Position of IP and other network-layer protocols in TCP/IP protocol suite*



The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery. The Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting. The Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses. Figure 19.1 shows the positions of these four protocols in the TCP/IP protocol suite.

INTERNET PROTOCOL (IP)

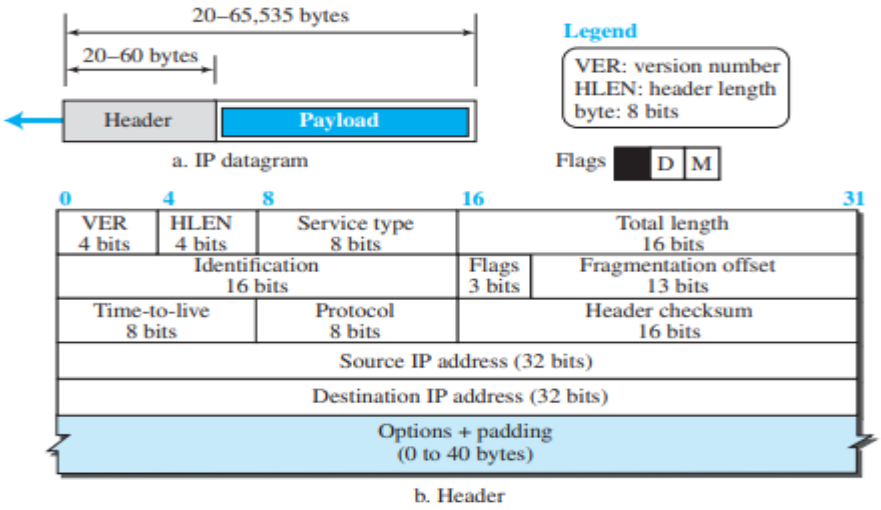
IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term best-effort means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network. If reliability is important, IPv4 must be paired with a reliable transport-layer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

IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route to the destination. This implies that datagrams sent by the same source to the same destination could arrive out of order. Again, IPv4 relies on a higher-level protocol to take care of all these problems.

Datagram Format:

IPv4 defines the format of a packet in which the data coming from the upper layer or other protocols are encapsulated. Packets used by the IP are called datagrams.

Figure 19.2 *IP datagram*

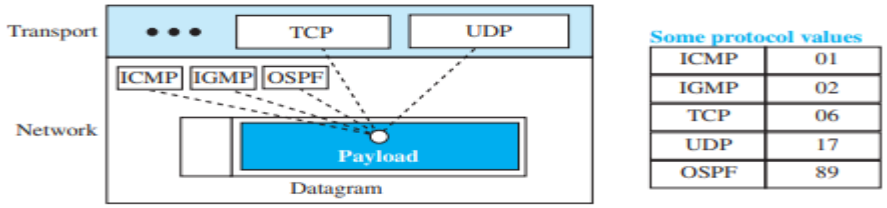


A datagram is a variable-length packet consisting of two parts: header and payload (data). The header is 20 to 60 bytes in length and contains information essential to routing and delivery. It is customary in TCP/IP to show the header in 4-byte sections.

- **Version Number**. The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.
- **Header Length**. The 4-bit header length (HLEN) field defines the total length of the datagram header in 4-byte words. The IPv4 datagram has a variable-length header.
- **Service Type.** In the original design of the IP header, this field was referred to as type of service (TOS), which defined how the datagram should be handled. In the late 1990s, IETF redefined the field to provide differentiated services (DiffServ).
- **Total Length.** This 16-bit field defines the total length (header plus data) of the IP datagram in bytes. A 16-bit number can define a total length of up to 65,535 (when all bits are 1s). However, the size of the datagram is normally much less than this. This field helps the receiving device to know when the packet has completely arrived.
- **Identification, Flags, and Fragmentation Offset.** These three fields are related to the fragmentation of the IP datagram when the size of the datagram is larger than the underlying network can carry.

- **Time-to-live.** Due to some malfunctioning of routing protocols (discussed later) a datagram may be circulating in the Internet, visiting some networks over and over without reaching the destination. This may create extra traffic in the Internet. The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram. When a source host sends the datagram, it stores a number in this field. This value is approximately two times the maximum number of routers between any two hosts. Each router that processes the datagram decrements this number by one. If this value, after being decremented, is zero, the router discards the datagram.
- **Protocol**. In TCP/IP, the data section of a packet, called the payload, carries the whole packet from another protocol. A datagram, for example, can carry a packet belonging to any transport-layer protocol such as UDP or TCP. A datagram can also carry a packet from other protocols that directly use the service of the IP, such as some routing protocols or some auxiliary protocols.

Figure 19.3 *Multiplexing and demultiplexing using the value of the protocol field*



- **Header checksum**. IP is not a reliable protocol; it does not check whether the payload carried by a datagram is corrupted during the transmission. IP puts the burden of error checking of the payload on the protocol that owns the payload, such as UDP or TCP. The datagram header, however, is added by IP, and its error-checking is the responsibility of IP. Errors in the IP header can be a disaster.
- **Source and Destination Addresses**. These 32-bit source and destination address fields define the IP address of the source and destination respectively.
- **Options**. A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging.
- **Payload.** Payload, or data, is the main reason for creating a datagram. Payload is the packet coming from other protocols that use the service of IP. Comparing a datagram to a postal package, payload is the content of the package; the header is only the information written on the package.

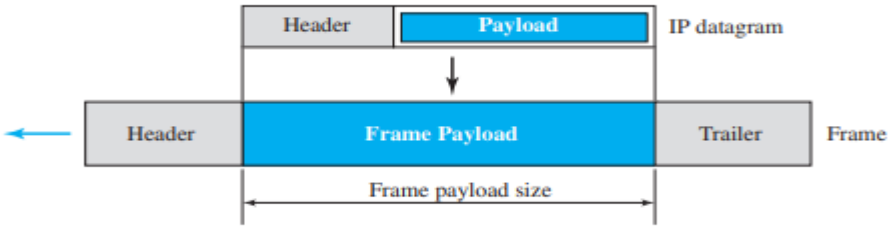
Fragmentation

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

Maximum Transfer Unit (MTU)

Each link-layer protocol has its own frame format. One of the features of each format is the maximum size of the payload that can be encapsulated. In other words, when a datagram is encapsulated in a frame, the total size of the datagram must be less than this maximum size, which is defined by the restrictions imposed by the hardware and software used in the network.

Figure 19.5 *Maximum transfer unit (MTU)*



MTU: Maximum size of frame payload

In order to make the IP protocol independent of the physical network, the designers decided to make the maximum length of the IP datagram equal to 65,535 bytes. This makes transmission more efficient if one day we use a link-layer protocol with an MTU of this size. However, for other physical networks, we must divide the datagram to make it possible for it to pass through these networks. This is called fragmentation.

When a datagram is fragmented, each fragment has its own header with most of the fields repeated, but some have been changed. A fragmented datagram may itself be fragmented if it encounters a network with an even smaller MTU. In other words, a datagram may be fragmented several times before it reaches the final destination.

A datagram can be fragmented by the source host or any router in the path. The reassembly of the datagram, however, is done only by the destination host, because each fragment becomes an independent datagram. Whereas the fragmented datagram can travel through different routes, and we can never control or guarantee which route a fragmented datagram may take, all of the fragments belonging to the same datagram should finally arrive at the destination host. So it is logical to do the reassembly at the final destination. An even stronger objection for reassembling packets during the transmission is the loss of efficiency it incurs.

Fields Related to Fragmentation

We mentioned before that three fields in an IP datagram are related to fragmentation: **identification, flags, and fragmentation offset.**

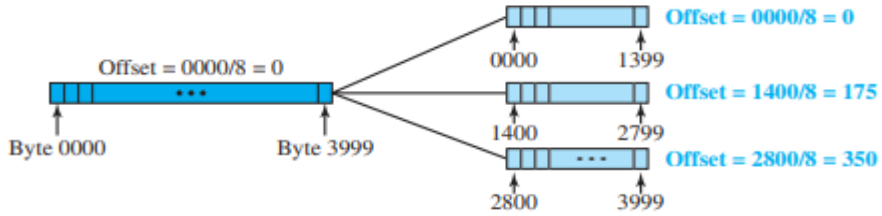
The 16-bit identification field identifies a datagram originating from the source host. The combination of the identification and source IP address must uniquely define a datagram as it leaves the source host. To guarantee uniqueness, the IP protocol uses a counter to label the datagrams. The counter is initialized to a positive number. When the IP protocol sends a datagram, it copies the current value of the counter to the identification field and increments the counter by one. As long as the counter is kept in the main memory, uniqueness is guaranteed. When a datagram is fragmented, the value in the identification field is copied into all fragments.

The 3-bit flags field defines three flags. The leftmost bit is reserved (not used). The second bit (D bit) is called the do not fragment bit. If its value is 1, the machine must not fragment the datagram. If it cannot pass the datagram through any available physical network, it discards the datagram and sends an ICMP error message to the source host (discussed later). If its value is 0, the datagram can be fragmented if necessary. The third bit (M bit) is called the more fragment bit. If its value is 1, it means the datagram is not the last fragment; there are more fragments after this one. If its value is 0, it means this is the last or only fragment.

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

units of 8 bytes. Figure 19.6 shows a datagram with a data size of 4000 bytes fragmented into three fragments. The bytes in the original datagram are numbered 0 to 3999. The first fragment carries bytes 0 to 1399. The offset for this datagram is 0/8 = 0. The second fragment carries bytes 1400 to 2799; the offset value for this fragment is 1400/8 = 175. Finally, the third fragment carries bytes 2800 to 3999. The offset value for this fragment is 2800/8 = 350.

Figure 19.6 *Fragmentation example*



Options

The header of the IPv4 datagram is made of two parts: a fixed part and a variable part. The fixed part is 20 bytes long.

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

Single-Byte Options

There are two single-byte options.

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

Multliple-Byte Options

There are four multiple-byte options.

- **Record Route:** A record route option is used to record the Internet routers that handle the datagram. It can list up to nine router addresses. It can be used for debugging and management purposes.
- **Strict Source Route**: A strict source route option is used by the source to predetermine a route for the datagram as it travels through the Internet.
- **Loose Source Route:** A loose source route option is similar to the strict source route, but it is less rigid. Each router in the list must be visited, but the datagram can visit other routers as well.
- **Timestamp**: A timestamp option is used to record the time of datagram processing by a router. The time is expressed in milliseconds from midnight, Universal time or Greenwich mean time.

Security of IPv4 Datagrams

There are three security issues that are particularly applicable to the IP protocol: **packet sniffing, packet modification, and IP spoofing.**

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

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

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

IPSec

IPSec provides the following four services:

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

ICMP v4

The IPv4 has no error-reporting or error-correcting mechanism. The IP protocol also lacks a mechanism for host and management queries. A host sometimes needs to determine if a router or another host is alive. And sometimes a network manager needs information from another host or router.

The **Internet Control Message Protocol** version 4 (ICMPv4) has been designed to compensate for the above two deficiencies. It is a companion to the IP protocol. ICMP itself is a network-layer protocol. However, its messages are not passed directly to the data-link layer as would be expected. Instead, the messages are first encapsulated inside IP datagrams before going

to the lower layer. When an IP datagram encapsulates an ICMP message, the value of the protocol field in the IP datagram is set to 1 to indicate that the IP payroll is an ICMP message.

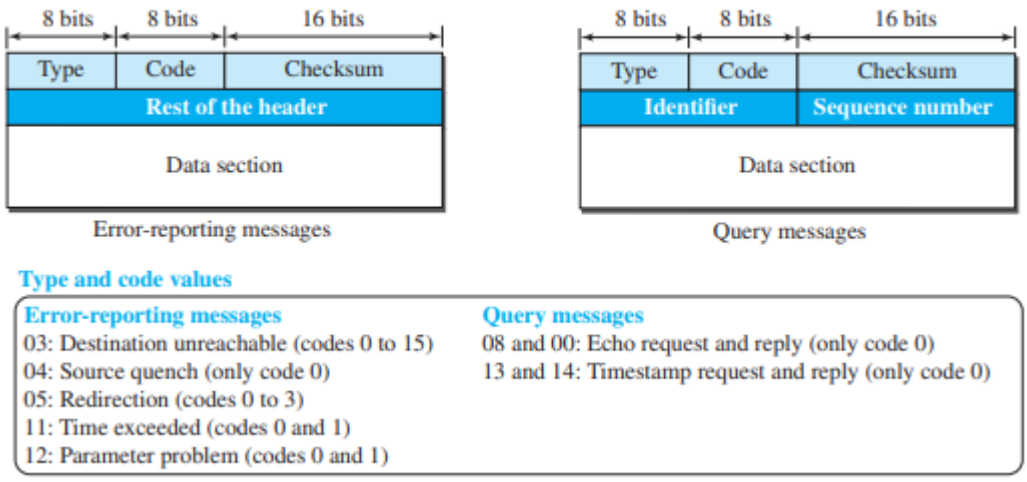
MESSAGES:

ICMP messages are divided into two broad categories: error-reporting messages and query messages.

The error-reporting messages report problems that a router or a host (destination) may encounter when it processes an IP packet. The query messages, which occur in pairs, help a host or a network manager get specific information from a router or another host.

An ICMP message has an 8-byte header and a variable-size data section. Although the general format of the header is different for each message type, the first 4 bytes are common to all. As Figure 19.8 shows, the first field, ICMP type, defines the type of the message. The code field specifies the reason for the particular message type. The last common field is the checksum field (to be discussed later in the chapter). The rest of the header is specific for each message type.

Figure 19.8 *General format of ICMP messages*



The data section in error messages carries information for finding the original packet that had the error. In query messages, the data section carries extra information based on the type of query.

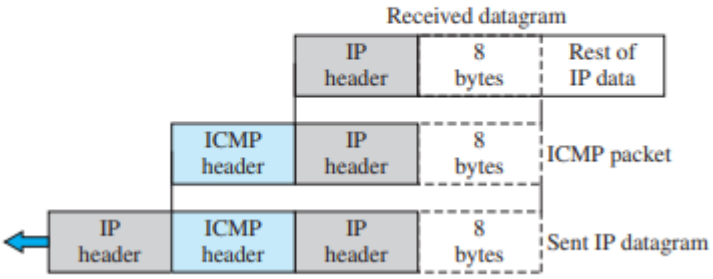
Error Reporting Messages

The following are important points about ICMP error messages:

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

The original datagram header is added to give the original source, which receives the error message, information about the datagram itself. The 8 bytes of data are included because, as we will see in Chapter 24 on UDP and TCP protocols, the first 8 bytes provide information about the port numbers (UDP and TCP) and sequence number (TCP). This information is needed so the source can inform the protocols (TCP or UDP) about the error. ICMP forms an error packet, which is then encapsulated in an IP datagram.

Figure 19.9 *Contents of data field for the error messages*



Destination Unreachable

The most widely used error message is the destination unreachable (type 3). This message uses different codes (0 to 15) to define the type of error message and the reason why a datagram has not reached its final destination. For example, code 0 tells the source that a host is unreachable.

Source Quench

Another error message is called the source quench (type 4) message, which informs the sender that the network has encountered congestion and the datagram has been dropped; the source needs to slow down sending more datagrams.

Redirection Message

The redirection message (type 5) is used when the source uses a wrong router to send out its message. The router redirects the message to the appropriate router, but informs the source that it needs to change its default router in the future. The IP address of the default router is sent in the message.

Parameter Problem

A parameter problem message (type 12) can be sent when either there is a problem in the header of a datagram (code 0) or some options are missing or cannot be interpreted (code 1).

Query Messages

Query messages are used to probe or test the liveliness of hosts or routers in the Internet, find the one-way or the round-trip time for an IP datagram between two devices, or even find out whether the clocks in two devices are synchronized. Naturally, query messages come in pairs: request and reply. The echo request (type 8) and the echo reply (type 0) pair of messages are used by a host or a router to test the liveliness of another host or router. A host or router sends an echo request message to another host or router; if the latter is alive, it responds with an echo reply message.

There are two debugging tools: **ping and traceroute**. The timestamp request (type 13) and the timestamp reply (type 14) pair of messages are used to find the round-trip time between two devices or to check whether the clocks in two devices are synchronized. The timestamp

request message sends a 32-bit number, which defines the time the message is sent. The timestamp reply resends that number, but also includes two new 32-bit numbers representing the time the request was received and the time the response was sent.

Deprecated Messages

Three pairs of messages are declared obsolete by IETF:

1. **Information request and replay messages** are not used today because their duties are done by the **Address Resolution Protocol** (ARP)
2. **Address mask request and reply messages** are not used today because their duties are done by the **Dynamic Host Configuration Protocol** (DHCP)
3. **Router solicitation and advertisement messages** are not used today because their duties are done by the **Dynamic Host Configuration Protocol** (DHCP)

Debugging Tools

We can determine the viability of a host or router. We can trace the route of a packet. We introduce two tools that use ICMP for debugging: **ping and traceroute**.

Ping

We can use the ping program to find if a host is alive and responding. We use ping here to see how it uses ICMP packets. The source host sends ICMP echo-request messages; the destination, if alive, responds with ICMP echo-reply messages. The ping program sets the identifier field in the echo-request and echo-reply message and starts the sequence number from 0; this number is incremented by 1 each time a new message is sent. Note that ping can calculate the round-trip time. It inserts the sending time in the data section of the message. When the packet arrives, it subtracts the arrival time from the departure time to get the round-trip time (RTT).

```
$ ping auniversity.edu
PING auniversity.edu (152.181.8.3)   56 (84)  bytes of data.
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=0    ttl=62    time=1.91 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=1    ttl=62    time=2.04 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=2    ttl=62    time=1.90 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=3    ttl=62    time=1.97 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=4    ttl=62    time=1.93 ms
64 bytes from auniversity.edu (152.181.8.3): icmp_seq=5    ttl=62    time=2.00 ms
--- auniversity.edu statistics ---
6 packets transmitted, 6 received, 0% packet loss
rtt min/avg/max = 1.90/1.95/2.04 ms
```

Traceroute or Tracert

The traceroute program in UNIX or tracert in Windows can be used to trace the path of a packet from a source to the destination. It can find the IP addresses of all the routers that are visited along the path. The program is usually set to check for the maximum of 30 hops (routers) to be visited. The number of hops in the Internet is normally less than this.

```
$ traceroute printers.com
traceroute to printers.com (13.1.69.93), 30 hops max, 38-byte packets
1 route.front.edu        (153.18.31.254)    0.622 ms    0.891 ms    0.875 ms
2 ceneric.net            (137.164.32.140)   3.069 ms    2.875 ms    2.930 ms
3 satire.net             (132.16.132.20)    3.071 ms    2.876 ms    2.929 ms
4 alpha.printers.com     (13.1.69.93)       5.922 ms    5.048 ms    4.922 ms
```

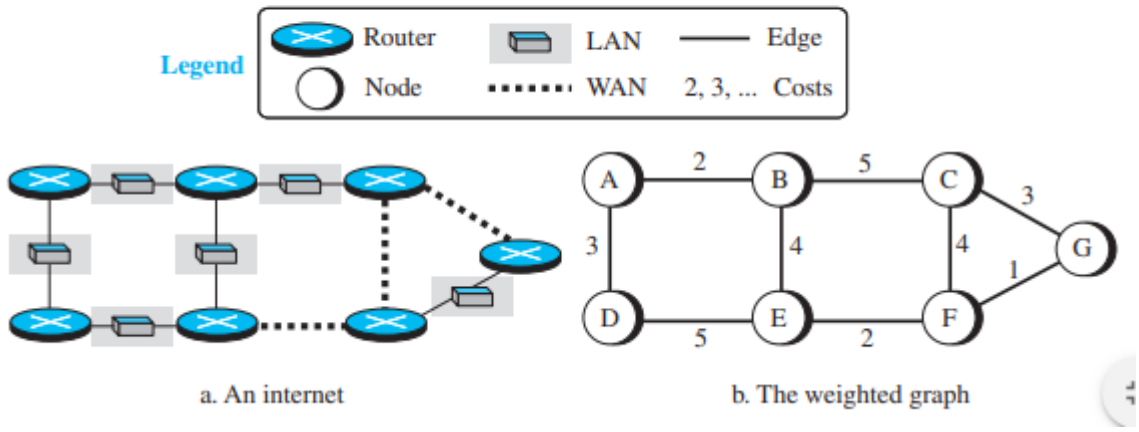
UNICAST ROUTING ALGORITHMS

In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.

An Internet as a Graph

To find the best route, an internet can be modeled 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, in fact, modeled 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, in this case, are distances between cities. In routing, however, the cost of an edge has a different interpretation in different routing protocols, which we discuss in a later section. For the moment, we assume that there is a cost associated with each edge. If there is no edge between the nodes, the cost is infinity.

Figure 20.1 *An internet and its graphical representation*



a. An internet b. The weighted graph

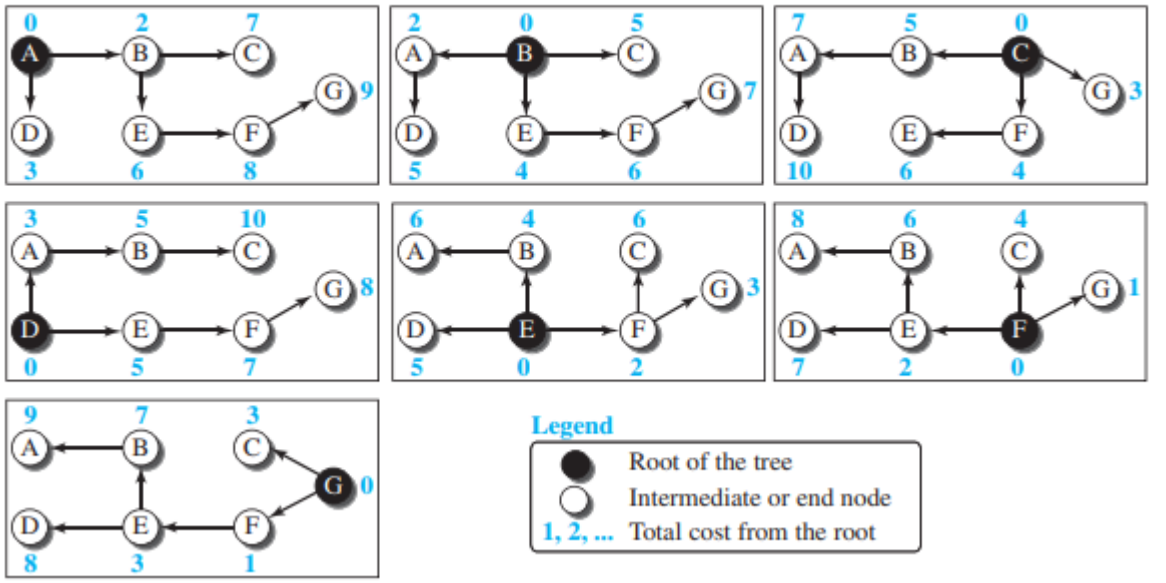
Least-Cost Routing

When an internet is modeled 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. In other words, 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 Figure 20.1, 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 criteria.

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 of 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 20.2 *Least-cost trees for nodes in the internet of Figure 20.1*



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 20.2, the route from A to F in A's tree is (A → B → E → F), but the route from F to A in F's tree is (F → E → B → A), which is the inverse of the first route. The cost is 8 in each case.
2. Instead of travelling from X to Z using X's tree, we can travel from X to Y using X's tree and continue from Y to Z using Y's tree. For example, in Figure 20.2, 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).

Distance-Vector Routing

The distance-vector (DV) routing uses the goal to find the best route. 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 neighbors. The incomplete trees are exchanged between immediate neighbors to make the trees more and more complete and to represent the whole internet.

Bellman-Ford Equation

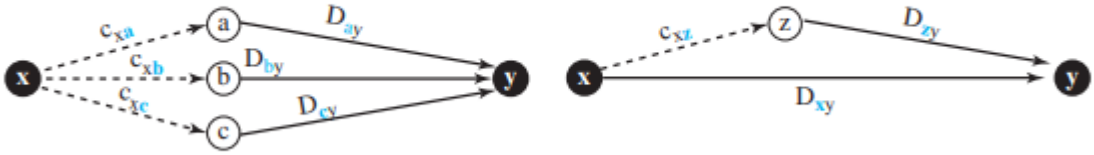
The heart of distance-vector routing is the famous Bellman-Ford equation. This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, . . .) when the costs between the source and the intermediary nodes 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.

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

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

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

Figure 20.3 *Graphical idea behind Bellman-Ford equation*



a. General case with three intermediate nodes

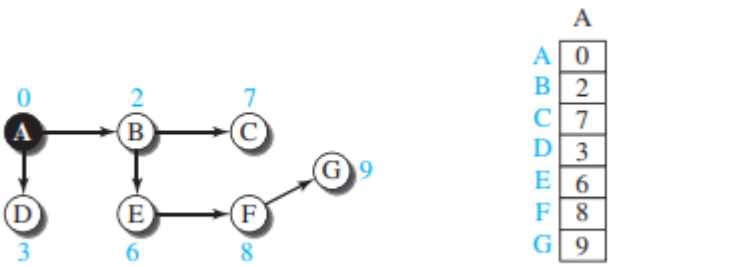
b. Updating a path with a new route

We can say that the Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths. In Figure 20.3, we can think of (a⟶y), (b⟶y), and (c⟶y) as previously established least-cost paths and (x⟶y) as the new least-cost path. We can even think of this equation as the builder of a new least-cost tree from previously established least-cost trees if we use the equation repeatedly. In other words, 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

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 20.4 *The distance vector corresponding to a tree*



a. Tree for node A

b. Distance vector for node A

Note that 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.

Figure 20.5 *The first distance vector for an internet*

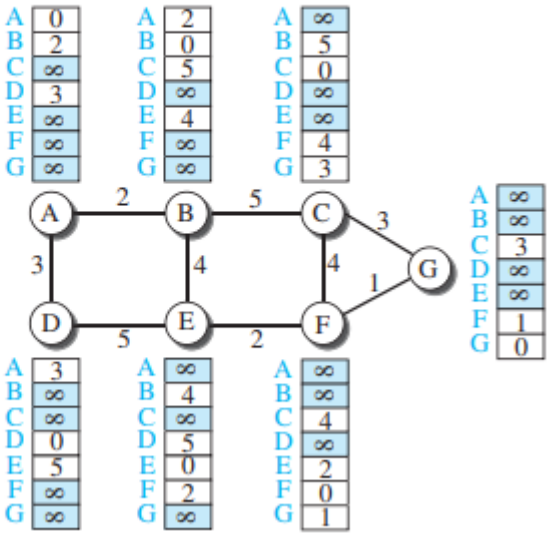
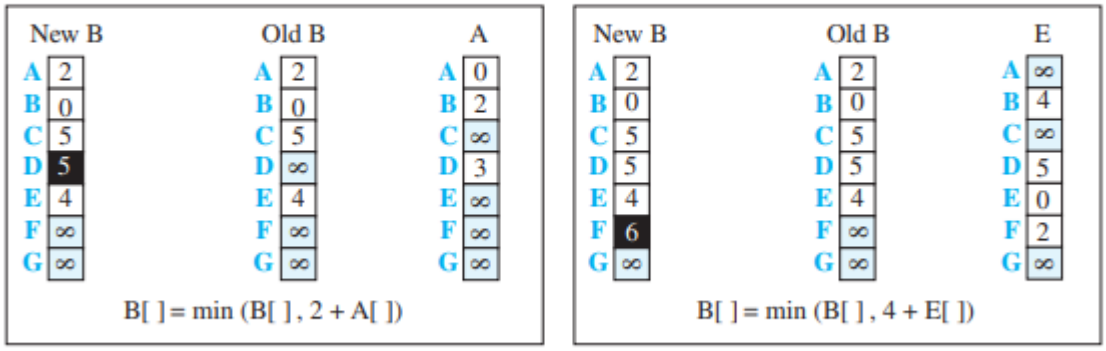


Figure 20.5 shows all distance vectors for our internet. However, we need to mention that these vectors are made asynchronously, when the corresponding node has been booted; the existence of all of them in a figure does not mean synchronous creation of them.

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 neighbors. After a node receives a distance vector from a neighbor, 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.

Figure 20.6 *Updating distance vectors*



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

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

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 cBA = 2. In the second event, node E has sent its vector to node B. Node B

updates its vector using the cost cEA = 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). We hope that we have convinced the reader that 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 neighbors. Even if its neighbors have received the previous vector, the updated one may help more.

Table 20.1 *Distance-Vector Routing Algorithm for a Node*

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

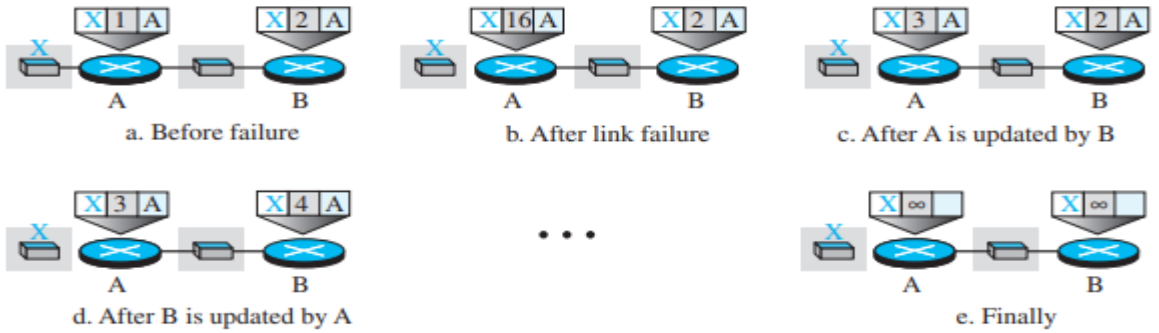
Count to Infinity

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 20.7.

Figure 20.7 *Two-node instability*



The figure shows a system with three nodes. We have shown only the portions of the forwarding table needed for our discussion. 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, assuming that B has found a way to reach X, immediately updates its forwarding table. Now A sends its new update to B. Now B thinks that something has been changed around A and updates its forwarding table. The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached. However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A receives a packet destined for X, the packet goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem. A few solutions have been proposed for instability of this kind.

Split Horizon

One solution to instability is called split horizon. In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows). Taking information from node A, modifying it, and sending it back to node A is what creates the confusion. In our scenario, node B eliminates the last line of its forwarding table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later, when node A sends its forwarding table to B, node B also corrects its forwarding table. The system becomes stable after the first update: both node A and node B know that X is not reachable.

Poison Reverse

Using the split-horizon strategy has one drawback. Normally, the corresponding protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the source of information

was A) or because B has not received any news about X recently. In the poison reverse strategy B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

Three-Node Instability

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

Link-State Routing

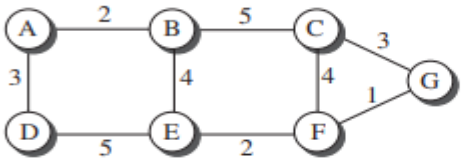
A routing algorithm that directly follows our discussion for creating least-cost trees and forwarding tables is link-state (LS) routing. This method uses the term link-state to define the characteristic of a link (an edge) that represents a network in the internet. In this algorithm the cost associated 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.

Figure 20.8 shows an example of an LSDB for the graph in Figure 20.1. 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.

Figure 20.8 *Example of a link-state database*



a. The weighted graph

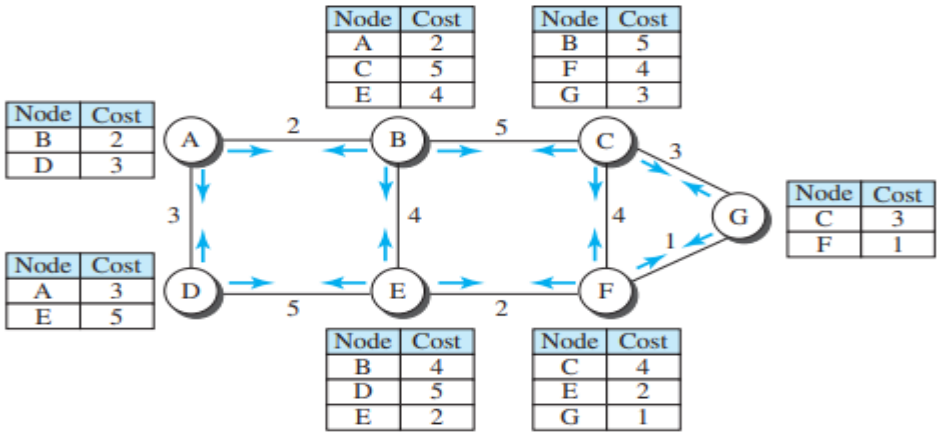
	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

b. Link state database

Now the question is how each node can create this LSDB that contains information about the whole internet. This can be done by a process called flooding. Each node can send some greeting messages to all its immediate neighbors (those nodes to which it is connected directly) to collect two pieces of information for each neighboring node: the identity of the node 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 20.9 for our internet in Figure 20.1. 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

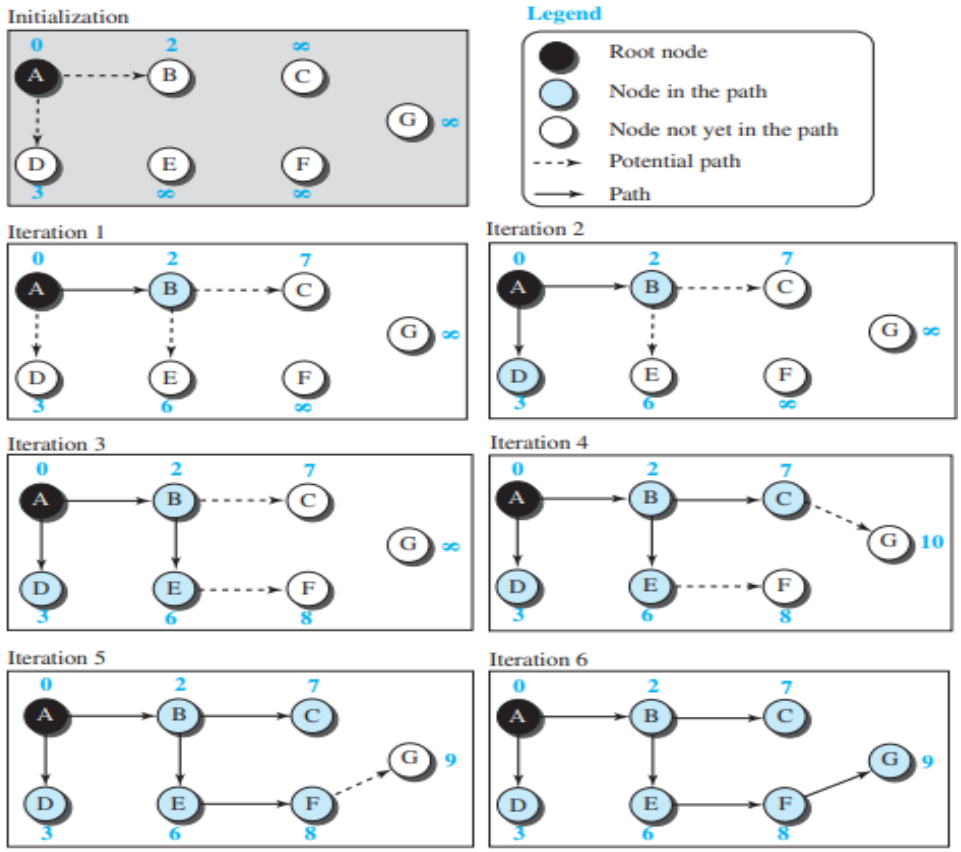
out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the network (where a node has only one interface). We need to convince ourselves that, after receiving all new LSPs, each node creates the comprehensive LSDB as shown in Figure 20.9. This 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.

Figure 20.9 *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 neighbors what it knows about the whole internet; in the link-state routing algorithm, each router tells the whole internet what it knows about its neighbors.

Figure 20.10 *Least-cost tree*



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.

Table 20.2 *Dijkstra's Algorithm*

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

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. For example, a router may belong to an organization that does not provide enough security or it may belong to a commercial rival of the sender which might inspect the packets for obtaining information. Least-cost routing does not prevent a packet from passing through an area when that area is in the least-cost path. In other words, the least-cost goal, applied by LS or DV

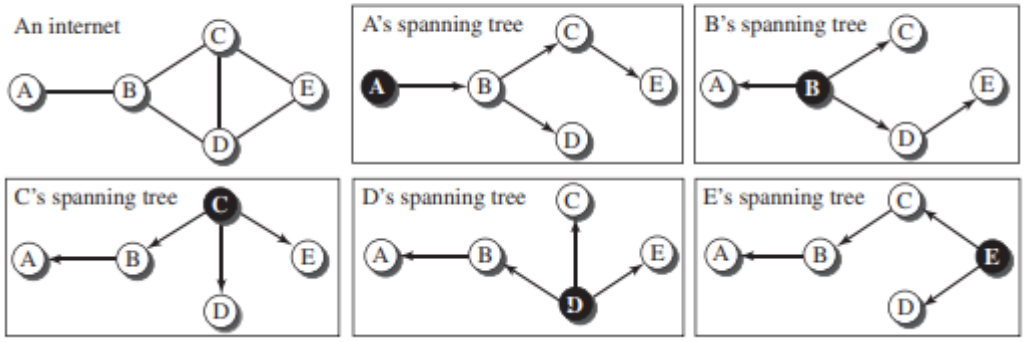
routing, does not allow a sender to apply specific policies to the route a packet may take. Aside from safety and security, there are occasions, as discussed in the next section, in which the goal of routing is merely reachability: to allow the packet to reach its destination more efficiently without assigning costs to the route.

To respond to these demands, a third routing algorithm, called path-vector (PV) routing has been devised. 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. In other words, 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, we discuss the principle of this method in this section as though applied to an internet. In the next section, we show how it is used in the Internet.

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, however, 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 (something similar to least-cost). Another common policy is to avoid some nodes as the middle node in a route. Figure 20.11 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.

Figure 20.11 *Spanning trees in path-vector routing*



Creation of Spanning Trees

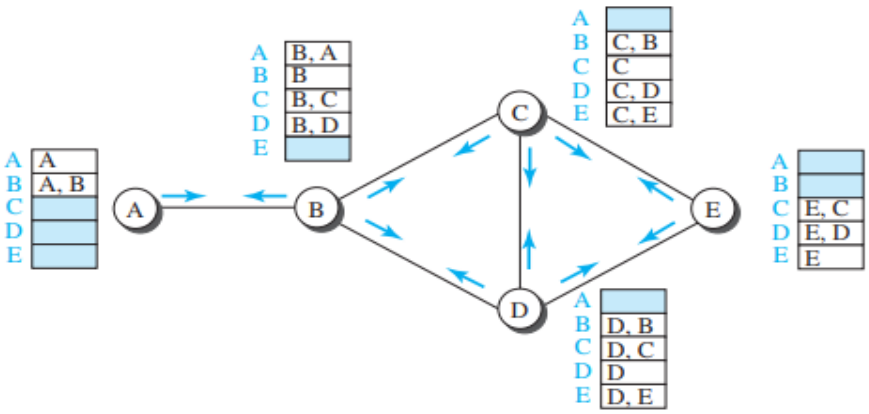
Path-vector routing, like distance-vector routing, is an asynchronous and distributed routing algorithm. The spanning trees are made, gradually and asynchronously, by each node. When a node is booted, it creates a path vector based on the information it can obtain about its immediate neighbor. A node sends greeting messages to its immediate neighbors to collect these pieces of information. Figure 20.12 shows all of these path vectors for our internet in Figure 20.11. Note, however, that we do not mean that all of 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 neighbors after they have been created (arrows). Each node, after the creation of the initial path vector, sends it to all its immediate neighbors. Each node, when it receives a path vector from a neighbor, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost. We can define this equation as

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

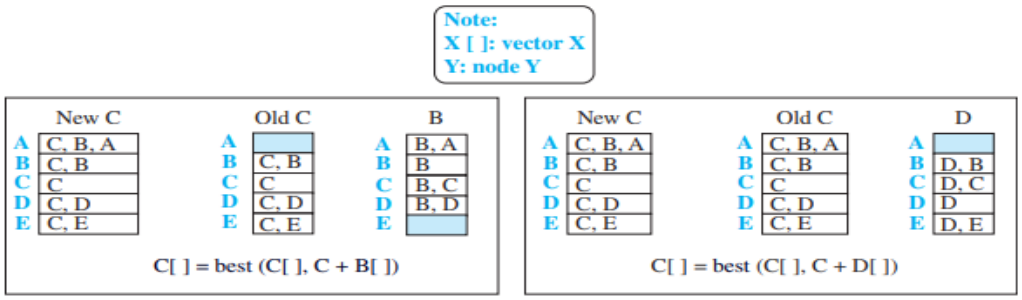
In this equation, the operator (+) means to add x to the beginning of the path. We also need to be cautious to avoid adding a node to an empty path because an empty path means one that does not exist.

Figure 20.12 *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 20.13 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.

Figure 20.13 *Updating path vectors*



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

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

Table 20.3 *Path-vector algorithm for a node*

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

PROTOCOLS

Three common protocols used in the Internet: **Routing Information Protocol (RIP)**, based on the distance-vector algorithm, **Open Shortest Path First (OSPF)**, based on the link-state algorithm, and **Border Gateway Protocol (BGP)**, based on the path-vector algorithm.

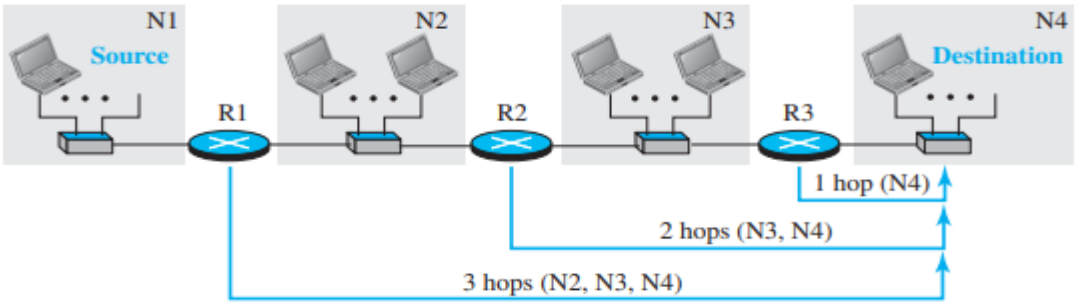
Routing Information Protocol (RIP)

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

Hop Count

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 20.15 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.

Figure 20.15 *Hop counts in RIP*



Forwarding Tables

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. Figure 20.16 shows the three forwarding tables for the routers in Figure 20.15. 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.

Figure 20.16 *Forwarding tables*

Forwarding table for R1

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

Forwarding table for R2

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

Forwarding table for R3

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

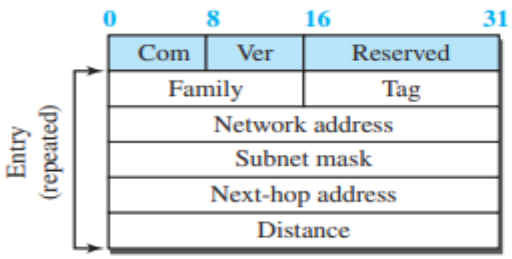
RIP Implementation

RIP has gone through two versions: RIP-1 and RIP-2. The second version is backward compatible with the first section; it allows the use of more information in the RIP messages that were set to 0 in the first version.

RIP Messages

Two RIP processes, a client and a server, like any other processes, need to exchange messages. RIP-2 defines the format of the message, as shown in Figure 20.17

Figure 20.17 *RIP message format*



Fields

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

RIP 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. An unsolicited response message, on the other hand, is sent periodically, every 30 seconds or when there is a change in the forwarding table.

RIP Algorithm

RIP implements the same algorithm as the distance-vector routing algorithm we discussed in the previous section. However, some changes need to be made to the algorithm to enable a router to update its forwarding table:

- Instead of sending only distance vectors, a router needs to send the whole contents of its forwarding table in a response message.
- The receiver adds one hop to each cost and changes the next router field to the address of the sending router. We call each route in the modified forwarding table the received route and each route in the old forwarding table the old route.

  The received router selects the old routes as the new ones 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. This is the case where the route was actually advertised by the same router in the past, but now the situation has been changed. For example, suppose a neighbor has previously advertised a route to a destination with cost 3, but now there is no path between this neighbor and that destination. The neighbor advertises this destination with cost value infinity (16 in RIP). The receiving router must not ignore this value even though its old route has a lower cost to the same destination.
- The new forwarding table needs to be sorted according to the destination route (mostly using the longest prefix first).

Timers in RIP

RIP uses three timers to support its operation.

- The periodic timer controls the advertising of regular update messages.
- The expiration timer governs the validity of a route.
- The garbage collection timer is used to purge a route from the forwarding table.

Performance

Before ending this section, let us briefly discuss the performance of RIP:

- **Update Messages**. The update messages in RIP have a very simple format and are sent only to neighbors; they are local. They do not normally create traffic because the routers try to avoid sending them at the same time.

- **Convergence of Forwarding Tables**. RIP uses the distance-vector algorithm, which can converge slowly if the domain is large, but, since RIP allows only 15 hops in a domain (16 is considered as infinity), there is normally no problem in convergence. The only problems that may slow down convergence are count-to-infinity and loops created in the domain; use of poison-reverse and split-horizon strategies added to the RIP extension may alleviate the situation.
- **Robustness.** As we said before, distance-vector routing is based on the concept that each router sends what it knows about the whole domain to its neighbors. This means that the calculation of the forwarding table depends on information received from immediate neighbors, which in turn receive their information from their own neighbors. If there is a failure or corruption in one router, the problem will be propagated to all routers and the forwarding in each router will be affected.

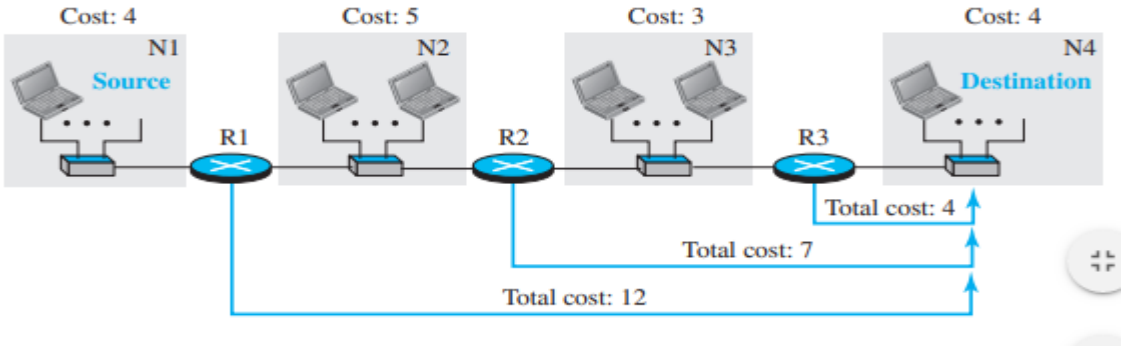
Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) is also an intradomain routing protocol like RIP, but 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. An administration can also decide to use the hop count as the cost. An interesting point about the cost in OSPF is that different service types (TOSs) can have different weights as the cost.

Figure 20.19 *Metric in OSPF*



Forwarding Tables

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

Figure 20.20 *Forwarding tables in OSPF*

Forwarding table for R1

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

Forwarding table for R2

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

Forwarding table for R3

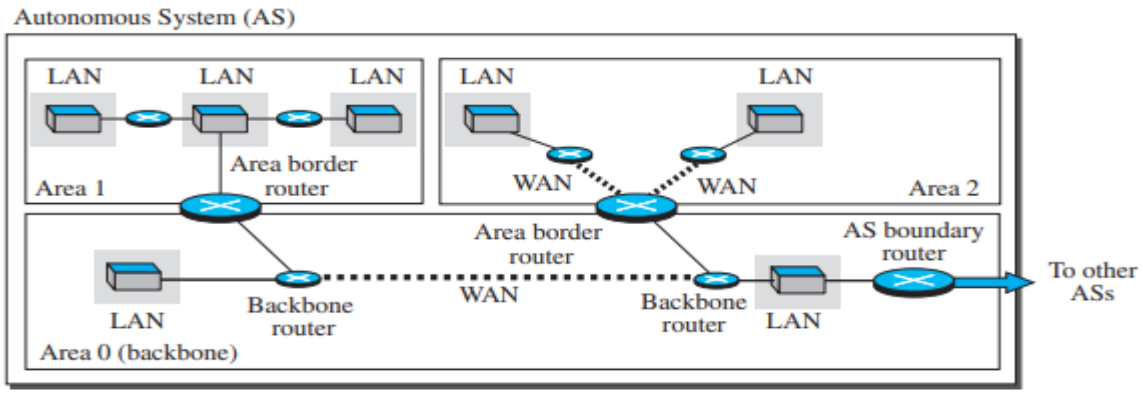
Destination network	Next router	Cost
N1	R2	12
N2	R2	8
N3	——	3
N4	——	4

Areas

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

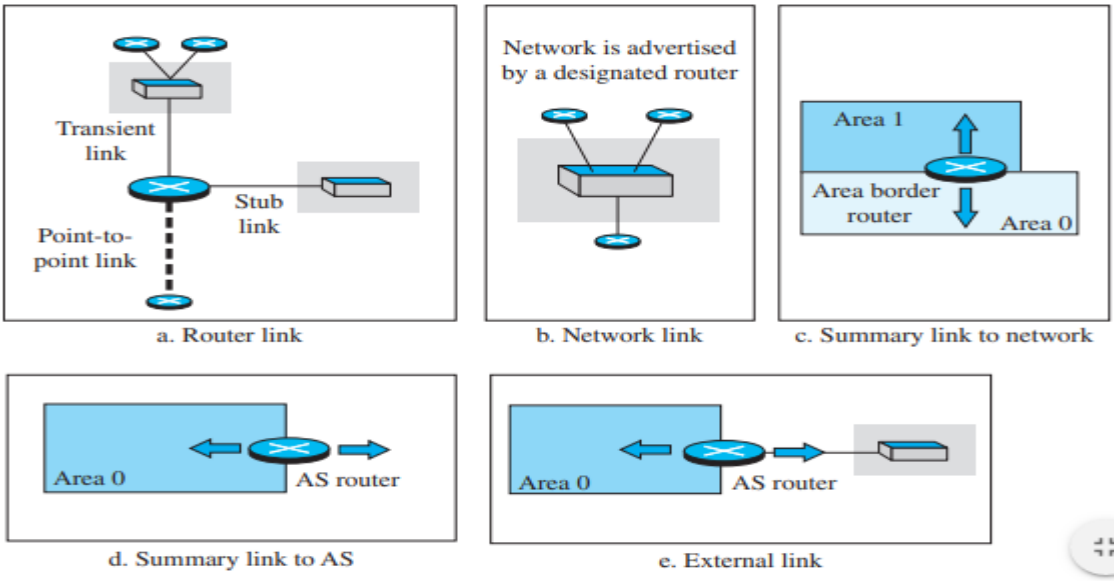
However, each router in an area needs to know the information about the link states not only in its area but also in other areas. For this reason, one of the areas in the AS is designated as the backbone area, responsible for gluing the areas together. The routers in the backbone area are responsible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

Figure 20.21 *Areas in an autonomous system*



Link-State Advertisement

Figure 20.22 *Five different LSPs*



We need to advertise the existence of different entities as nodes, the different types of links that connect each node to its neighbors, and the different types of cost associated with each link. This means we need different types of advertisements, each capable of advertising different situations. We can have five types of link-state advertisements: router link, network link, summary link to network, summary link to AS border router, and external link.

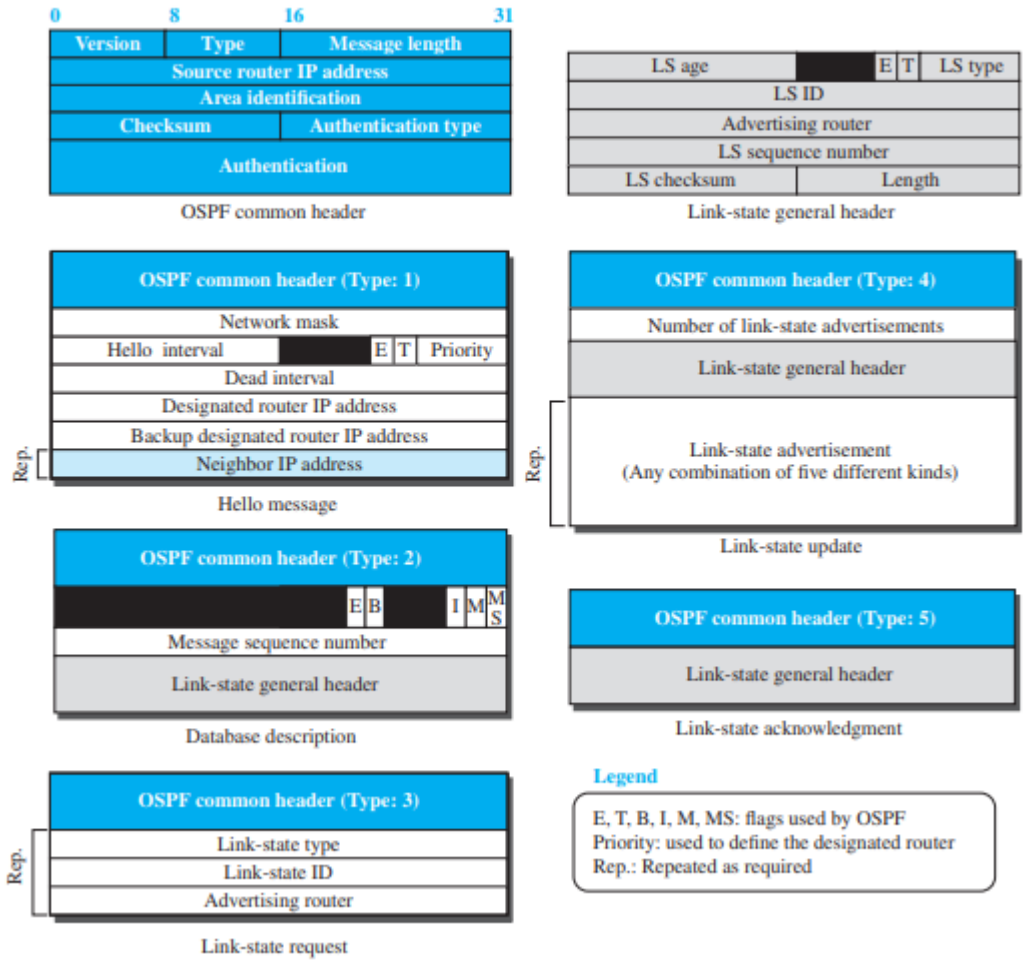
OSPF Implementation

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, although OSPF is a routing protocol to help IP to route its datagrams inside an AS, the OSPF messages are encapsulated inside datagrams. OSPF has gone through two versions: version 1 and version 2. Most implementations use version 2.

OSPF Messages

OSPF is a very complex protocol; it uses five different types of messages.

Figure 20.23 *OSPF message formats*



In Figure 20.23, we first show the format of the OSPF common header (which is used in all messages) and the link-state general header (which is used in some messages). We then give the outlines of five message types used in OSPF. The hello message (type 1) is used by a router to introduce itself to the neighbors and announce all neighbors that it already knows. The

database description message (type 2) is normally sent in response to the hello message to allow a newly joined router to acquire the full LSDB. The linkstate 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, in fact, has five different versions (router link, network link, summary link to network, summary link to AS border router, and external link), as we discussed before. The link-state acknowledgment message (type 5) is used to create reliability in OSPF; each router that receives a link-state update message needs to acknowledge it.

Authentication

As Figure 20.23 shows, the OSPF common header has the provision for authentication of the message sender. This prevents a malicious entity from sending OSPF messages to a router and causing the router to become part of the routing system to which it actually does not belong.

OSPF Algorithm

OSPF implements the link-state routing algorithm we discussed in the previous section. However, some changes and augmentations need to be added to the algorithm:

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

Performance

Before ending this section, let us briefly discuss the performance of OSPF:

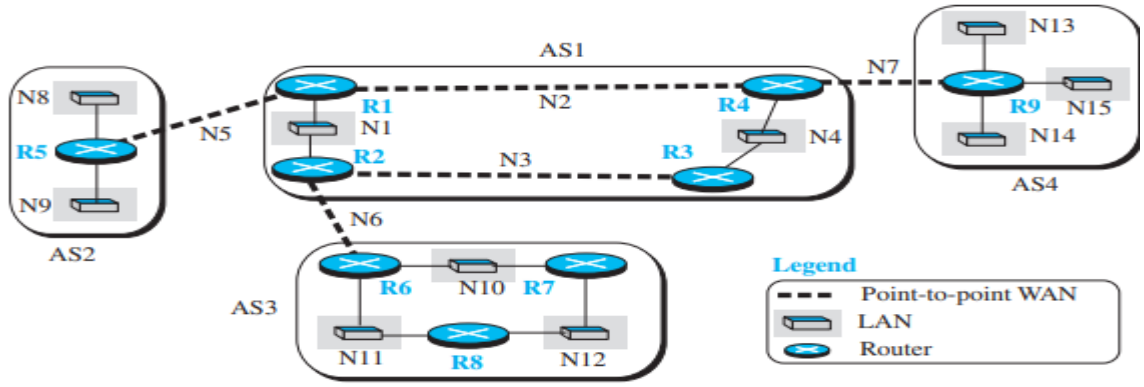
- **Update Messages**. The link-state messages in OSPF have a somewhat complex format. They also are flooded to the whole area. If the area is large, these messages may create heavy traffic and use a lot of bandwidth.
- **Convergence of Forwarding Tables**. When the flooding of LSPs is completed, each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

Border Gateway Protocol Version 4 (BGP4)

The Border Gateway Protocol version 4 (BGP4) is the only interdomain routing protocol used in the Internet today. BGP4 is based on the path-vector algorithm we described before, but it is tailored to provide information about the reachability of networks in the Internet.

BGP, and in particular BGP4, is a complex protocol. In this section, we introduce the basics of BGP and its relationship with intradomain routing protocols (RIP or OSPF). Figure 20.24 shows an example of an internet with four autonomous systems. AS2, AS3, and AS4 are stub autonomous systems; AS1 is a transient one. In our example, data exchange between AS2, AS3, and AS4 should pass through AS1.

Figure 20.24 *A sample internet with four ASs*



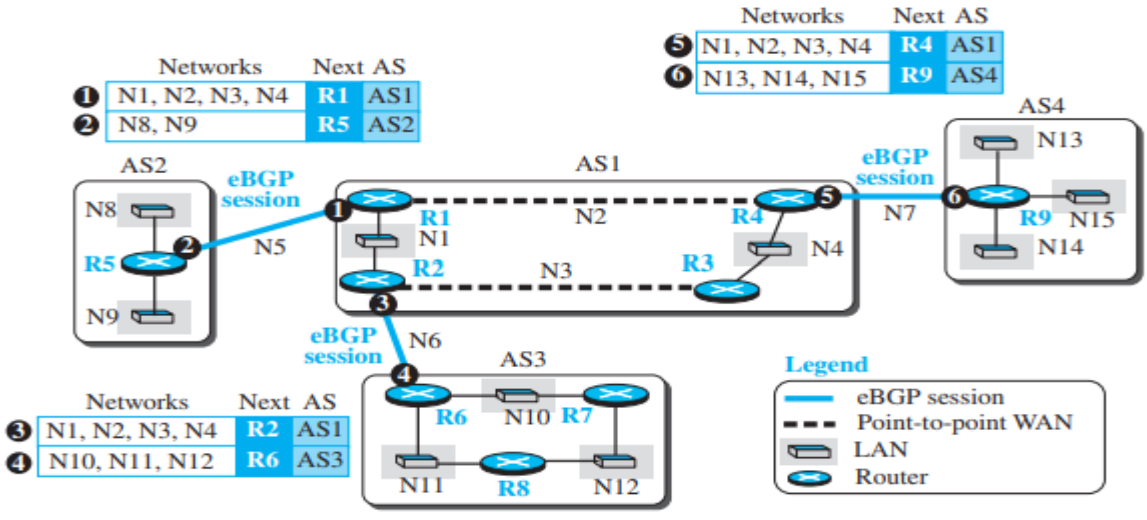
To enable each router to route a packet to any network in the internet, we first install a variation of BGP4, called external BGP (eBGP), on each border router (the one at the edge of each AS which is connected to a router at another AS). We then install the second variation of BGP, called internal BGP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

We can say that BGP is a kind of 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. In other words, a pair of client and server processes continuously communicate with each other to exchange messages. 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 are eligible in our example in Figure 20.24 form three pairs: R1-R5, R2-R6, and R4- R9.

Figure 20.25 *eBGP operation*



The connection between these pairs is established over three physical WANs (N5, N6, and N7). However, 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 parlance is referred to as a session.

There are two problems that need to be addressed:

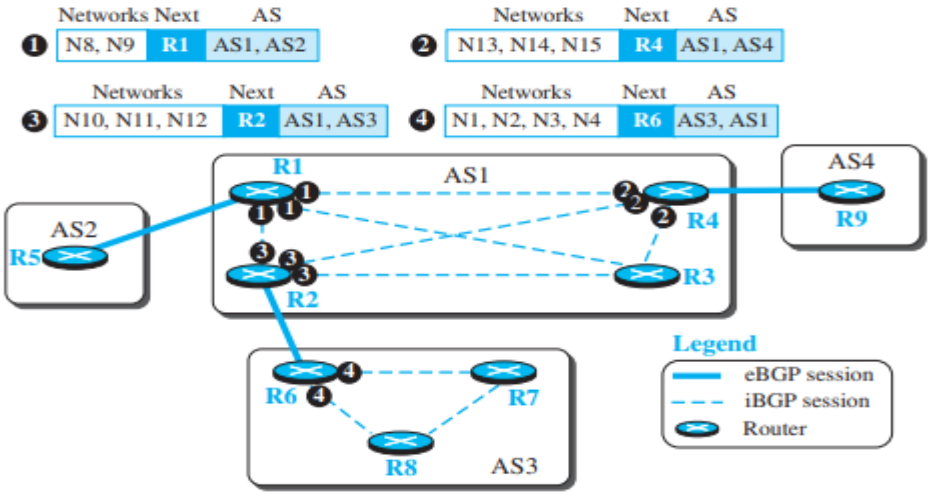
1. Some border routers do not know how to route a packet destined for nonneighbor ASs. For example, R5 does not know how to route packets destined for networks in AS3 and AS4. Routers R6 and R9 are in the same situation as R5: R6 does not know about networks in AS2 and AS4; R9 does not know about networks in AS2 and AS3.
2. None of the nonborder routers know how to route a packet destined for any networks in other ASs.

To address the above two problems, we need to allow all pairs of routers (border or nonborder) to run the second variation of the BGP protocol, iBGP.

Operation of Internal BGP (iBGP)

The iBGP protocol is similar to the eBGP protocol in that it uses the service of TCP on the well-known port 179, but it creates a session between any possible pair of routers inside an autonomous system. However, some points should be made clear. First, if an AS has only one router, there cannot be an iBGP session. For example, we cannot create an iBGP session inside AS2 or AS4 in our internet. Second, if there are n routers in an autonomous system, there should be [n × (n − 1) / 2] iBGP sessions in that autonomous system (a fully connected mesh) to prevent loops in the system. In other words, each router needs to advertise its own reachability to the peer in the session instead of flooding what it receives from another peer in another session.

Figure 20.26 *Combination of eBGP and iBGP sessions in our internet*



The first message (numbered 1) is sent by R1 announcing that networks N8 and N9 are reachable through the path AS1-AS2, but the next router is R1. This message is sent, through separate sessions, to R2, R3, and R4. Routers R2, R4, and R6 do the same thing but send different messages to different destinations. The interesting point is that, at this stage, R3, R7, and R8 create sessions with their peers, but they actually have no message to send. The updating process does not stop here. For example, after R1 receives the update message from R2, it combines the reachability information about AS3 with the reachability information it already knows about AS1 and sends a new update message to R5. Now R5 knows how to reach

networks in AS1 and AS3. The process continues when R1 receives the update message from R4. The point is that we need to make certain that at a point in time there are no changes in the previous updates and that all information is propagated through all ASs. At this time, each router combines the information received from eBGP and iBGP and creates what we may call a path table after applying the criteria for finding the best path, including routing policies.

Figure 20.27 *Finalized BGP path tables*

Path table for R1

Networks	Next	Path
N8, N9	R5	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R2

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R6	AS1, AS3
N13, N14, N15	R1	AS1, AS4

Path table for R3

Networks	Next	Path
N8, N9	R2	AS1, AS2
N10, N11, N12	R2	AS1, AS3
N13, N14, N15	R4	AS1, AS4

Path table for R4

Networks	Next	Path
N8, N9	R1	AS1, AS2
N10, N11, N12	R1	AS1, AS3
N13, N14, N15	R9	AS1, AS4

Path table for R5

Networks	Next	Path
N1, N2, N3, N4	R1	AS2, AS1
N10, N11, N12	R1	AS2, AS1, AS3
N13, N14, N15	R1	AS2, AS1, AS4

Path table for R6

Networks	Next	Path
N1, N2, N3, N4	R2	AS3, AS1
N8, N9	R2	AS3, AS1, AS2
N13, N14, N15	R2	AS3, AS1, AS4

Path table for R7

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R8

Networks	Next	Path
N1, N2, N3, N4	R6	AS3, AS1
N8, N9	R6	AS3, AS1, AS2
N13, N14, N15	R6	AS3, AS1, AS4

Path table for R9

Networks	Next	Path
N1, N2, N3, N4	R4	AS4, AS1
N8, N9	R4	AS4, AS1, AS2
N10, N11, N12	R4	AS4, AS1, AS3

For example, router R1 now knows that any packet destined for networks N8 or N9 should go through AS1 and AS2 and the next router to deliver the packet to is router R5. Similarly, router R4 knows that any packet destined for networks N10, N11, or N12 should go through AS1 and AS3 and the next router to deliver this packet to is router R1, and so on.

Injection of Information into Intradomain Routing

The role of an interdomain routing protocol such as BGP is to help the routers inside the AS to augment their routing information. In other words, the path tables collected and organized by BPG are not used, per se, for routing packets; they are injected into intradomain forwarding tables (RIP or OSPF) for routing packets. This can be done in several ways depending on the type of AS.

Figure 20.28 *Forwarding tables after injection from BGP*

Table for R1

Des.	Next	Cost
N1	—	1
N4	R4	2
N8	R5	1
N9	R5	1
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R2

Des.	Next	Cost
N1	—	1
N4	R3	2
N8	R1	2
N9	R1	2
N10	R6	1
N11	R6	1
N12	R6	1
N13	R3	3
N14	R3	3
N15	R3	3

Table for R3

Des.	Next	Cost
N1	R2	2
N4	—	1
N8	R2	3
N9	R2	3
N10	R2	2
N11	R2	2
N12	R2	2
N13	R4	2
N14	R4	2
N15	R4	2

Table for R4

Des.	Next	Cost
N1	R1	2
N4	—	1
N8	R1	2
N9	R1	2
N10	R3	3
N11	R3	3
N12	R3	3
N13	R9	1
N14	R9	1
N15	R9	1

Table for R5

Des.	Next	Cost
N8	—	1
N9	—	1
0	R1	1

Table for R6

Des.	Next	Cost
N10	—	1
N11	—	1
N12	R7	2
0	R2	1

Table for R7

Des.	Next	Cost
N10	—	1
N11	R6	2
N12	—	1
0	R6	2

Table for R8

Des.	Next	Cost
N10	R6	2
N11	—	1
N12	—	1
0	R6	2

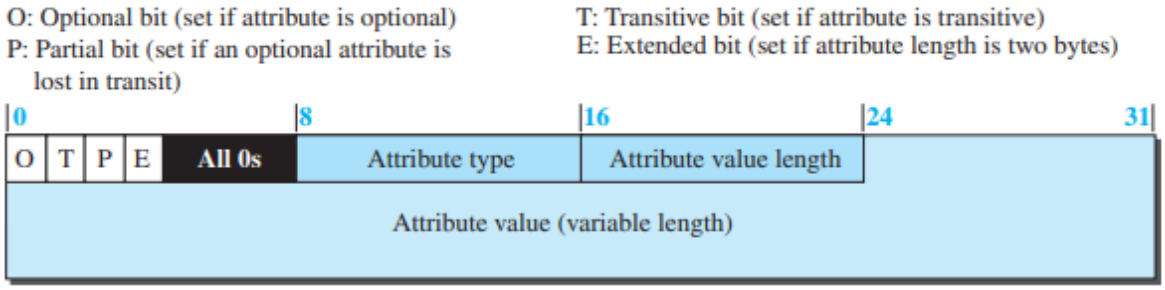
Table for R9

Des.	Next	Cost
N13	—	1
N14	—	1
N15	—	1
0	R4	1

Path Attributes

In both intradomain routing protocols (RIP or OSPF), a destination is normally associated with two pieces of information: next hop and cost. The first one shows the address of the next router to deliver the packet; the second defines the cost to the final destination. Interdomain routing is more involved and naturally needs more information about how to reach the final destination. In BGP these pieces are called path attributes. BGP allows a destination to be associated with up to seven path attributes. Path attributes are divided into two broad categories: **well-known and optional.**

Figure 20.29 *Format of path attribute*



The first byte in each attribute defines the four attribute flags (as shown in the figure). The next byte defines the type of attributes assigned by ICANN (only seven types have been assigned, as explained next). The attribute value length defines the length of the attribute value field (not the length of the whole attributes section). The following gives a brief description of each attribute.

- **ORIGIN (type 1)**. This is a well-known mandatory attribute, which defines the source of the routing information. This attribute can be defined by one of the three values: 1, 2, and 3. Value 1 means that the information about the path has been taken from an intradomain protocol (RIP or OSPF). Value 2 means that the information comes from BGP. Value 3 means that it comes from an unknown source.
- **AS-PATH (type 2).** This is a well-known mandatory attribute, which defines the list of autonomous systems through which the destination can be reached. We have used this attribute in our examples. The AS-PATH attribute, as we discussed in path-vector routing in the last section, helps prevent a loop. Whenever an update message arrives at a router that lists the current AS as the path, the router drops that path. The AS-PATH can also be used in route selection.
- **NEXT-HOP (type 3)**. This is a well-known mandatory attribute, which defines the next router to which the data packet should be forwarded. We have also used this attribute in our examples. As we have seen, this attribute helps to inject path information collected through the operations of eBGP and iBGP into the intradomain routing protocols such as RIP or OSPF.
- **MULT-EXIT-DISC (type 4)**. The multiple-exit discriminator is an optional intransitive attribute, which discriminates among multiple exit paths to a destination. The value of this attribute is normally defined by the metric in the corresponding intradomain protocol (an attribute value of 4-byte unsigned integer). For example, if a router has multiple paths to the destination with different values related to these attributes, the one

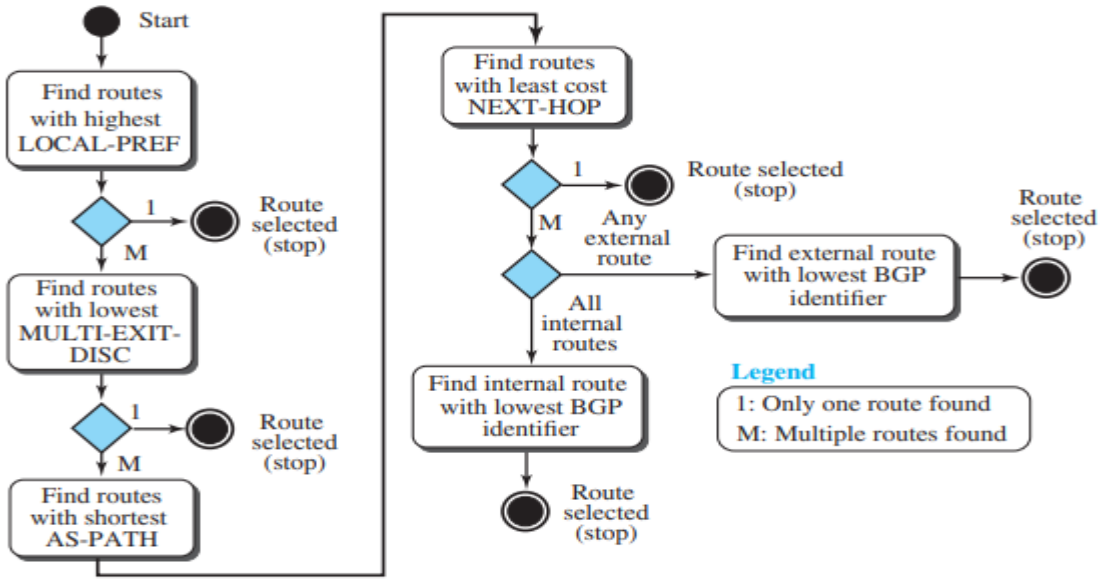
with the lowest value is selected. Note that this attribute is intransitive, which means that it is not propagated from one AS to another.

- **LOCAL-PREF (type 5)**. The local preference attribute is a well-known discretionary attribute. It is normally set by the administrator, based on the organization policy. The routes the administrator prefers are given a higher local preference value (an attribute value of 4-byte unsigned integer). For example, in an internet with five ASs, the administrator of AS1 can set the local preference value of 400 to the path AS1 → AS2 → AS5, the value of 300 to AS1 → AS3 → AS5, and the value of 50 to AS1 → AS4 → AS5. This means that the administrator prefers the first path to the second one and prefers the second one to the third one. This may be a case where AS2 is the most secured and AS4 is the least secured AS for the administration of AS1. The last route should be selected if the other two are not available.
- **ATOMIC-AGGREGATE (type 6)**. This is a well-known discretionary attribute, which defines the destination prefix as not aggregate; it only defines a single destination network. This attribute has no value field, which means the value of the length field is zero.
- **AGGREGATOR (type 7)**. This is an optional transitive attribute, which emphasizes that the destination prefix is an aggregate. The attribute value gives the number of the last AS that did the aggregation followed by the IP address of the router that did so.

Route Selection

So far in this section, we have been silent about how a route is selected by a BGP router mostly because our simple example has one route to a destination. In the case where multiple routes are received to a destination, BGP needs to select one among them. The route selection process in BGP is not as easy as the ones in the intradomain routing protocol that is based on the shortest-path tree. A route in BGP has some attributes attached to it and it may come from an eBGP session or an iBGP session. Figure 20.30 shows the flow diagram as used by common implementations.

Figure 20.30 *Flow diagram for route selection*

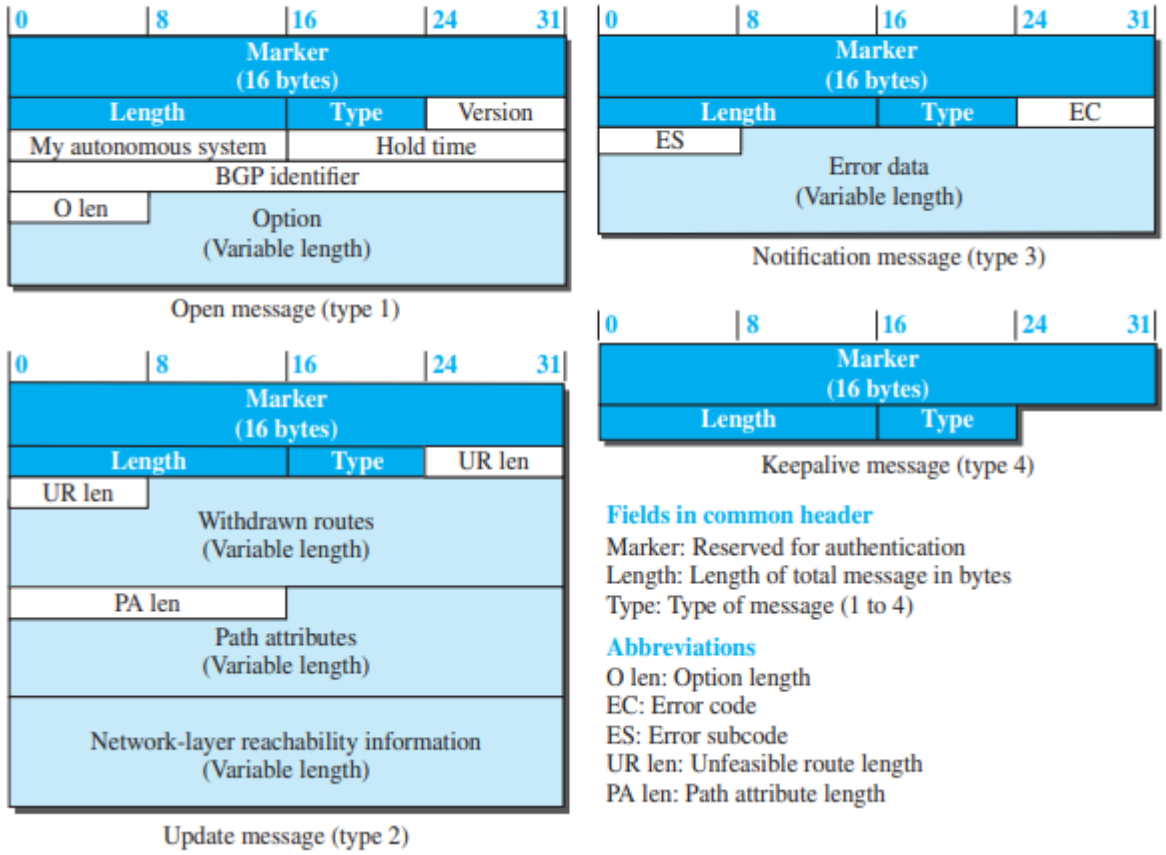


Messages

BGP uses four types of messages for communication between the BGP speakers across the ASs and inside an AS: open, update, keepalive, and notification (see Figure 20.31). 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 the heart of the BGP protocol. It 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 (or multiple destinations with the same path attributes) in a single update message.
- **Keepalive Message**. The BGP peers that are running exchange keepalive 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.

Figure 20.31 *BGP messages*



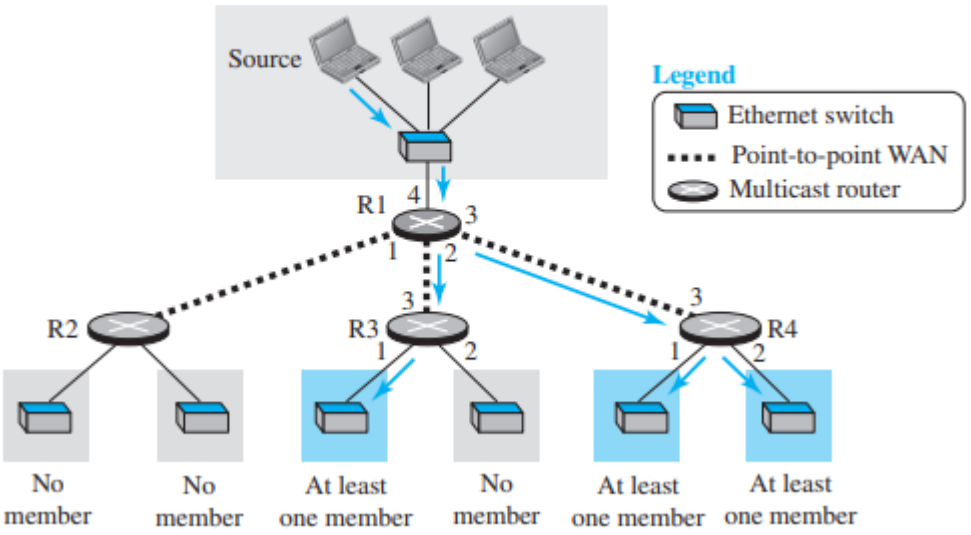
Performance

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 same weakness we mention for RIP about propagation of failure and corruption also exists in BGP.

MULTICASTING BASICS

In multicasting, there is one source and a group of destinations. The relationship is one to many. In this type of communication, the source address is a unicast address, but the destination address is a group address, a group of one or more destination networks in which there is at least one member of the group that is interested in receiving the multicast datagram. The group address defines the members of the group.

Figure 21.2 *Multicasting*



Multicast Addresses

A multicast address defines a group of recipients, not a single one. In other words, a multicast address is an identifier for a group. If a new group is formed with some active members, an authority can assign an unused multicast address to this group to uniquely define it.

This means that the source address of a packet in multicast communication can be a unicast address that uniquely defines the sender, but the destination address can be the multicast address that defines a group. In this way, a host, which is a member of n groups, actually has (n + 1) addresses: one unicast address that is used for source or destination address in unicast communication and n multicast addresses that are used only for destination addresses to receive messages sent to a group.

Multicast addresses in IPv4 belong to a large block of addresses that are specially designed for this purpose. In classful addressing, all of class D was composed of these addresses; classless addressing used the same block, but it was referred to as the block 224.0.0.0/4 (from 224.0.0.0 to 239.255.255.255). Figure 21.5 shows the block in binary. Four bits define the block; the rest of the bits are used as the identifier for the group.

Two Approaches to Multicasting

Two different approaches in multicast routing have been developed: routing using **source-based trees and routing using group-shared trees.**

Source-Based Tree Approach

In the source-based tree approach to multicasting, each router needs to create a separate tree for each source-group combination. In other words, if there are m groups and n sources in the internet, a router needs to create (m × n) routing trees. In each tree, the corresponding

source is the root, the members of the group are the leaves, and the router itself is somewhere on the tree. We can compare the situation with unicast routing in which a router needs only one tree with itself as the root and all networks in the internet as the leaves. Although it may appear that each router needs to create and store a huge amount of information about all of these trees, there are two protocols that use this approach in the Internet today, which we discuss later. These protocols use some strategies that alleviate the situation.

Group-Shared Tree Approach

In the group-shared tree approach, we designate a router to act as the phony source for each group. The designated router, which is called the core router or the rendezvouspoint router, acts as the representative for the group. Any source that has a packet to send to a member of that group sends it to the core center (unicast communication) and the core center is responsible for multicasting. The core center creates one single routing tree with itself as the root and any routers with active members in the group as the leaves. In this approach, there are m core routers (one for each group) and each core router has a routing tree, for the total of m trees. This means that the number of routing trees is reduced from (m × n) in the source-based tree approach to m in this approach. The reader may have noticed that we have divided a multicast delivery from the source to all group members into two deliveries. The first is a unicast delivery from the source to the core router; the second is the delivery from the core router to all group members. Note that the first part of the delivery needs to be done using tunneling. The multicast packet created by the source needs to be encapsulated in a unicast packet and sent to the core router. The core router decapsulates the unicast packet, extracts the multicast packet, and sends it to the group members. Although the reduction in number of trees in this approach looks very attractive, this approach has its own overhead: using an algorithm to select a router among all routers as the core router for a group.

INTRADOMAIN MULTICAST PROTOCOLS

There are three intradomain multicast protocols:

1. Multicast Distance Vector (DVMRP)
2. Multicast Link State (MOSPF)
3. Protocol Independent Multicast (PIM)

Multicast Distance Vector (DVMRP)

The Distance Vector Multicast Routing Protocol (DVMRP) is the extension of the Routing Information Protocol (RIP) which is used in unicast routing. It uses the source based tree approach to multicasting. It is worth mentioning that each router in this protocol that receives a multicast packet to be forwarded implicitly creates a source-based multicast tree in three steps:

1. The router uses an algorithm called **reverse path forwarding (RPF**) to simulate creating part of the optimal source-based tree between the source and itself.
2. The router uses an algorithm called **reverse path broadcasting (RPB**) to create a broadcast (spanning) tree whose root is the router itself and whose leaves are all networks in the internet.
3. The router uses an algorithm called **reverse path multicasting (RPM**) to create a multicast tree by cutting some branches of the tree that end in networks with no member in the group.

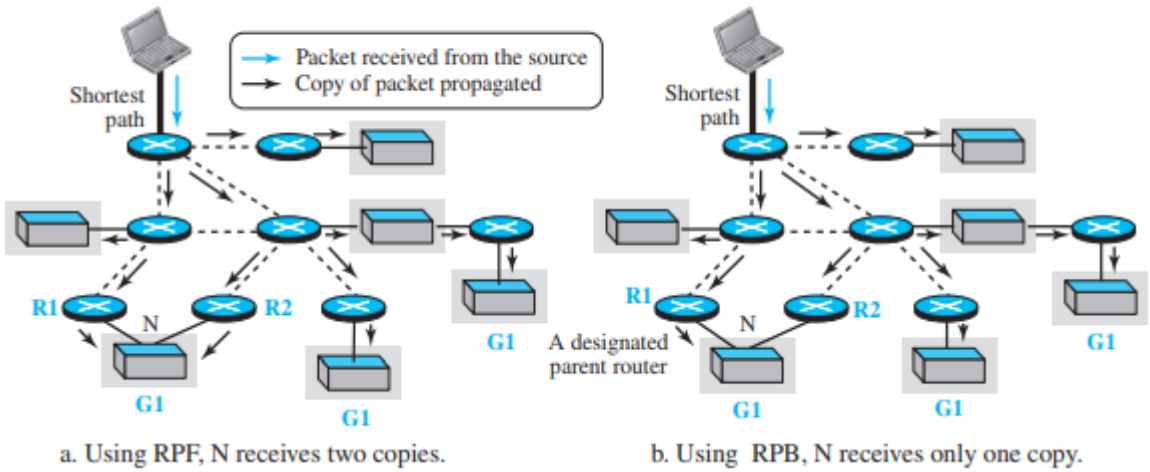
Reverse Path Forwarding (RPF)

The first algorithm, reverse path forwarding (RPF), forces the router to forward a multicast packet from one specific interface: the one which has come through the shortest path from the source to the router. How can a router know which interface is in this path if the router does not have a shortest-path tree rooted at the source? The router uses the first property of the shortest-path tree we discussed in unicast routing, which says that the shortest path from A to B is also the shortest path from B to A. The router does not know the shortest path from the source to itself, but it can find which is the next router in the shortest path from itself to the source (reverse path). The router simply consults its unicast forwarding table, pretending that it wants to send a packet to the source; the forwarding table gives the next router and the interface the message that the packet should be sent out in this reverse direction. The router uses this information to accept a multicast packet only if it arrives from this interface. This is needed to prevent looping. In multicasting, a packet may arrive at the same router that has forwarded it. If the router does not drop all arrived packets except the one, multiple copies of the packet will be circulating in the internet. Of course, the router may add a tag to the packet when it arrives the first time and discard packets that arrive with the same tag, but the RPF strategy is simpler.

Reverse Path Broadcasting (RPB)

Reverse path broadcasting (RPB) actually creates a broadcast tree from the graph that has been created by the RPF algorithm. RPB has cut those branches of the tree that cause cycles in the graph. If we use the shortest path criteria for choosing the parent router, we have actually created a shortest-path broadcast tree. In other words, after this step, we have a shortest-path tree with the source as the root and all networks (LANs) as the leaves. Every packet started from the source reaches all LANs in the internet travelling the shortest path. Figure 21.11 shows how RPB can avoid duplicate reception in a network by assigning a designated parent router, R1, for network N.

Figure 21.11 *RPF versus RPB*



a. Using RPF, N receives two copies.

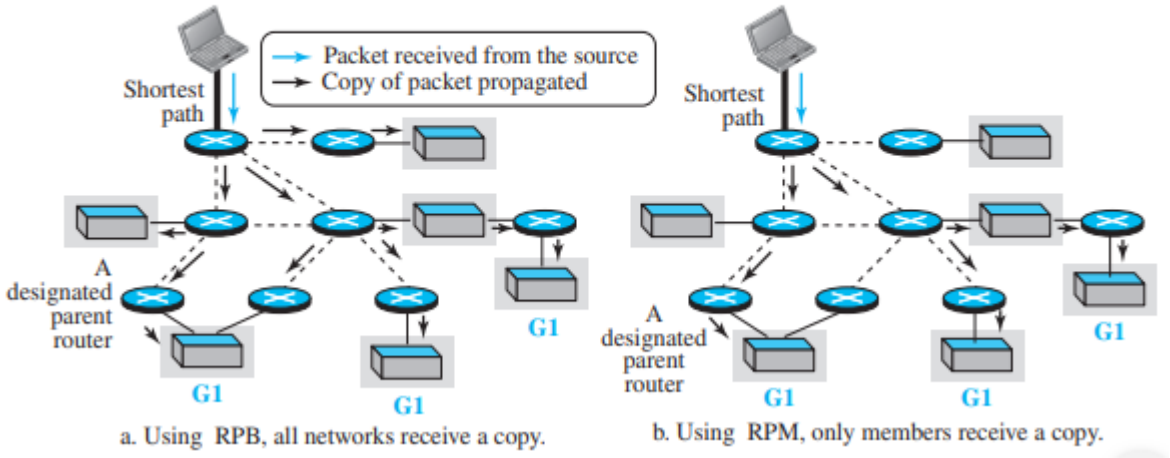
b. Using RPB, N receives only one copy.

Reverse Path Multicasting (RPM)

As you may have noticed, RPB does not multicast the packet, it broadcasts it. This is not efficient. To increase efficiency, the multicast packet must reach only those networks that have

active members for that particular group. This is called reverse path multicasting (RPM). To change the broadcast shortest-path tree to a multicast shortest-path tree, each router needs to prune (make inactive) the interfaces that do not reach a network with active members corresponding to a particular source-group combination. This step can be done bottom-up, from the leaves to the root. At the leaf level, the routers connected to the network collect the membership information using the IGMP protocol discussed before. The parent router of the network can then disseminate this information upward using the reverse shortest-path tree from the router to the source, the same way as the distance vector messages are passed from one neighbor to another. When a router receives all of these membership-related messages, it knows which interfaces need to be pruned. Of course, since these packets are disseminated periodically, if a new member is added to some networks, all routers are informed and can change the status of their interfaces accordingly.

Figure 21.12 *RPB versus RPM*



a. Using RPB, all networks receive a copy.

b. Using RPM, only members receive a copy.

Multicast Link State (MOSPF)

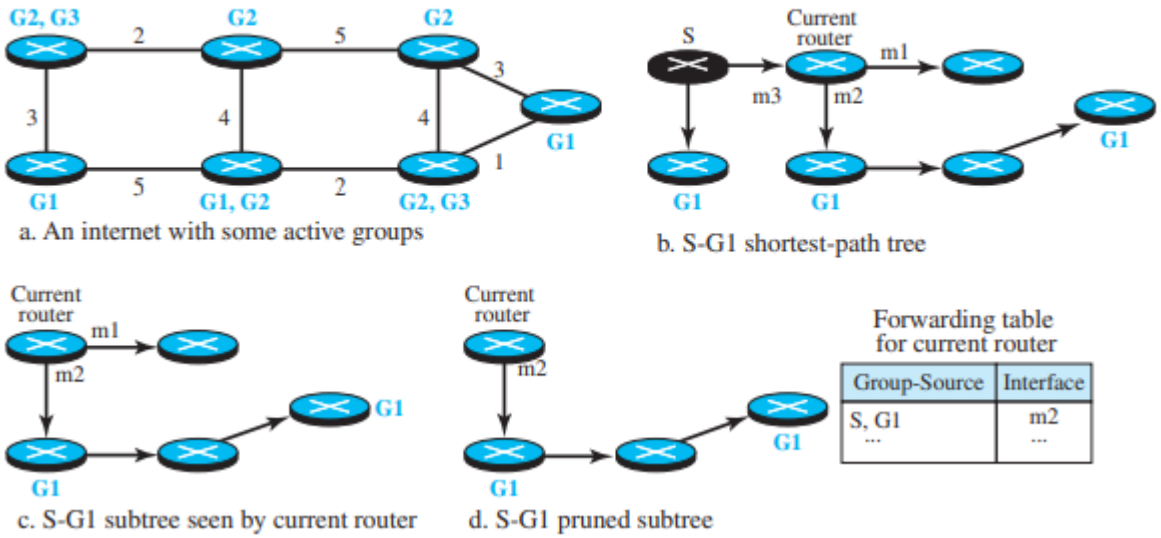
Multicast Open Shortest Path First (MOSPF) is the extension of the Open Shortest Path First (OSPF) protocol, which is used in unicast routing. It also uses the sourcebased tree approach to multicasting. If the internet is running a unicast link-state routing algorithm, the idea can be extended to provide a multicast link-state routing algorithm. Recall that in unicast link-state routing, each router in the internet has a link-state database (LSDB) that can be used to create a shortest-path tree. To extend unicasting to multicasting, each router needs to have another database, as with the case of unicast distance-vector routing, to show which interface has an active member in a particular group. Now a router goes through the following steps to forward a multicast packet received from source S and to be sent to destination G (a group of recipients):

1. The router uses the Dijkstra algorithm to create a shortest-path tree with S as the root and all destinations in the internet as the leaves. Note that this shortest-path tree is different from the one the router normally uses for unicast forwarding, in which the root of the tree is the router itself. In this case, the root of the tree is the source of the packet defined in the source address of the packet. The router is capable of creating this tree because it has the LSDB, the whole topology of the internet; the Dijkstra algorithm can be used to create a tree with any root, no matter which router is using it. The point we

need to remember is that the shortest-path tree created this way depends on the specific source. For each source we need to create a different tree.

2. The router finds itself in the shortest-path tree created in the first step. In other words, the router creates a shortest-path subtree with itself as the root of the subtree.
3. The shortest-path subtree is actually a broadcast subtree with the router as the root and all networks as the leaves. The router now uses a strategy similar to the one we describe in the case of DVMRP to prune the broadcast tree and to change it to a multicast tree. The IGMP protocol is used to find the information at the leaf level. MOSPF has added a new type of link state update packet that floods the membership to all routers. The router can use the information it receives in this way and prune the broadcast tree to make the multicast tree.
4. The router can now forward the received packet out of only those interfaces that correspond to the branches of the multicast tree. We need to make certain that a copy of the multicast packet reaches all networks that have active members of the group and that it does not reach those networks that do not.

Figure 21.13 *Example of tree formation in MOSPF*



Protocol Independent Multicast (PIM)

Protocol Independent Multicast (PIM) is the name given to a common protocol that needs a unicast routing protocol for its operation, but the unicast protocol can be either a distance-vector protocol or a link-state protocol. In other words, PIM needs to use the forwarding table of a unicast routing protocol to find the next router in a path to the destination, but it does not matter how the forwarding table is created. PIM has another interesting feature: it can work in two different modes: dense and sparse. The term dense here means that the number of active members of a group in the internet is large; the probability that a router has a member in a group is high. This may happen, for example, in a popular teleconference that has a lot of members. The term sparse, on the other hand, means that only a few routers in the internet have active members in the group; the probability that a router has a member of the group is low. This may happen, for example, in a very technical teleconference where a number of members are spread somewhere in the internet. When the protocol is working in the dense mode, it is

referred to as PIM-DM; when it is working in the sparse mode, it is referred to as PIMSM. We explain both protocols next.

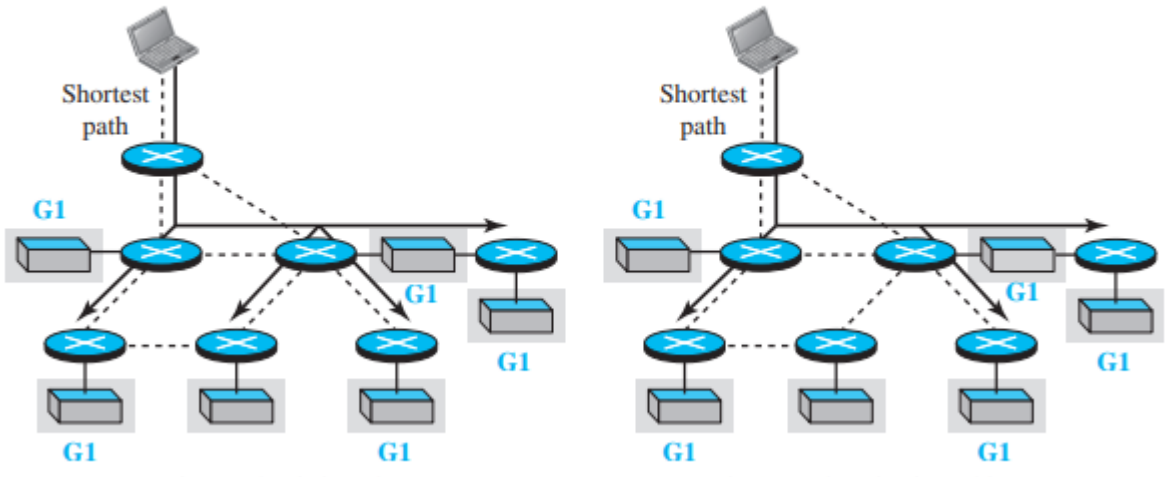
Protocol Independent Multicast-Dense Mode (PIM-DM)

When the number of routers with attached members is large relative to the number of routers in the internet, PIM works in the dense mode and is called PIM-DM. In this mode, the protocol uses a source-based tree approach and is similar to DVMRP, but simpler. PIM-DM uses only two strategies described in DVMRP: RPF and RPM. But unlike DVMRP, forwarding of a packet is not suspended awaiting pruning of the first subtree.

Let us explain the two steps used in PIM-DM to clear the matter.

1. A router that has received a multicast packet from the source S destined for the group G first uses the RPF strategy to avoid receiving a duplicate of the packet. It consults the forwarding table of the underlying unicast protocol to find the next router if it wants to send a message to the source S (in the reverse direction). If the packet has not arrived from the next router in the reverse direction, it drops the packet and sends a prune message in that direction to prevent receiving future packets related to (S, G).
2. If the packet in the first step has arrived from the next router in the reverse direction, the receiving router forwards the packet from all its interfaces except the one from which the packet has arrived and the interface from which it has already received a prune message related to (S, G). Note that this is actually a broadcasting instead of a multicasting if the packet is the first packet from the source S to group G. However, each router downstream that receives an unwanted packet sends a prune message to the router upstream, and eventually the broadcasting is changed to multicasting. Note that DVMRP behaves differently: it requires that the prune messages (which are part of DV packets) arrive and the tree is pruned before sending any message through unpruned interfaces. PIM-DM does not care about this precaution because it assumes that most routers have an interest in the group (the idea of the dense mode)

Figure 21.14 *Idea behind PIM-DM*



a. First packet is broadcast.

b. Second packet is multicast.

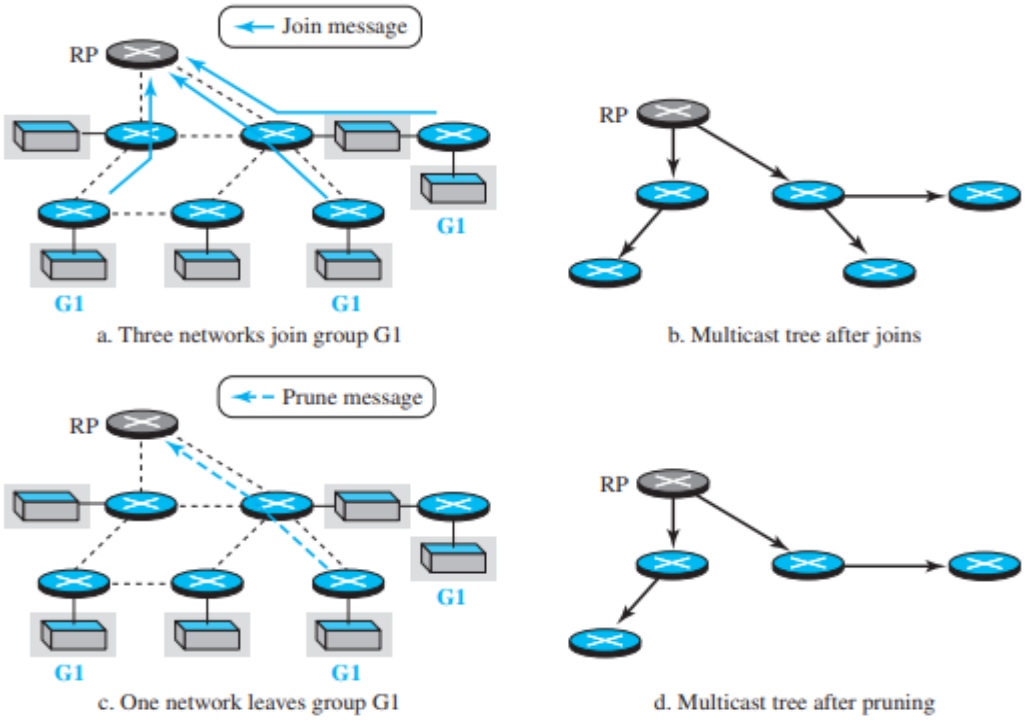
Protocol Independent Multicast-Sparse Mode (PIM-SM)

When the number of routers with attached members is small relative to the number of routers in the internet, PIM works in the sparse mode and is called PIM-SM. In this environment, the use of a protocol that broadcasts the packets until the tree is pruned is not justified; PIM-SM uses a group-shared tree approach to multicasting. The core router in PIM-SM is called the rendezvous point (RP). Multicast communication is achieved in two steps. Any router that has a multicast packet to send to a group of destinations first encapsulates the multicast packet in a unicast packet (tunneling) and sends it to the RP. The RP then decapsulates the unicast packet and sends the multicast packet to its destination. PIM-SM uses a complex algorithm to select one router among all routers in the internet as the RP for a specific group. This means that if we have m active groups, we need m RPs, although a router may serve more than one group. After the RP for each group is selected, each router creates a database and stores the group identifier and the IP address of the RP for tunneling multicast packets to it.

PIM-SM uses a spanning multicast tree rooted at the RP with leaves pointing to designated routers connected to each network with an active member. A very interesting point in PIM-SM is the formation of the multicast tree for a group. The idea is that each router helps to create the tree.

To create a multicast tree rooted at the RP, PIM-SM uses join and prune messages. The join message is used to add possible new branches to the tree; the prune message is used to cut branches that are not needed. When a designated router finds out that a network has a new member in the corresponding group (via IGMP), it sends a join message in a unicast packet destined for the RP. The packet travels through the unicast shortest-path tree to reach the RP. Any router in the path receives and forwards the packet, but at the same time, the router adds two pieces of information to its multicast forwarding table.

Figure 21.15 *Idea behind PIM-SM*



a. Three networks join group G1

b. Multicast tree after joins

c. One network leaves group G1

d. Multicast tree after pruning

INTERDOMAIN MULTICAST PROTOCOLS

One common protocol for interdomain multicast routing is called Multicast Border Gateway Protocol (MBGP), which is the extension of BGP. MBGP provides two paths between ASs: one for unicasting and one for multicasting. Information about multicasting is exchanged between border routers in different ASs. MBGP is a shared-group multicast routing protocol in which one router in each AS is chosen as the rendezvous point (RP).

The problem with MBGP protocol is that it is difficult to inform an RP about the sources of groups in other ASs. The Multicast Source Discovery Protocol (MSDP) is a new suggested protocol that assigns a source representative router in each AS to inform all RPs about the existence of sources in that AS.

Another protocol that is thought of as a possible replacement for the MBGP is Border Gateway Multicast Protocol (BGMP), which allows construction of sharedgroup trees with a single root in one of the ASs.

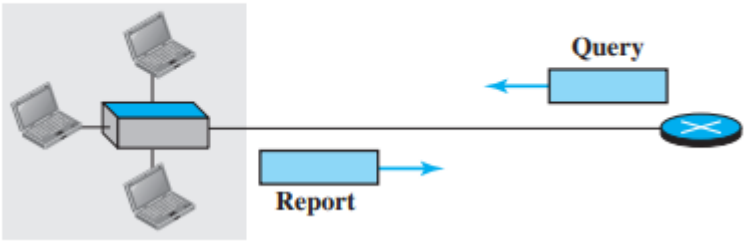
IGMP

The protocol that is used today for collecting information about group membership is the Internet Group Management Protocol (IGMP). IGMP is a protocol defined at the network layer; it is one of the auxiliary protocols, like ICMP, which is considered part of the IP. IGMP messages, like ICMP messages, are encapsulated in an IP datagram.

Messages

There are only two types of messages in IGMP version 3, query and report messages, as shown in Figure 21.16. A query message is periodically sent by a router to all hosts attached to it to ask them to report their interests about membership in groups. A report message is sent by a host as a response to a query message.

Figure 21.16 *IGMP operation*



Query Message

The query message is sent by a router to all hosts in each interface to collect information about their membership. There are three versions of query messages, as described below:

a. A general query message is sent about membership in any group. It is encapsulated in a datagram with the destination address 224.0.0.1 (all hosts and routers). Note that all routers attached to the same network receive this message to inform them that this message is already sent and that they should refrain from resending it.

b. A group-specific query message is sent from a router to ask about the membership related to a specific group. This is sent when a router does not receive a response about a specific group and wants to be sure that there is no active member of that group in the network. The group identifier (multicast

address) is mentioned in the message. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

c. A source-and-group-specific query message is sent from a router to ask about the membership related to a specific group when the message comes from a specific source or sources. Again the message is sent when the router does not hear about a specific group related to a specific host or hosts. The message is encapsulated in a datagram with the destination address set to the corresponding multicast address. Although all hosts receive this message, those not interested drop it.

Report Message

A report message is sent by a host as a response to a query message. The message contains a list of records in which each record gives the identifier of the corresponding group (multicast address) and the addresses of all sources that the host is interested in receiving messages from (inclusion). The record can also mention the source addresses from which the host does not desire to receive a group message (exclusion). The message is encapsulated in a datagram with the multicast address 224.0.0.22 (multicast address assigned to IGMPv3). In IGMPv3, if a host needs to join a group, it waits until it receives a query message and then sends a report message. If a host needs to leave a group, it does not respond to a query message. If no other host responds to the corresponding message, the group is purged from the router database.

Propagation of Membership Information

After a router has collected membership information from the hosts and other routers at its own level in the tree, it can propagate it to the router located in a higher level of the tree. Finally, the router at the tree root can get the membership information to build the multicast tree. The process, however, is more complex than what we can explain in one paragraph. Interested readers can check the book website for the complete description of this protocol.

Encapsulation

The IGMP message is encapsulated in an IP datagram with the value of the protocol field set to 2 and the TTL field set to 1. The destination IP address of the datagram, however, depends on the type of message, as shown in Table 21.1.

Table 21.1 *Destination IP Addresses*

Message Type	*IP Address*
General Query	224.0.0.1
Other Queries	Group address
Report	224.0.0.22

IPv6 Addressing

The main reason for migration from IPv4 to IPv6 is the small size of the address space in IPv4. In this section, we show how the huge address space of IPv6 prevents address depletion in the future. An IPv6 address is 128 bits or 16 bytes (octets) long, four times the address length in IPv4.

Representation

A computer normally stores the address in binary, but it is clear that 128 bits cannot easily be handled by humans. Several notations have been proposed to represent IPv6 addresses

when they are handled by humans. The following shows two of these notations: binary and colon hexadecimal.

Binary (128 bits)	**1111111011110110**	**...**	**1111111100000000**
Colon Hexadecimal	**FEF6:BA98:7654:3210:ADEF:BBFF:2922:FF00**		

Binary notation is used when the addresses are stored in a computer. The colon hexadecimal notation (or colon hex for short) divides the address into eight sections, each made of four hexadecimal digits separated by colons.

Address Space

The address space of IPv6 contains 2^{128} addresses. This address space is 2^{96} times the IPv4 address—definitely no address depletion—as shown, the size of the space is To give some idea about the number of addresses, we assume that only 1/64 (almost 2 percent) of the addresses in the space can be assigned to the people on planet Earth and the rest are reserved for special purposes. We also assume that the number of people on the earth is soon to be 2^{34} (more than 16 billion). Each person can have 2^{88} addresses to use. Address depletion in this version is impossible.

Three Address Types

In IPv6, a destination address can belong to one of three categories: unicast, anycast, and multicast.

Unicast Address

A unicast address defines a single interface (computer or router). The packet sent to a unicast address will be routed to the intended recipient.

Anycast Address

An anycast address defines a group of computers that all share a single address. A packet with an anycast address is delivered to only one member of the group, the most reachable one. An anycast communication is used, for example, when there are several servers that can respond to an inquiry. The request is sent to the one that is most reachable. The hardware and software generate only one copy of the request; the copy reaches only one of the servers. IPv6 does not designate a block for anycasting; the addresses are assigned from the unicast block.

Multicast Address

A multicast address also defines a group of computers. However, there is a difference between anycasting and multicasting. In anycasting, only one copy of the packet is sent to one of the members of the group; in multicasting each member of the group receives a copy. As we will see shortly, IPv6 has designated a block for multicasting from which the same address is assigned to the members of the group. It is interesting that IPv6 does not define broadcasting, even in a limited version. IPv6 considers broadcasting as a special case of multicasting.

Address Space Allocation

Like the address space of IPv4, the address space of IPv6 is divided into several blocks of varying size and each block is allocated for a special purpose. Most of the blocks are still unassigned and have been set aside for future use. Table 22.1 shows only the assigned blocks. In this table, the last column shows the fraction each block occupies in the whole address space.

Table 22.1 *Prefixes for assigned IPv6 addresses*

Block prefix	*CIDR*	*Block assignment*	*Fraction*
0000 0000	0000::/8	Special addresses	1/256
001	**2000::/3**	**Global unicast**	**1/8**
1111 110	FC00::/7	Unique local unicast	1/128
1111 1110 10	FE80::/10	Link local addresses	1/1024
1111 1111	FF00::/8	Multicast addresses	1/256

Autoconfiguration

One of the interesting features of IPv6 addressing is the autoconfiguration of hosts. As we discussed in IPv4, the host and routers are originally configured manually by the network manager. However, the Dynamic Host Configuration Protocol, DHCP, can be used to allocate an IPv4 address to a host that joins the network. In IPv6, DHCP protocol can still be used to allocate an IPv6 address to a host, but a host can also configure itself.

When a host in IPv6 joins a network, it can configure itself using the following process:

1. The host first creates a link local address for itself. This is done by taking the 10-bit link local prefix (1111 1110 10), adding 54 zeros, and adding the 64-bit interface identifier, which any host knows how to generate from its interface card. The result is a 128-bit link local address.
2. The host then tests to see if this link local address is unique and not used by other hosts. Since the 64-bit interface identifier is supposed to be unique, the link local address generated is unique with a high probability. However, to be sure, the host sends a neighbor solicitation message and waits for a neighbor advertisement message. If any host in the subnet is using this link local address, the process fails and the host cannot autoconfigure itself; it needs to use other means such as DHCP for this purpose.
3. If the uniqueness of the link local address is passed, the host stores this address as its link local address (for private communication), but it still needs a global unicast address. The host then sends a router solicitation message (discussed later in the chapter) to a local router. If there is a router running on the network, the host receives a router advertisement message that includes the global unicast prefix and the subnet prefix that the host needs to add to its interface identifier to generate its global unicast address. If the router cannot help the host with the configuration, it informs the host in the router advertisement message (by setting a flag). The host then needs to use other means for configuration.

Renumbering

To allow sites to change the service provider, renumbering of the address prefix (n) was built into IPv6 addressing. As we discussed before, each site is given a prefix by the service provider to which it is connected. If the site changes the provider, the address prefix needs to be changed. A router to which the site is connected can advertise a new prefix and let the site use the old prefix for a short time before disabling it. In other words, during the transition period, a site has two prefixes. The main problem in using the renumbering mechanism is the support of the DNS, which needs to propagate the new addressing associated with a domain name. A new protocol for DNS, called Next Generation DNS, is under study to provide support for this mechanism.

IPv6 PROTOCOL

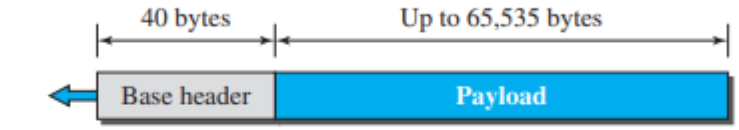
The change of the IPv6 address size requires the change in the IPv4 packet format. The designer of IPv6 decided to implement remedies for other shortcomings now that a change is inevitable. The following shows other changes implemented in the protocol in addition to changing address size and format.

- **Better header format**. IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
- **New options**. IPv6 has new options to allow for additional functionalities.
- **Allowance for extension**. IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- **Support for resource allocation**. In IPv6, the type-of-service field has been removed, but two new fields, traffic class and flow label, have been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- **Support for more security**. The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

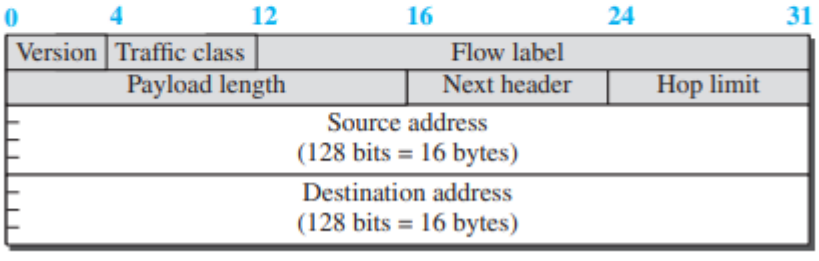
Packet Format

The IPv6 packet is shown in Figure 22.6. Each packet is composed of a base header followed by the payload. The base header occupies 40 bytes, whereas payload can be up to 65,535 bytes of information. The description of fields follows.

Figure 22.6 *IPv6 datagram*



a. IPv6 packet

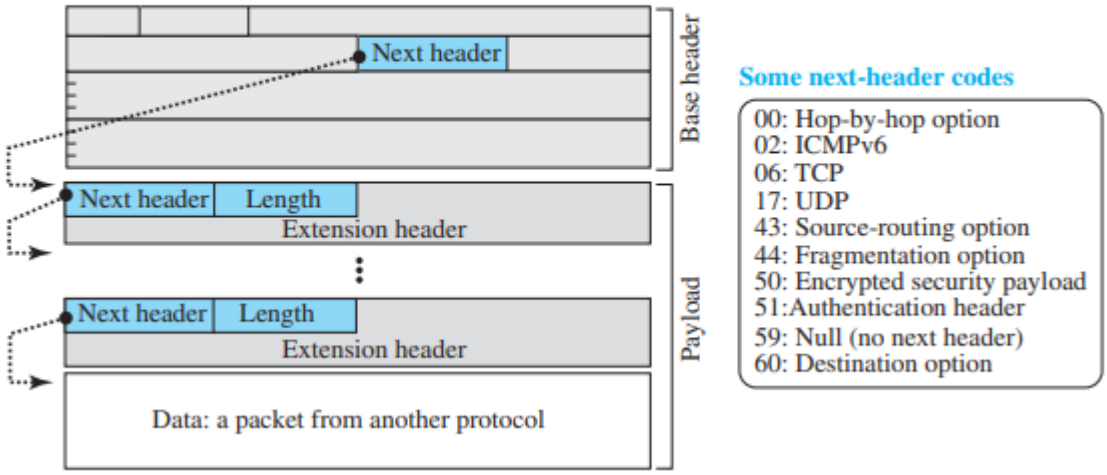


b. Base header

- **Version**. The 4-bit version field defines the version number of the IP. For IPv6, the value is 6.
- **Traffic class**. The 8-bit traffic class field is used to distinguish different payloads with different delivery requirements. It replaces the type-of-service field in IPv4.

- **Flow label**. The flow label is a 20-bit field that is designed to provide special handling for a particular flow of data. We will discuss this field later.
- **Payload length.** The 2-byte payload length field defines the length of the IP datagram excluding the header. Note that IPv4 defines two fields related to the length: header length and total length. In IPv6, the length of the base header is fixed (40 bytes); only the length of the payload needs to be defined.
- **Next header**. The next header is an 8-bit field defining the type of the first extension header (if present) or the type of the data that follows the base header in the datagram. This field is similar to the protocol field in IPv4, but we talk more about it when we discuss the payload.
- **Hop limit**. The 8-bit hop limit field serves the same purpose as the TTL field in IPv4.
- **Source and destination addresses.** The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram. The destination address field is a 16-byte (128-bit) Internet address that identifies the destination of the datagram.
- **Payload**. Compared to IPv4, the payload field in IPv6 has a different format and meaning, as shown in Figure 22.7.

Figure 22.7 *Payload in an IPv6 datagram*



The payload in IPv6 means a combination of zero or more extension headers (options) followed by the data from other protocols (UDP, TCP, and so on). In IPv6, options, which are part of the header in IPv4, are designed as extension headers. The payload can have as many extension headers as required by the situation. Each extension header has two mandatory fields, next header and the length, followed by information related to the particular option. Note that each next header field value (code) defines the type of the next header (hop-by-hop option, sourcerouting option, . . .); the last next header field defines the protocol (UDP, TCP, . . .) that is carried by the datagram.

Fragmentation and Reassembly

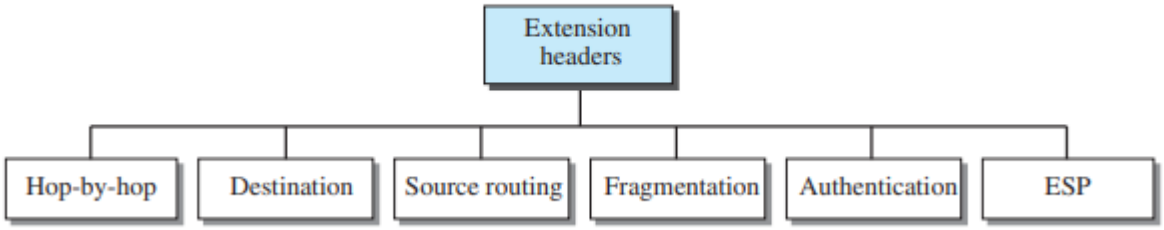
There are still fragmentation and reassembly of datagrams in the IPv6 protocol, but there is a major difference in this respect. IPv6 datagrams can be fragmented only by the source, not

by the routers; the reassembly takes place at the destination. The fragmentation of packets at routers is not allowed to speed up the processing of packets in the router. The fragmentation of a packet in a router needs a lot of processing. The packet needs to be fragmented, all fields related to the fragmentation need to be recalculated. In IPv6, the source can check the size of the packet and make the decision to fragment the packet or not. When a router receives the packet, it can check the size of the packet and drop it if the size is larger than allowed by the MTU of the network ahead. The router then sends a packet-too-big ICMPv6 error message (discussed later) to inform the source.

Extension Header

An IPv6 packet is made of a base header and some extension headers. The length of the base header is fixed at 40 bytes. However, to give more functionality to the IP datagram, the base header can be followed by up to six extension headers. Many of these headers are options in IPv4. Six types of extension headers have been defined. These are hop-by-hop option, source routing, fragmentation, authentication, encrypted security payload, and destination option (see Figure 22.8).

Figure 22.8 *Extension header types*



Hop-by-Hop Option

The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram. For example, perhaps routers must be informed about certain management, debugging, or control functions. Or, if the length of the datagram is more than the usual 65,535 bytes, routers must have this information. So far, only three hopby-hop options have been defined: **Pad1, PadN, and jumbo payload**.

- **Pad1.** This option is 1 byte long and is designed for alignment purposes. Some options need to start at a specific bit of the 32-bit word. If an option falls short of this requirement by exactly one byte, Pad1 is added.
- **PadN.** PadN is similar in concept to Pad1. The difference is that PadN is used when 2 or more bytes are needed for alignment.
- **Jumbo payload**. Recall that the length of the payload in the IP datagram can be a maximum of 65,535 bytes. However, if for any reason a longer payload is required, we can use the jumbo payload option to define this longer length.

Destination Option

The destination option is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information. The format of the destination option is the same as the hop-by-hop option. So far, only the Pad1 and PadN options have been defined.

Source Routing

The source routing extension header combines the concepts of the strict source route and the loose source route options of IPv4.

Fragmentation

The concept of fragmentation in IPv6 is the same as that in IPv4. However, the place where fragmentation occurs differs. In IPv4, the source or a router is required to fragment if the size of the datagram is larger than the MTU of the network over which the datagram travels. In IPv6, only the original source can fragment. A source must use a Path MTU Discovery technique to find the smallest MTU supported by any network on the path. The source then fragments using this knowledge.

If the source does not use a Path MTU Discovery technique, it fragments the datagram to a size of 1280 bytes or smaller. This is the minimum size of MTU required for each network connected to the Internet.

Authentication

The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data. The former is needed so the receiver can be sure that a message is from the genuine sender and not from an imposter. The latter is needed to check that the data is not altered in transition by some hacker.

Encrypted Security Payload

The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping.

Comparison of Options between IPv4 and IPv6

The following shows a quick comparison between the options used in IPv4 and the options used in IPv6 (as extension headers).

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