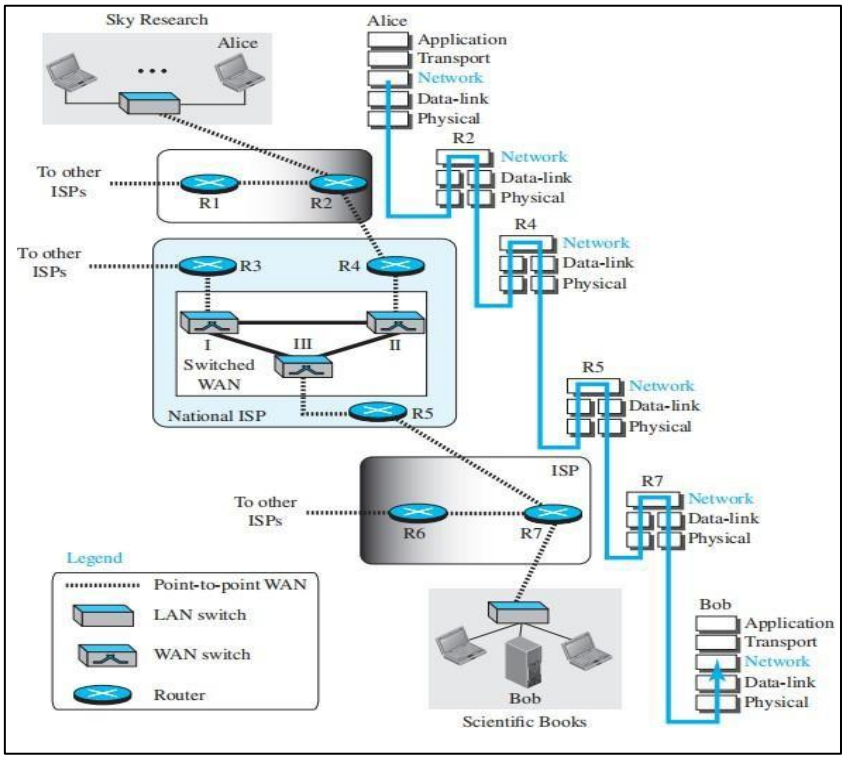
Module 3

Network Layer

Introduction to network layer

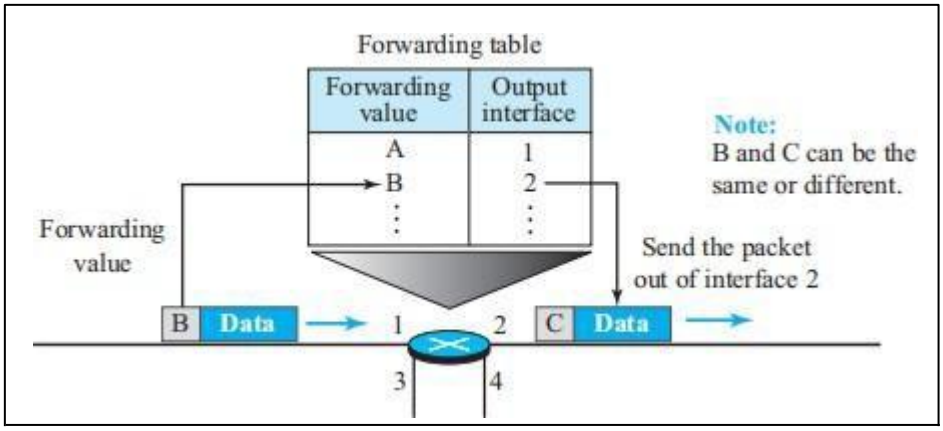
The network layer in the TCP/IP protocol suite is responsible for the host-to-host delivery of datagrams. It provides services to the transport layer and receives services from the data-link layer.



The following are the services provided by the network layer

1. **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.
2. ***Routing***
   The network layer is responsible for routing the packet from its source to the destination. There will be more than one route from the source to the destination. The network layer is responsible for finding the best one among these possible routes.

3. ***Forwarding***
   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*.

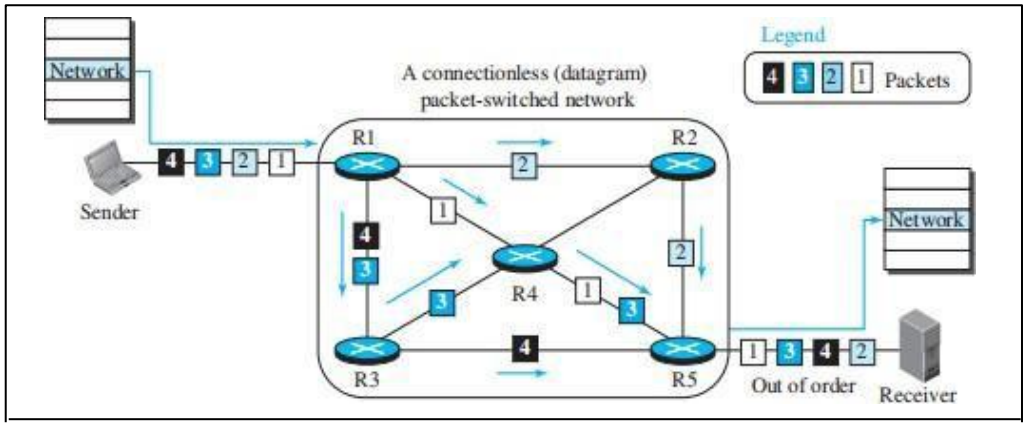


Other Services

1. ***Error Control***
2. ***Flow Control***
3. ***Congestion Control***
   Congestion in the network layer is a situation in which too many datagrams are present in an area of the Internet. Congestion may occur if the number of datagrams sent by source computers is beyond the capacity of the network or routers.
4. ***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.
5. ***Security***

PACKET SWITCHING

When the Internet was first designed, the network layer provided a connectionless service. Each packet is treated independently, with no relationship to other packets. The network layer's role is to deliver packets from the source to the destination, and packets may take different paths to get there.



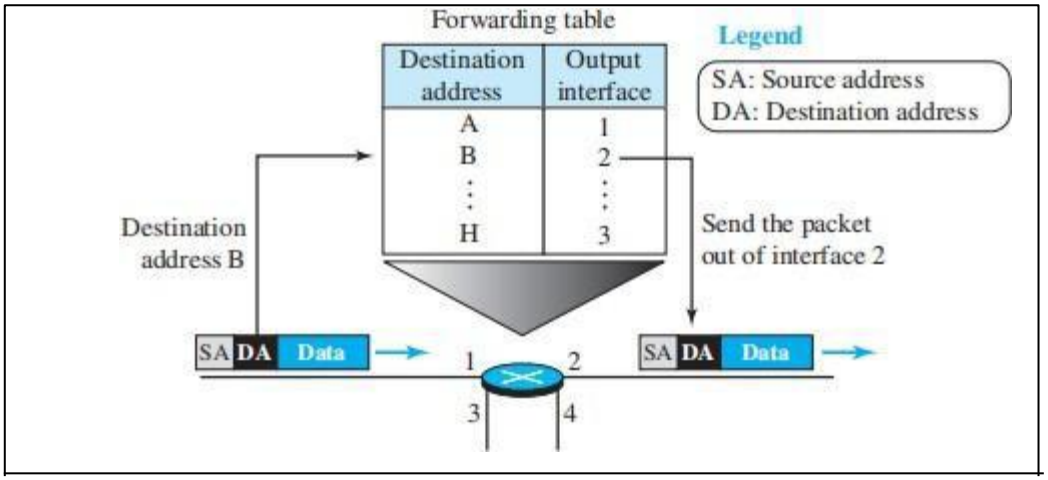
At the network layer, routers act like switches, connecting an input port to one or more output ports, similar to how an electrical switch works. In data communication, there are two main switching types: circuit switching and packet switching. Circuit switching is mostly used at the physical layer, but at the network layer, only packet switching is used because data is sent as packets.

When a message is sent from one device to another, it is divided into packets. The source sends these packets one by one, and the destination waits for all packets to arrive before reconstructing the message. To send the packets, the network uses two routing methods: the **datagram approach** and the **virtual circuit approach**.

.

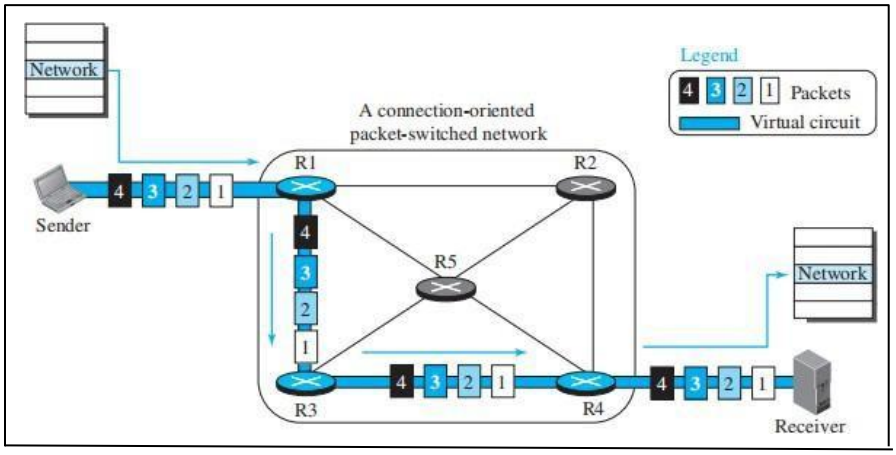
Datagram Approach: Connectionless Service

- ➔ When the network layer provides a connectionless service, each packet traveling in the Internet is an independent entity; there is no relationship between packets belonging to the same message.
- ➔ The switches in this type of network are called **routers**.
- ➔ A packet belonging to a message may be followed by a packet belonging to the same message or to a different message.
- ➔ A packet may be followed by a packet coming from the same or from a different source.
- ➔ Each packet is routed based on the information contained in its header: **source and destination addresses**. The destination address defines where it should go; the source address defines where it comes from. T
- ➔ he router in this case routes the packet based only on the destination address. The source address may be used to send an error message to the source if the packet is discarded.
- ➔ Figure shows the forwarding process in a router in this case. We have used symbolic addresses such as A and B.

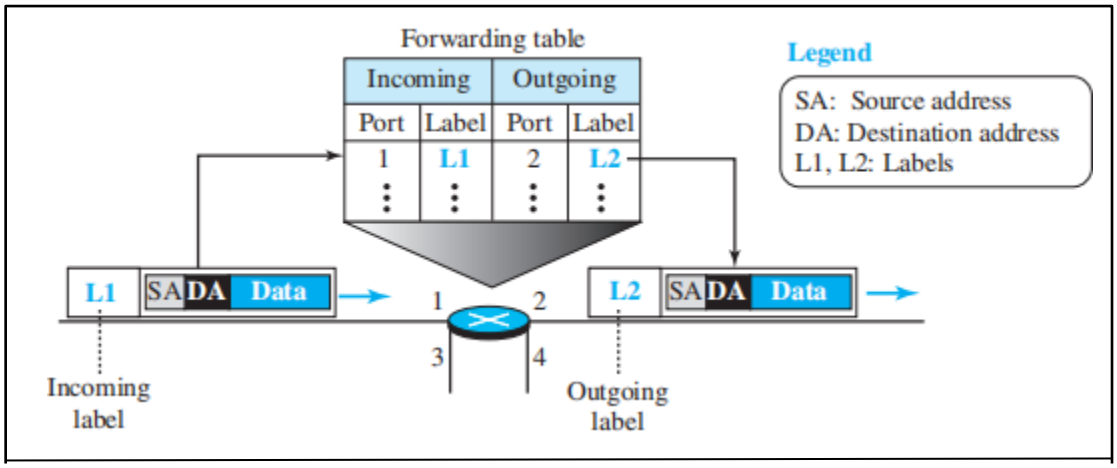


Virtual-Circuit Approach: Connection-Oriented Service

- ➔ In a connection-oriented service (also called virtual-circuit approach), there is a relationship between all packets belonging to a message.
- ➔ Before all datagrams in a message can be sent, a virtual connection should be set up to define the path for the datagrams.
- ➔ After connection setup, the datagrams can all follow the same path.
- ➔ In this type of service, not only must the packet contain the source and destination addresses, it must also contain a **flow label**, a virtual circuit identifier that defines the virtual path the packet should follow.



Each packet is forwarded based on a label in the packet. In a connection-oriented design, we assume the packet has a label when it reaches the router. The router uses this label, also called a virtual circuit identifier, to decide where to send the packet next.



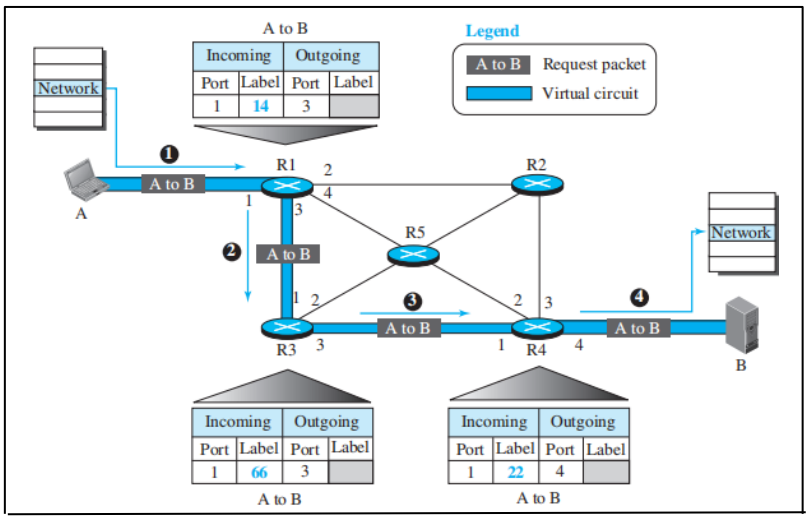
To create a connection-oriented service, a three-phase process is used: setup, data transfer, and teardown.

Setup Phase

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

Request packet

A request packet is sent from the source to the destination. This auxiliary packet carries the source and destination addresses.



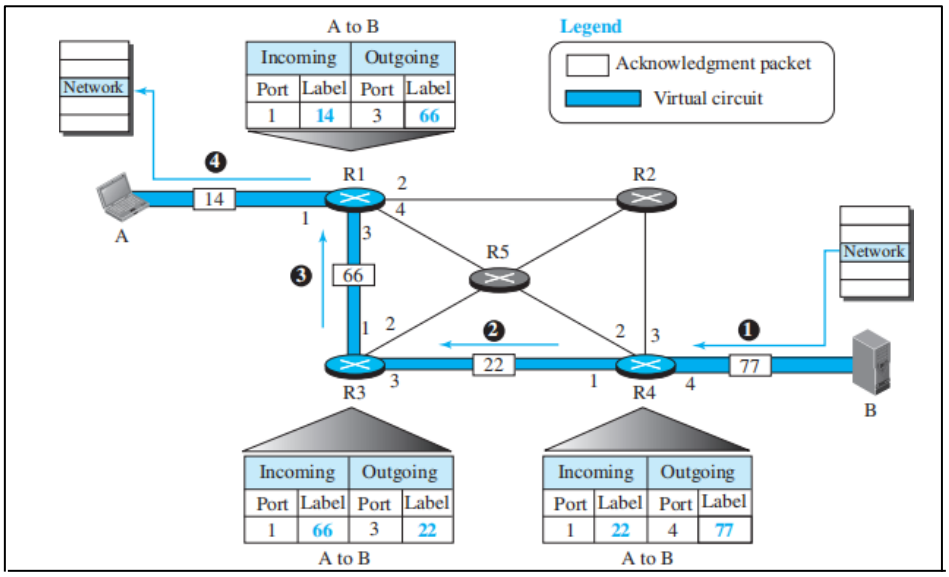
1. Source A sends a request packet to router R1.
2. Router R1 receives the request packet. It knows that a packet going from A to B goes out through port 3. How the router has obtained this information is a point covered later. For the moment, assume that it knows the output port. The router creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The router assigns the incoming port (1) and chooses an avail- able incoming label (14) and the

outgoing port (3). It does not yet know the outgo- ing label, which will be found during the acknowledgment step. The router then forwards the packet through port 3 to router R3.

3. Router R3 receives the setup request packet. The same events happen here as at router R1; three columns of the table are completed: in this case, incoming port (1), incoming label (66), and outgoing port (3).
4. Router R4 receives the setup request packet. Again, three columns are completed: incoming port (1), incoming label (22), and outgoing port (4).
5. Destination B receives the setup packet, and if it is ready to receive packets from A, it assigns a label to the incoming packets that come from A, in this case 77. This label lets the destination know that the packets come from A, and not from other sources.

Acknowledgment Packet

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



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

4. Finally, router R1 sends an acknowledgment to source A that contains its incoming label in the table, chosen in the setup phase.
5. The source uses this as the outgoing label for the data packets to be sent to destination B.

Data-Transfer Phase
The second phase is called the data-transfer phase. After all routers have created their forwarding table for a specific virtual circuit, then the network-layer packets belonging to one message can be sent one after another.

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

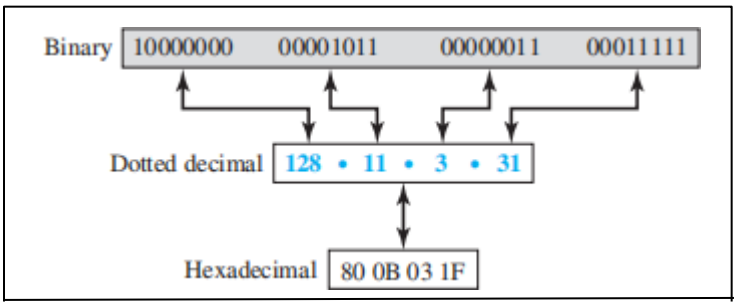
IPV4 ADDRESSES

The identifier used in the IP layer of the TCP/IP protocol suite to identify the connection of each device to the Internet is called the Internet address or IP address. An IPv4 address is a 32-bit address that uniquely and universally defines the connection of a host or a router to the Internet. The IP address is the address of the connection, not the host or the router, because if the device is moved to another network, the IP address may be changed.
IPv4 addresses are unique in the sense that each address defines one, and only one, connection to the Internet. If a device has two connections to the Internet, via two networks, it has two IPv4 addresses. IPv4 addresses are universal in the sense that the addressing system must be accepted by any host that wants to be connected to the Internet.

Address Space
A protocol like IPv4 that defines addresses has an address space. An address space is the total number of addresses used by the protocol. If a protocol uses b bits to define an address, the address space is 2^b because each bit can have two different values (0 or 1).
IPv4 uses 32-bit addresses, which means that the address space is 2^{32} or 4,294,967,296 (more than four billion). If there were no restrictions, more than 4 billion devices could be connected to the Internet.

Notation
There are three common notations to show an IPv4 address: binary notation (base 2), dotted-decimal notation (base 256), and hexadecimal notation (base 16). In binary notation, an IPv4 address is displayed as 32 bits.
To make the address more readable, one or more spaces are usually inserted between each octet (8 bits). Each octet is often referred to as a byte. To make the IPv4 address more compact and easier to read, it is usually written in decimal form with a decimal point (dot) separating the bytes. This format is referred to as dotted-decimal notation. Note that because each byte (octet) is only 8 bits, each number in the dotted-decimal notation is between 0 and 255. We sometimes see

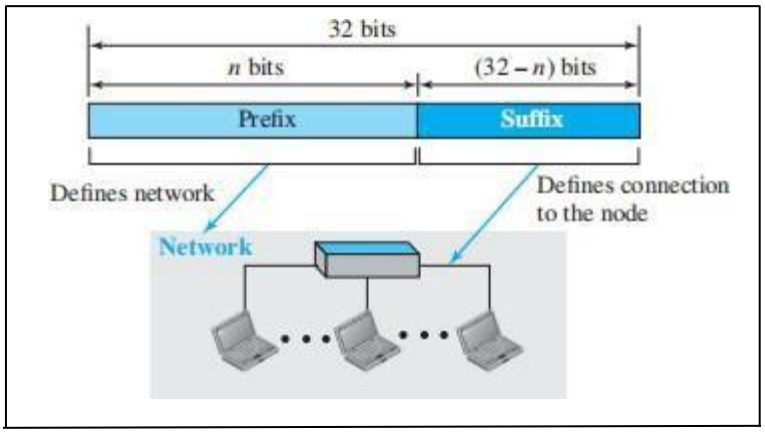
an IPv4 address in hexadecimal notation. Each hexadecimal digit is equivalent to four bits. This means that a 32-bit address has 8 hexadecimal digits. This notation is often used in network programming.



Hierarchy in Addressing

In any communication network that involves delivery, such as a telephone network or a postal network, the addressing system is hierarchical. In a postal network, the postal address (mailing address) includes the country, state, city, street, house number, and the name of the mail recipient. Similarly, a telephone number is divided into the country code, area code, local exchange, and the connection.

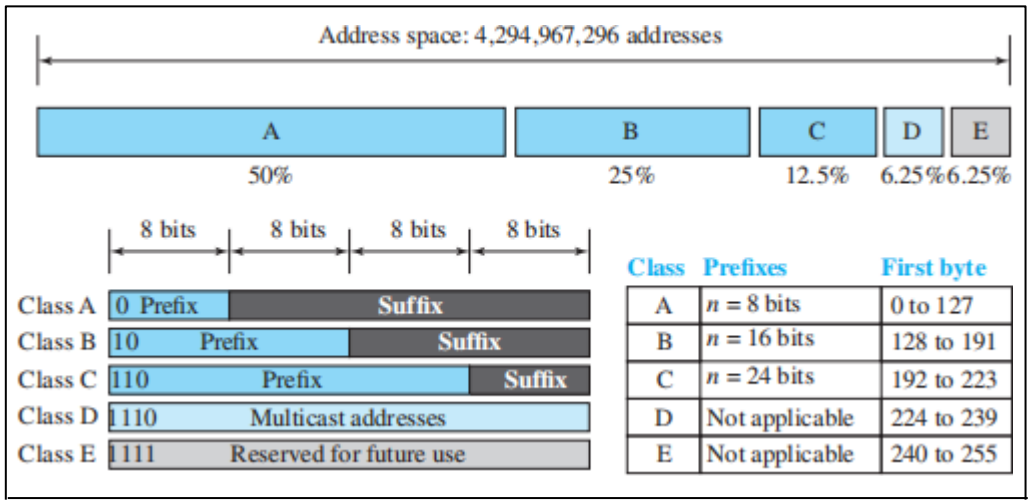
A 32-bit IPv4 address is also hierarchical, but divided only into two parts. The first part of the address, called the *prefix*, defines the network; the second part of the address, called the *suffix*, defines the node (connection of a device to the Internet). Figure shows the prefix and suffix of a 32-bit IPv4 address. The prefix length is n bits and the suffix length is $(32 - n)$ bits.



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

Classful Addressing

- ➔ When the Internet started, an IPv4 address was designed with a fixed-length prefix, but to accommodate both small and large networks, three fixed-length prefixes were designed instead of one (n = 8, n = 16, and n = 24). The whole address space was divided into five classes (class A, B, C, D, and E). This scheme is referred to as **classful addressing**.
- ➔ In class A, the network length is 8 bits, but since the first bit, which is 0, defines the class, we can have only seven bits as the network identifier. This means there are only $2^7 = 128$ networks in the world that can have a class A address.
- ➔ In class B, the network length is 16 bits, but since the first two bits, which are $(10)_2$, define the class, we can have only 14 bits as the network identifier. This means there are only 214 = 16,384 networks in the world that can have a class B address.
- ➔ All addresses that start with $(110)_2$ belong to class C. In class C, the network length is 24 bits, but since three bits define the class, we can have only 21 bits as the network identifier. This means there are $2^{21} = 2{,}097{,}152$ networks in the world that can have a class C address.



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

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

Address Depletion

Classful addressing became obsolete due to address depletion. The inefficient distribution of addresses led to a rapid shortage. For example, Class A, which could only be assigned to 128 organizations, provided more addresses (16.7 million per network) than most needed, resulting in wasted space. Class B, for midsize organizations, also had many unused addresses. Class C's flaw was offering too few addresses (256 per network), making it impractical for many companies. Class E was rarely used, leading to further waste.

Subnetting and Supernetting

Two strategies were proposed to address depletion: **subnetting** and **supernetting**. Subnetting divides large blocks (Class A or B) into smaller subnets with longer prefixes, but it failed because large organizations didn't want to share unused addresses. Supernetting combined multiple Class C blocks to create larger blocks for organizations needing more than 256 addresses, but this made packet routing more complex and also proved ineffective.

Advantage of Classful Addressing

Although classful addressing had several problems and became obsolete, it had one advantage: Given an address, we can easily find the class of the address and, since the prefix length for each class is fixed, we can find the prefix length immediately.

Classless Addressing

Subnetting and supernetting didn't fully solve address depletion. As the Internet grew, it became clear that a larger address space was needed, leading to the development of **IPv6** as a long-term solution. However, a short-term fix was **classless addressing**, which kept IPv4 but removed class boundaries to distribute addresses more fairly.

ISPs played a role in this change, providing Internet access by subdividing large address ranges and assigning smaller blocks (1, 2, 4, etc.) to customers. **Classless addressing**, introduced in 1996, allows for variable-length blocks of addresses (e.g., 1, 2, 4, or 128). The prefix of the address identifies the network, while the suffix identifies the device. The number of addresses in each block must be a power of 2, and the blocks are non-overlapping.

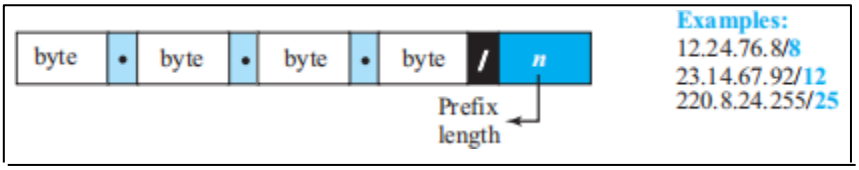
In classless addressing, the prefix length is variable, ranging from 0 to 32. A shorter prefix indicates a larger network, while a longer prefix means a smaller network. Classless addressing can be applied to classful addressing by treating Class A as having a prefix of 8, Class B with a prefix of 16, and so on.

Prefix Length and Slash Notation (CIDR)

In classless addressing, the prefix length (n) is added to the address, separated by a slash, known as slash notation or CIDR (Classless Inter-Domain Routing). For example, in 167.199.170.82/27, "27" is the prefix length. This notation is necessary since the address alone does not define the block.

To extract information from a classless address:

Number of addresses = 2^(32 - n)
First address: Keep the first n bits, set the rest to 0.
Last address: Keep the first n bits, set the rest to 1.
For 167.199.170.82/27, there are 32 addresses in the block.



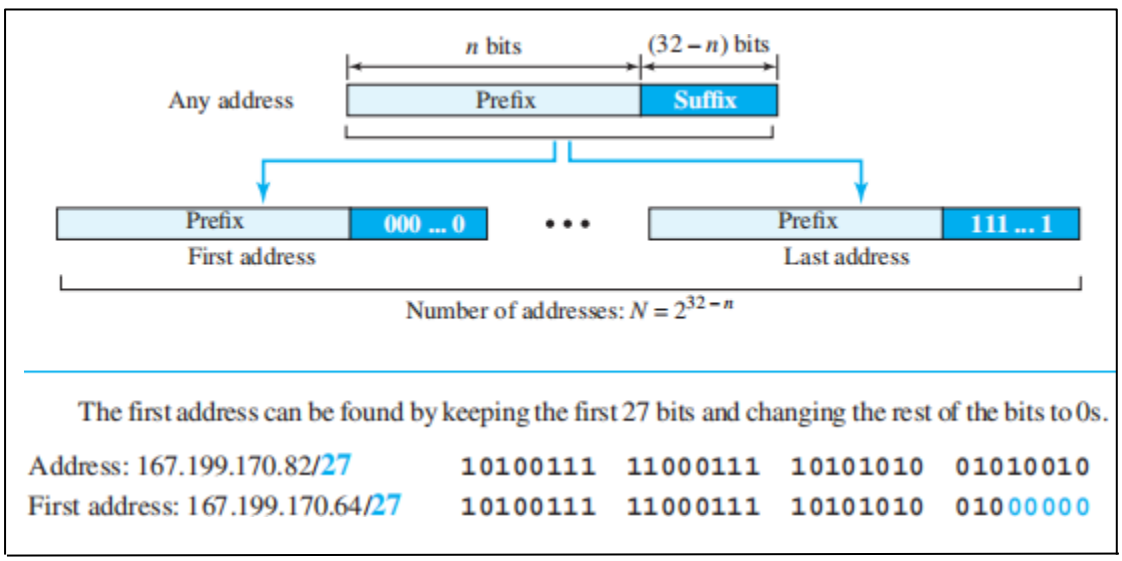
Extracting Information from an Address

Given any address in the block, we normally like to know three pieces of information about the block to which the address belongs: the number of addresses, the first address in the block, and the last address. Since the value of prefix length, *n*, is given, we can easily find these three pieces of information, as shown in Figure 18.21.

1. The number of addresses in the block is found as $N = 232 - n$.
2. To find the first address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 0s.
3. To find the last address, we keep the *n* leftmost bits and set the (32 − *n*) rightmost bits all to 1s.

Example 1

A classless address is given as 167.199.170.82/27. We can find the above three pieces of information as follows. The number of addresses in the network is 232 − n $= 2^5 = 32$ addresses.



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 (232 − *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)].

Example 2

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

Number of addresses in the block:	N = NOT (mask) + 1 = 0.0.0.31 + 1 = 32 addresses
First address:	First = (address) AND (mask) = 167.199.170.82
Last address:	Last = (address) OR (NOT mask) = 167.199.170.255

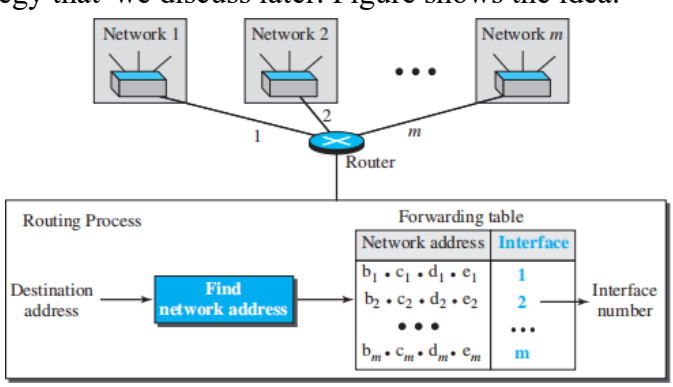
Example 3

In classless addressing, an address cannot per se define the block the address belongs to. For example, the address 230.8.24.56 can belong to many blocks. Some of them are shown below with the value of the prefix associated with that block.

Prefix length: 16	→	Block:	230.8.0.0	to	230.8.255.255
Prefix length: 20	→	Block:	230.8.16.0	to	230.8.31.255
Prefix length: 26	→	Block:	230.8.24.0	to	230.8.24.63
Prefix length: 27	→	Block:	230.8.24.32	to	230.8.24.63
Prefix length: 29	→	Block:	230.8.24.56	to	230.8.24.63
Prefix length: 31	→	Block:	230.8.24.56	to	230.8.24.57

Network Address

The above examples show that, given any address, we can find all information about the block. The first address**, the network address**, is particularly important because it is used in routing a packet to its destination network. For the moment, let us assume that an internet is made of *m* networks and a router with *m* interfaces. When a packet arrives at the router from any source host, the router needs to know to which network the packet should be sent: from which interface the packet should be sent out. When the packet arrives at the network, it reaches its destination host using another strategy that we discuss later. Figure shows the idea.



After the network address has been found, the router consults its forwarding table to find the corresponding interface from which the packet should be sent out. The network address is actually the identifier of the network; each network is identified by its network address.

Block Allocation

Block allocation is managed by the Internet Corporation for Assigned Names and Numbers (ICANN), which assigns large blocks of addresses to Internet Service Providers (ISPs) rather than individual users. For CIDR (Classless Inter-Domain Routing) to function correctly, two key restrictions are applied to the allocated block:

1. The number of requested addresses, N, needs to be a power of 2. The reason is that N = 232 − n or n = 32 − log2N. If N is not a power of 2, we cannot have an integer value for n.
2. The requested block must be allocated where enough contiguous addresses are available. The first address in the block must be divisible by the total number of addresses. This ensures the first address is the prefix followed by (32 − n) zeros, making it divisible by the block size.

$$\textbf{first address} = \textbf{(prefix in decimal)} \times 232 - n = \textbf{(prefix in decimal)} \times N.$$

Example

An ISP has requested a block of 1000 addresses. Since 1000 is not a power of 2, 1024 addresses are granted. The prefix length is calculated as $n = 32 - \log_2 1024 = 22$. An available block, 18.14.12.0/**22**, is granted to the ISP. It can be seen that the first address in decimal is 302,910,464, which is divisible by 1024.

Subnetting

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

Designing Subnets

The subnetworks in a network should be carefully designed to enable the routing of pack ets. We assume the total number of addresses granted to the organization is N, the prefix length is *n*, the assigned number of addresses to each subnetwork is Nsub, and the prefix length for each subnetwork is *n*sub. Then the following steps need to be carefully followed to guarantee the proper operation of the subnetworks.

1. The number of addresses in each subnetwork should be a power of 2.
2. The prefix length for each subnetwork should be found using the following formula: first address = (prefix in decimal) × 232 − *n* = (prefix in decimal) × *N*.

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

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

Finding Information about Each Subnetwork

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

Example 18.5

An organization is granted a block of addresses with the beginning address 14.24.74.0/24. The organization needs to have 3 subblocks of addresses to use in its three subnets: one subblock of 10 addresses, one subblock of 60 addresses, and one subblock of 120 addresses. Design the subblocks.

Solution

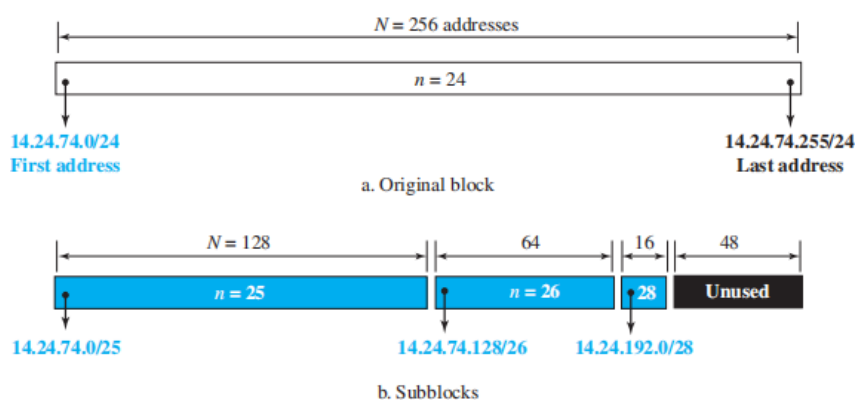
There are $2^{32-24} = 256$ addresses in this block. The first address is 14.24.74.0/24; the last address is 14.24.74.255/24. To satisfy the third requirement, we assign addresses to subblocks, starting with the largest and ending with the smallest one.

a. The number of addresses in the largest subblock, which requires 120 addresses, is not a power of 2. We allocate 128 addresses. The subnet mask for this subnet can be found as $n_1 = 32 - \log_2 128 = 25$. The first address in this block is 14.24.74.0/25; the last address is 14.24.74.127/25.

b. The number of addresses in the second largest subblock, which requires 60 addresses, is not a power of 2 either. We allocate 64 addresses. The subnet mask for this subnet can be found as $n_2 = 32 - \log_2 64 = 26$. The first address in this block is 14.24.74.128/26; the last address is 14.24.74.191/26.

c. The number of addresses in the smallest subblock, which requires 10 addresses, is not a power of 2 either. We allocate 16 addresses. The subnet mask for this subnet can be found as $n_3 = 32 - \log_2 16 = 28$. The first address in this block is 14.24.74.192/28; the last address is 14.24.74.207/28.

If we add all addresses in the previous subblocks, the result is 208 addresses, which means 48 addresses are left in reserve. The first address in this range is 14.24.74.208. The last address is 14.24.74.255. We don't know about the prefix length yet. Figure 18.23 shows the configuration of blocks. We have shown the first address in each block.

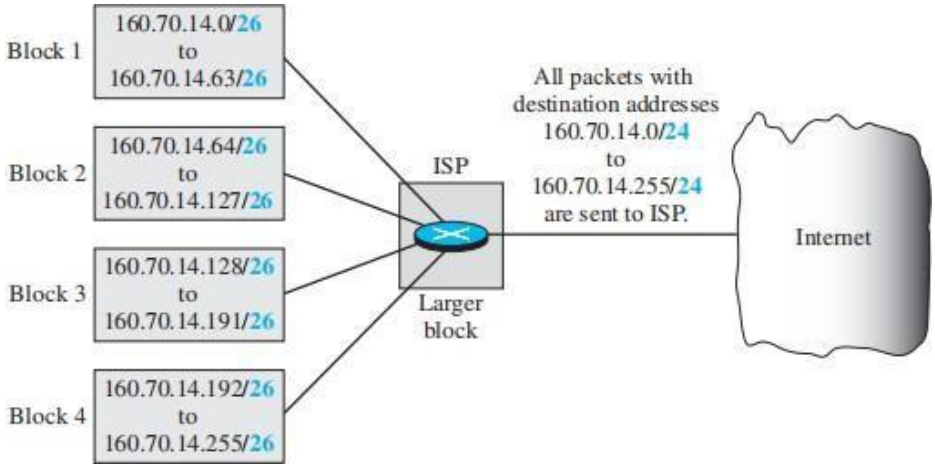


Address Aggregation

One of the advantages of the CIDR strategy is **address aggregation** (sometimes called *address summarization* or *route summarization*). When blocks of addresses are combined to create a larger block, routing can be done based on the prefix of the larger block. ICANN assigns a large block of addresses to an ISP. Each ISP in turn divides its assigned block into smaller subblocks and grants the subblocks to its customers.

Example

Figure 18.24 shows how four small blocks of addresses are assigned to four organizations by an ISP. The ISP combines these four blocks into one single block and advertises the larger block to the rest of the world. Any packet destined for this larger block should be sent to this ISP. It is the responsibility of the ISP to forward the packet to the appropriate organization. This is similar to routing we can find in a postal network. All packages coming from outside a country are sent first to the capital and then distributed to the corresponding destination.



Special Addresses

1. **This-host Address (0.0.0.0/32):** Used when a host doesn't know its own address.
2. **Limited-broadcast Address (255.255.255.255/32):** Used to send a datagram to all devices within a network. Routers block this outside the network.
3. **Loopback Address (127.0.0.0/8):** Used for testing software on the same host; packets never leave the host.
4. **Private Addresses**: Reserved for internal use (e.g., 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, 169.254.0.0/16); used with NAT.
5. **Multicast Addresses (224.0.0.0/4):** Reserved for multicast communication.

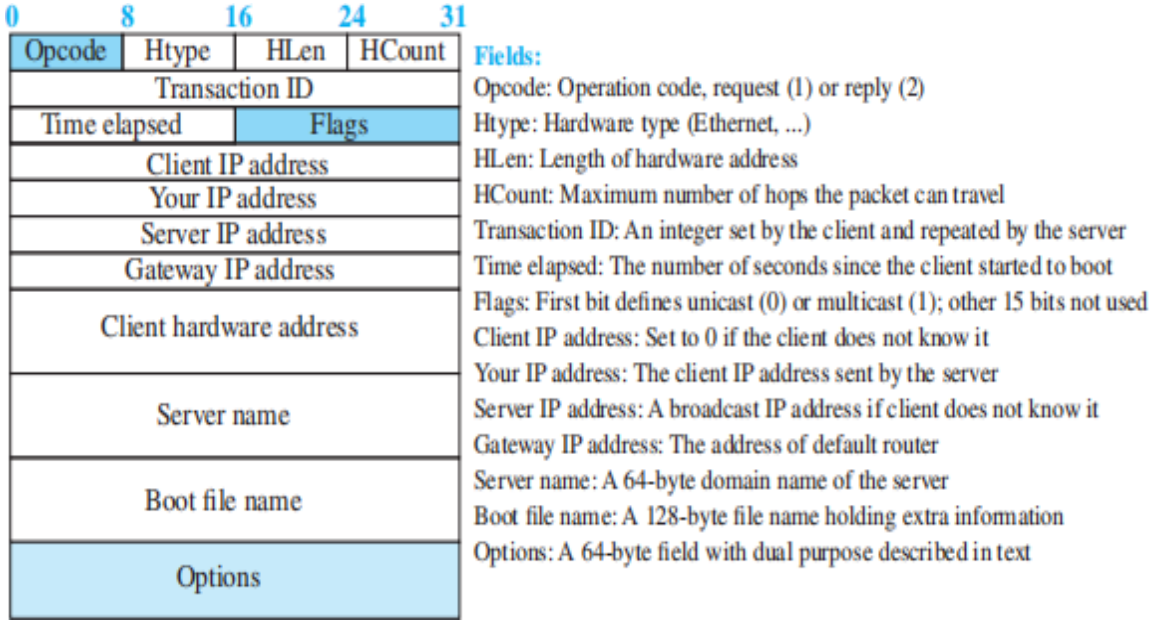
Dynamic Host Configuration Protocol (DHCP)

DHCP is an application-layer protocol that automates IP address assignment, using a client-server model. Widely used on the Internet, it is often referred to as a "plug-and-play" protocol.

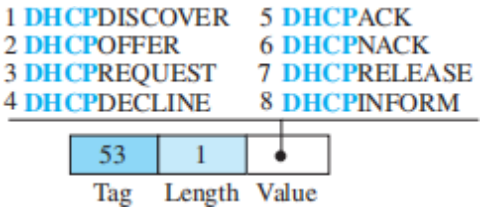
- **Address Assignment:** DHCP can assign permanent or temporary IP addresses. For example, ISPs can use DHCP to provide temporary addresses to users, allowing limited IP resources to serve more customers.

- **Additional Information:** DHCP can also provide essential details like the network prefix, default router address, and name server address.

DHCP Message Format

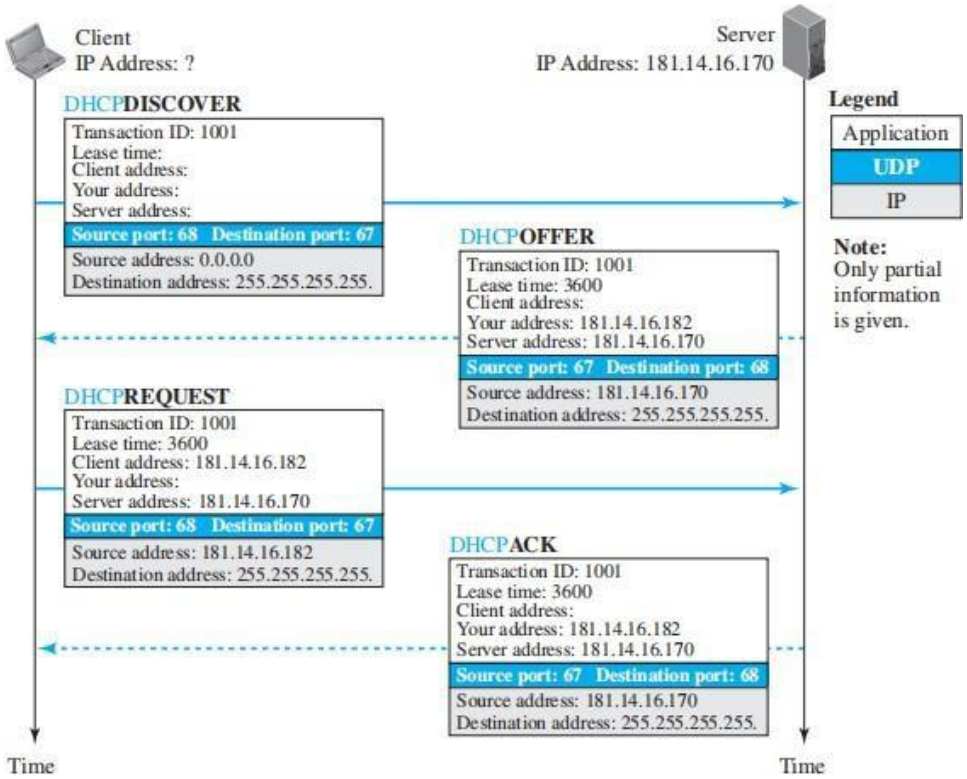


The 64-byte option field in DHCP serves two purposes: carrying additional or vendor-specific information. A special value, called a "magic cookie" (99.130.83.99), helps the client recognize options in the message. The next 60 bytes contain options, structured in three fields: a 1-byte tag, 1-byte length, and variable-length value. The tag field (e.g., 53) can indicate one of the 8 DHCP message types used by the protocol.



DHCP Operation

Figure shows a simple scenario.



1. The host creates a **DHCPDISCOVER** message with only a random transaction-ID, as it doesn't know its own IP address or the server's. The message is sent using UDP (source port 68, destination port 67) and broadcasted (source IP: 0.0.0.0, destination IP: 255.255.255.255).
2. The DHCP server responds with a **DHCPOFFER** message, containing the offered IP address, server address, and lease time. This message is sent with the same port numbers but reversed, using a broadcast address so other servers can also offer better options.
3. The host selects the best offer and sends a **DHCPREQUEST** to the server. The message includes the chosen IP address and is sent as a broadcast (source: new client IP, destination: 255.255.255.255) to notify other servers that their offers were declined.
4. The selected server responds with a **DHCPACK** if the IP address is valid, completing the process. If the IP address is unavailable, a **DHCPNACK** is sent, and the host must restart the process.

Two Well-Known Ports

DHCP uses well-known ports (68 for the client, 67 for the server) to avoid conflicts with other services using ephemeral ports. Since DHCP responses are broadcast, using a well-

known port (68) ensures the DHCP response is only delivered to the correct client, preventing confusion with other clients (e.g., DAYTIME) that might be using the same temporary port. If two DHCP clients are running simultaneously (e.g., after a power failure), they are distinguished by their unique transaction IDs.

Using FTP

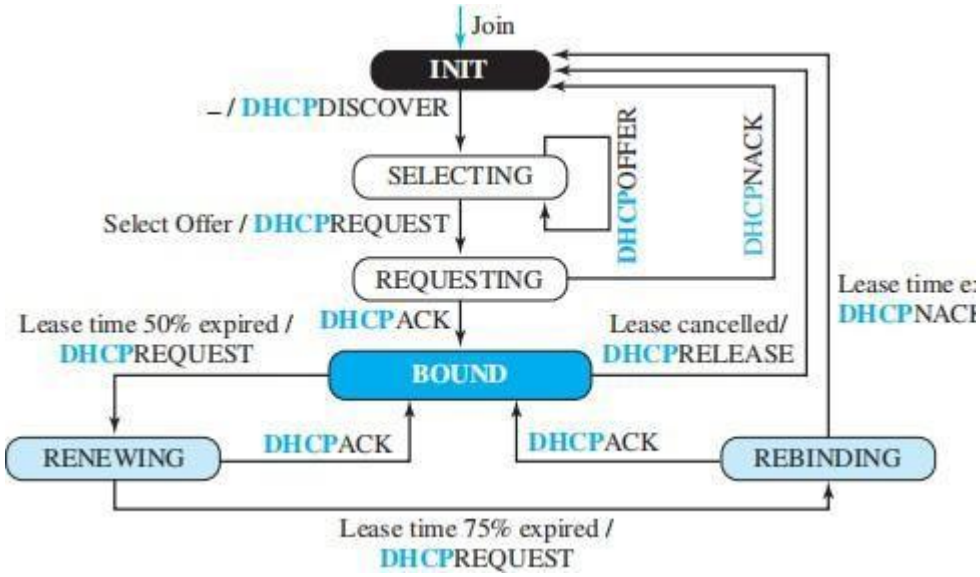
The DHCPACK message includes a pathname to a file with additional information (e.g., DNS server address). The client uses FTP to retrieve this information.

Error Control

Since DHCP relies on unreliable UDP, it ensures error control by requiring UDP checksums and using timers with a retransmission policy. To avoid traffic congestion (e.g., after a power failure), clients use random timers for retransmission.

Transition States

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



1. **INIT state**: The client starts here and sends a *Discover* message to find a DHCP server.
2. **SELECTING state**: After receiving one or more *Offer* messages, the client selects one offer.
3. **REQUESTING state**: The client sends a *Request* message to the selected server and waits.

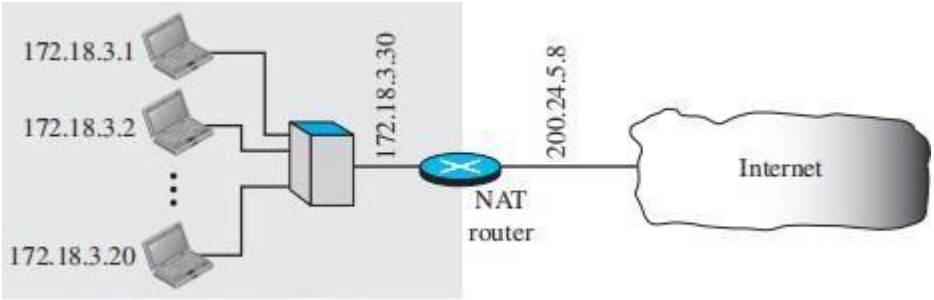
4. **BOUND state**: If the server responds with an *ACK* message, the client uses the assigned IP address.
5. **RENEWING state**: When 50% of the lease time is expired, the client tries to renew the lease by contacting the server. If successful, it stays in the BOUND state.
6. **REBINDING state**: If the lease is 75% expired and no response is received, the client tries to contact any DHCP server. If the server responds, it stays BOUND; otherwise, it goes back to INIT to request a new IP.

Timers:

- **Renewal timer**: Triggered at 50% lease expiration.
- **Rebinding timer**: Triggered at 75% lease expiration.
- **Expiration timer**: Triggered when the full lease expires.

Network Address Resolution (NAT)

ISPs allocate limited IP address ranges to small businesses or households, but expanding these ranges can be difficult due to neighboring allocations. Often, not all devices in a network need simultaneous Internet access. For example, a small business with 20 computers may only need 4 to access the Internet at once, while the rest handle internal tasks. Using private IP addresses for internal communication and a few global IP addresses from the ISP for external communication solves this. Network Address Translation (NAT) allows a network to use private addresses internally and maps them to global addresses for Internet access via a NAT-enabled router.

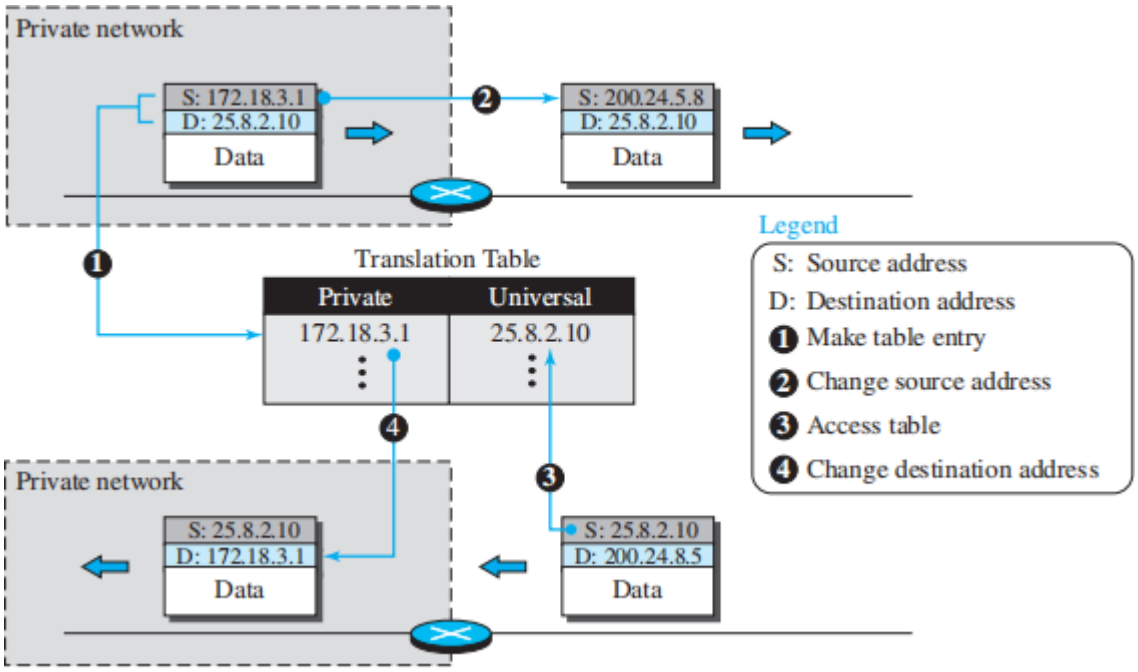


Site using private addresses

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

- ➔ In Network Address Translation (NAT), all outgoing packets pass through a NAT router, which replaces the source address with the global NAT address.
- ➔ Incoming packets also pass through the router, where the destination (global) address is replaced with the corresponding private address. To manage this, the NAT router uses a translation table that links private addresses to external addresses.
- ➔ When an outgoing packet is sent, the router records the destination address.

➔ When a response arrives, the router uses the source (external) address to find the corresponding private address, allowing proper routing within the local network.



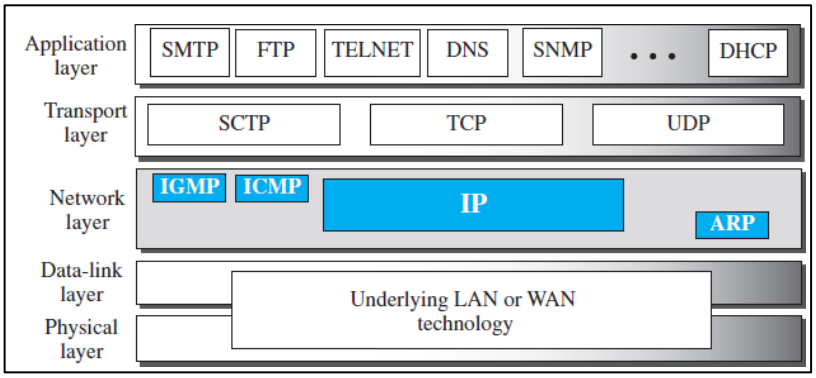
INTERNET PROTOCOL (IP)

➔ 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.

➔ The Internet Control Message Protocol version 4 (ICMPv4) helps IPv4 to handle some errors that may occur in the network-layer delivery. The Internet Group Management Protocol (IGMP) is used to help IPv4 in multicasting.

➔ The Address Resolution Protocol (ARP) is used to glue the network and data-link layers in mapping network-layer addresses to link-layer addresses. Figure shows the positions of these four protocols in the TCP/IP protocol suite.

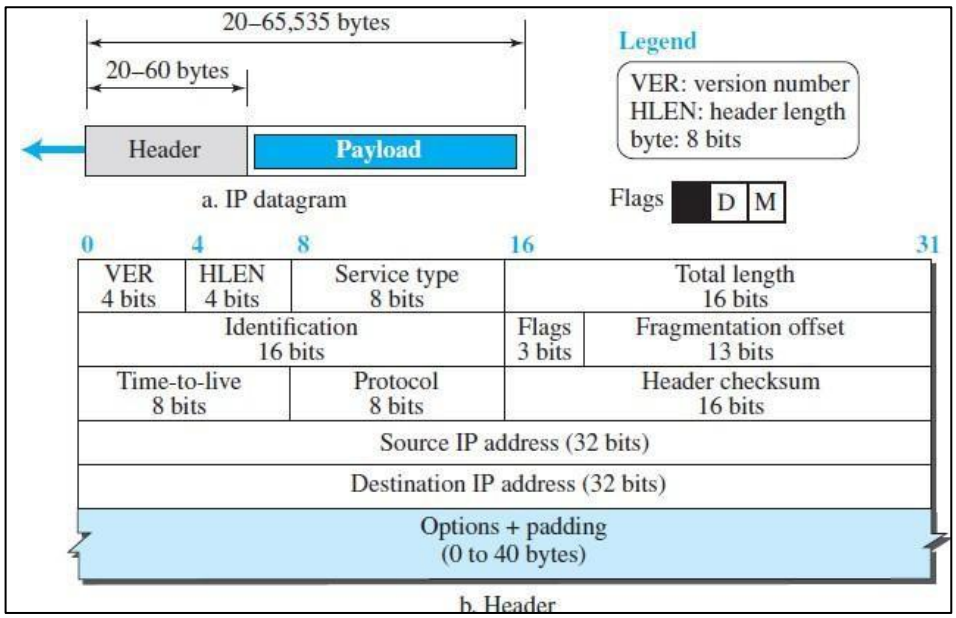


- ➔ IPv4 is an unreliable datagram protocol—a best-effort delivery service. The term *best-effort* means that IPv4 packets can be corrupted, be lost, arrive out of order, or be delayed, and may create congestion for the network.
- ➔ IPv4 is also a connectionless protocol that uses the datagram approach. This means that each datagram is handled independently, and each datagram can follow a different route

.

Datagram Format

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

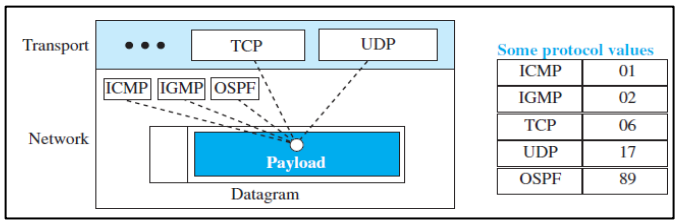


1. ***Version Number.*** The 4-bit version number (VER) field defines the version of the IPv4 protocol, which, obviously, has the value of 4.
2. ***Header Length.*** The 4-bit header length (HLEN) field defines the total length of he datagram header in 4-byte words. The IPv4 datagram has a variable-length header.
3. ***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.

4. ***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).

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

5. ***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.
6. ***Time-to-live.*** The time-to-live (TTL) field is used to control the maximum number of hops (routers) visited by the datagram.
7. ***Protocol.*** In TCP/IP, the data section of a packet, called the *payload,* carries the whole packet from another protocol. This field provides multiplexing at the source and demultiplexing at the destination, as shown in Figure .



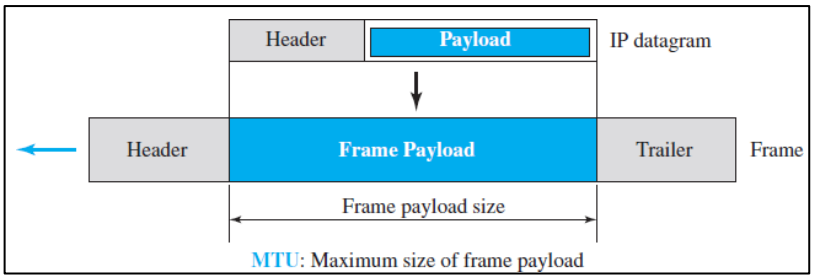
8. ***Header checksum.*** Errors in the IP header can be a disaster. IP adds a header checksum field to check the header, but not the payload.
9. ***Source and Destination Addresses.*** These 32-bit source and destination address fields define the IP address of the source and destination respectively.
10. ***Options.*** A datagram header can have up to 40 bytes of options. Options can be used for network testing and debugging.
11. ***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.

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 sent frame depend on the protocol used by the physical network through which the frame is going to travel.
- ➔ For example, if a router connects a LAN to a WAN, it receives a frame in the LAN format and sends a frame in the WAN format.

Maximum Transfer Unit (MTU)

- ➔ Each link-layer protocol has its own frame format.
- ➔ One of the features of each format is the maximum size of the payload that can be encapsulated.



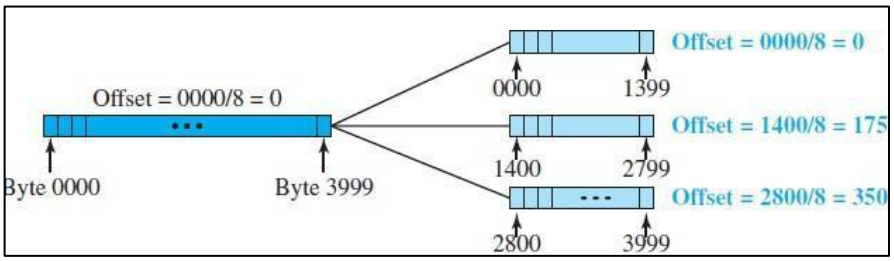
- ➔ The value of the MTU differs from one physical network protocol to another.
- ➔ In order to make the IP protocol independent of the physical network, the designers decided to make the maximum length of the IP datagram equal to 65,535 bytes..
- ➔ For other physical networks, we must divide the datagram 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.
- ➔ The *reassembly* of the datagram, is done only by the destination host, because each fragment becomes an independent datagram.
- ➔ In fragmentation, the payload of the IP datagram is fragmented. most parts of the header, with the exception of some options, must be copied by all fragments.
- ➔ The host or router that fragments a datagram must change the values of three fields: flags, fragmentation offset, and total length.
- ➔ The value of the checksum must be recalculated regardless of fragmentation.

Fields Related to Fragmentation

Three fields in an IP datagram are related to fragmentation:

1. ***Identification-***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 identification number helps the destination in reassembling the datagram.
2. ***Flags***- 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 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.
3. ***fragmentation offset*** The 13-bit *fragmentation offset field* shows the relative position of this fragment with respect to the whole datagram. It is the offset of the data in the original datagram measured in units of 8 bytes.

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



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, are not required for a datagram. They can be used for network testing and debugging.
- ➔ Options are divided into two broad categories: single-byte options and multiple-byte options.

1. ***Single-Byte Options*** There are two single-byte options.
   - ***I. No Operation*** A *no-operation option* is a 1-byte option used as a filler between options.
   - ***II. 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.

2. ***Multliple-Byte Options*** There are four multiple-byte options.
   - ***I. Record Route*** A *record route option* is used to record the Internet routers that handle the datagram.
   - ***II. 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.
   - ***III. 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.
   - ***IV. Timestamp*** A *timestamp option* is used to record the time of datagram processing by a router.

Security of IPv4 Datagrams

There are three security issues that are particularly applicable to the IP protocol

1. ***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.

2. ***Packet Modification***

- ➔ 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.

3. ***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. T
- ➔ This type of attack can be prevented using an origin authentication mechanism.

IPSec

The IP packets today can be protected from the previously mentioned attacks using a protocol called IPSec (IP Security). IPSec provides the following four services:

1. ***Defining Algorithms and Keys.*** The two entities that want to create a secure channel between them can agree on some available algorithms and keys to be used for security purposes.
2. ***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.
3. ***Data Integrity.*** **D**ata 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.
4. ***Origin Authentication.*** **I**PSec can authenticate the origin of the packet to be sure that the packet is not created by an imposter.

IPv6 ADDRESSING

The main reason for migration from IPv4 to IPv6 is the small size of the address space in IPv4. 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. There are 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.

Abbreviation

Although an IPv6 address, even in hexadecimal format, is very long, many of the digits are zeros. In this case, we can abbreviate the address. The leading zeros of a section can be omitted. Further abbreviation, often called ***zero compression,*** can be applied to colon hex notation if there are consecutive sections consisting of zeros only. We can remove all the zeros and replace them with a double semicolon.

FDEC:0:0:0:0:BBFF:0:FFFF ⟶ FDEC::BBFF:0:FFFF

Note that this type of abbreviation is allowed only once per address. If there is more than one run of zero sections, only one of them can be compressed.

Mixed Notation

Sometimes we see a mixed representation of an IPv6 address: colon hex and dotted decimal notation. This is appropriate during the transition period in which an IPv4 address is embedded in an IPv6 address (as the rightmost 32 bits).

CIDR Notation

IPv6 uses hierarchical addressing. For this reason, IPv6 allows slash or CIDR notation. For example, the following shows how we can define a prefix of 60 bits using CIDR. We will later show how an IPv6 address is divided into a prefix and a suffix.

FDEC::BBFF:0:FFFF/60

Address Space

The address space of IPv6 contains 2128 addresses. This address space is 296 times the IPv4 address—definitely no address depletion—as shown, the size of the space is

340, 282, 366, 920, 938, 463, 374, 607, 431, 768, 211, 456.

.

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.

Multicast Address

A multicast address also defines a group of computers. Each member of the group receives a copy. 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.

THE IPv6 PROTOCOL

The change of the IPv6 address size requires the change in the IPv4 packet format. 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.

New options. IPv6 has new options to allow for additional functionalities.

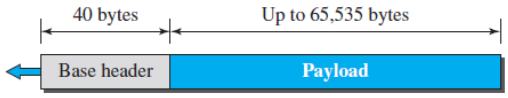
Allowance for extension. IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.

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.

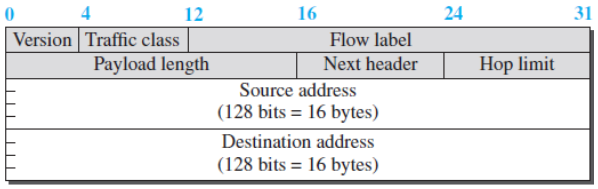
Support for more security. The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format

The IPv6 packet is shown in Figure. 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.



a. IPv6 packet



b. Base header

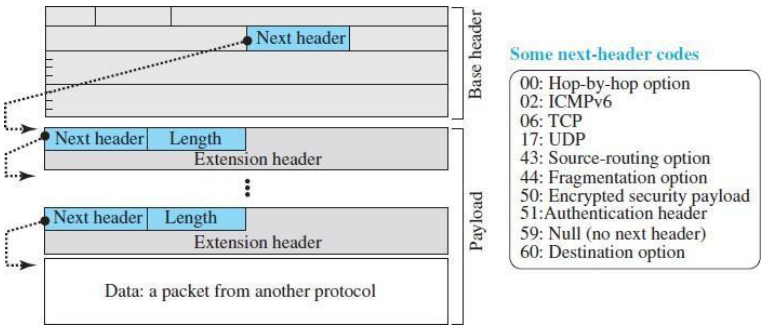
Version. The 4-bit version field defines the version number of the IP. For IPv6, the value is 6.

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.

Payload length. The 2-byte payload length field defines the length of the IP datagram excluding the header.

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..
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.



- ➔ 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.

Concept of Flow and Priority in IPv6

- ➔ In version 6, the flow label has been added to the format of the IPv6 datagram to allow us to use IPv6 as a connection-oriented protocol.
- ➔ To a router, a flow is a sequence of packets that share the same characteristics, such as traveling the same path, using the same resources, having the same kind of security, and so on.
- ➔ A router that supports the handling of flow labels has a flow label table. The table has an entry for each active flow label; each entry defines the services required by the corresponding flow label.
- ➔ In its simplest form, a flow label can be used to speed up the processing of a packet by a router.

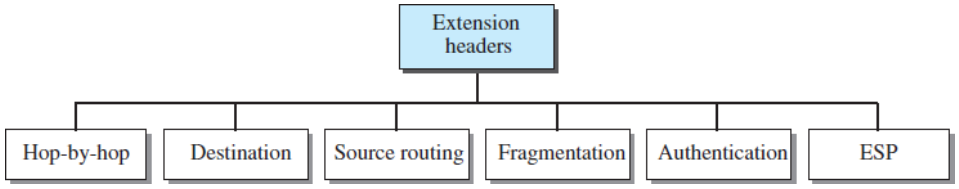
Fragmentation and Reassembly

- ➔ IPv6 datagrams can be fragmented only by the source, not by the routers; the reassembly takes place at the destination.

- ➔ The fragmentation of a packet in a router needs a lot of processing hence processing at router is not allowed.
- ➔ 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 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. The base header can be followed by up to six **extension headers**, to give more functionality**.**



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. So far, only three hop by- 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. If for any reason a longer payload(>65,535 bytes) is required, the jumbo payload option is used 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.

Source Routing

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

Fragmentation

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.

Authentication

The **authentication** extension header has a dual purpose: it validates the message sender and ensures the integrity of data.

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

Introduction to Routing Algorithms

Routing algorithms are essential in computer networks to determine the most efficient path for data packets to travel from a source to a destination. These algorithms help manage network traffic, minimize delays, and ensure optimal usage of network resources.

If a datagram is destined for only one destination (one-to-one delivery), we have *unicast routing*. If the datagram is destined for several destinations (one-to-many delivery), we have *multicast routing*.

Unicast Routing Algorithm

Introduction

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

General Idea

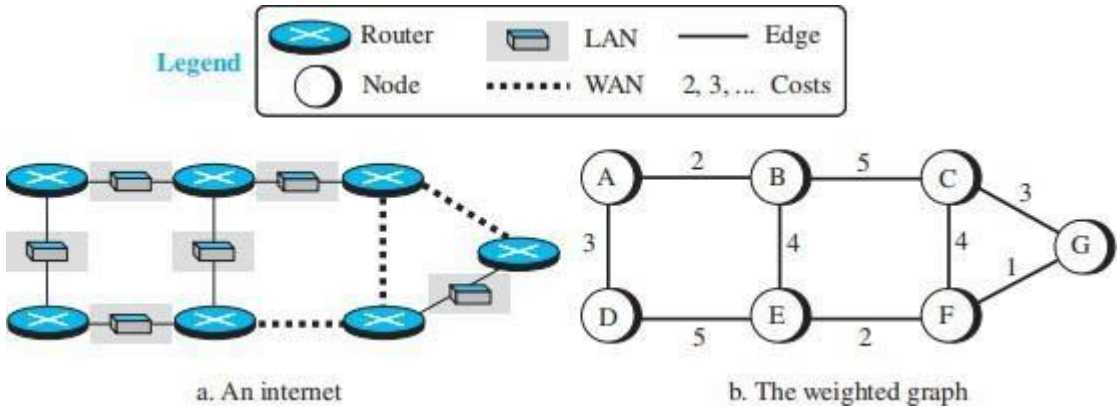
- ➔ In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
- ➔ The source host and the destination host need no forwarding table either because they send and receive the packet from its default router in its local network.
- ➔ The only the routers that glue together the networks in the internet need forwarding tables.
- ➔ Routing a packet from its source to its destination means routing the packet from a *source router* (the default router of the source host) to a *destination router* (the router connected to the destination network).
- ➔ Although a packet needs to visit the source and the destination routers, the packet should visit intermediate routers. There are several routes that a packet can travel from the source to the destination; it must be determined is which route the packet should take.

An Internet as a Graph

- ➔ To find the best route, an internet can be modelled as a graph. A graph in computer science is a set of nodes and edges (lines) that connect the nodes.
- ➔ To model an internet as a graph, we can think of each router as a node and each network between a pair of routers as an edge.
- ➔ An internet is, in fact, modelled as a weighted graph, in which each edge is associated with a cost.
- ➔ If a weighted graph is used to represent a geographical area, the nodes can be cities and the edges can be roads connecting the cities; the weights, in this case, are distances between cities.
- ➔ In routing, however, the cost of an edge has a different interpretation in different routing protocols.

Least-Cost Routing

- ➔ When an internet is modelled as a weighted graph, one of the ways to interpret the ***best route*** from the source router to the destination router is to find the *least cost* between the two.
- ➔ In Figure below, the best route between A and E is A-B-E, with the cost of 6. This means that each router needs to find the least-cost route between itself and all the other routers to be able to route a packet using this criterion.

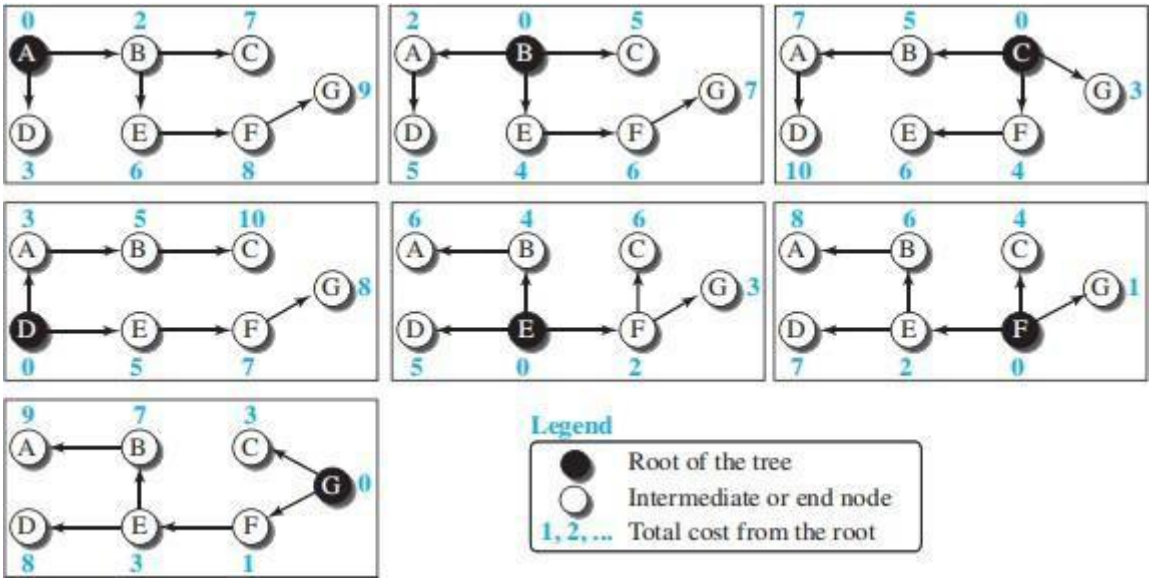


a. An internet b. The weighted graph

Least-Cost Trees

- ➔ If there are N routers in a network, each router needs to find the cheapest (least-cost) path to every other router. For example, with 10 routers, you would need to calculate 90 different paths to connect all routers.
- ➔ Instead of calculating all the paths separately, we can group them into something called a least-cost tree (also known as a shortest-path tree).
- ➔ A least-cost tree starts from one router (the source) and connects to all the other routers using the cheapest path for each connection. This way, you can use just one tree to find the best routes from that source router to every other router.

- ➔ Instead of calculating N × (N - 1) paths (which is 90 paths for 10 routers), you only need to calculate N trees (one for each router). So, with 10 routers, you only need 10 trees instead of 90 paths.



ROUTING ALGORITHMS

Several routing algorithms have been designed in the past. The differences between these methods are in the way they interpret the least cost and the way they create the least-cost tree for each node.

The Distance-Vector (DV) Routing Algorithm

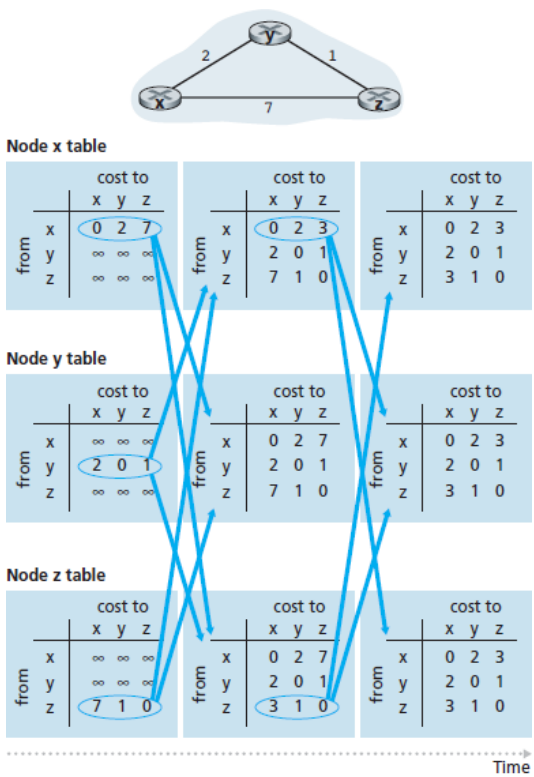
- ➔ The distance vector (DV) algorithm is iterative, asynchronous, and distributed.
- ➔ It is iterative that the process continues on until no more information is exchanged between neighbours.
- ➔ The algorithm is asynchronous in that it does not require all of the nodes to operate in lockstep with each other.
- ➔ Let dx(y) be the cost of the least-cost path from node x to node y. Then the least costs are related by the celebrated Bellman-Ford equation, namely,

$$\mathbf{dx(y) = minv\{c(x,v) + dv(y)\}}$$

- ➔ With the DV algorithm, each node x maintains the following routing information:
  - For each neighbour v, the cost c(x,v) from x to directly attached neighbour, v
  - Node x's distance vector, that is, Dx = [Dx(y): y in N], containing x's estimate of its cost to all destinations, y, in N
  - The distance vectors of each of its neighbours, that is, Dv = [Dv(y): y in N] for each neighbour v of x

```
Initialization:
   for all destinations y in N:
      D_x(y) = c(x,y)    /* if y is not a neighbor then c(x,y) = ∞ */
   for each neighbor w
      D_w(y) = ? for all destinations y in N
   for each neighbor w
      send distance vector D_x = [D_x(y): y in N] to w

loop
   wait (until I see a link cost change to some neighbor w or
         until I receive a distance vector from some neighbor w)

   for each y in N:
      D_x(y) = min_v{c(x,v) + D_v(y)}

   if D_x(y) changed for any destination y
      send distance vector D_x = [D_x(y): y in N] to all neighbors

forever
```

Figure illustrates the operation of the DV algorithm for the simple three node network shown at the top of the figure.



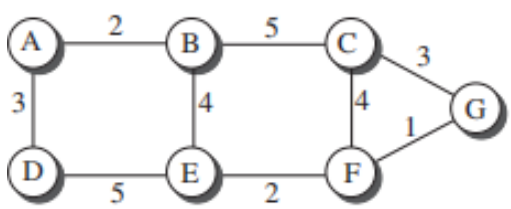
Distance-vector (DV) algorithm

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.

Link-State Routing

- ➔ Link-State (LS) routing is a method used by routers to determine the best, or least-cost, path through a network.
- ➔ In this method, each connection between routers is referred to as a **link**, and the **link-state** describes the cost of using that connection. Lower costs are preferred, while a link with an infinite cost means it is unavailable or broken.
- ➔ To create efficient routing paths, each router must have a complete map of the network, which includes the state (cost) of every link. This information is stored in the **Link-State Database (LSDB)**, which is shared across all routers.
- ➔ The LSDB is often represented as a matrix, where each value shows the cost of a link between two routers.
- ➔ With this database, each router can build the least-cost tree, ensuring it knows the shortest routes to every other router.



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

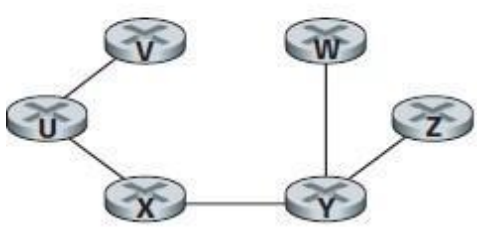
b. Link state database

- ➔ To create the **Link-State Database (LSDB)**, routers use a process called **flooding**. Each router sends a greeting to its direct neighbors to gather two pieces of information: the neighbor's identity and the cost of the link.
- ➔ This data is packaged into a **Link-State Packet (LSP)** and sent to all neighboring routers. When a router receives an LSP, it checks if the information is new by comparing the sequence number.
- ➔ If the LSP is new, the router keeps it and sends copies to other neighbors, except the one from which it was received.
- ➔ This process repeats until all routers have the latest LSPs, allowing each router to create an identical **LSDB**, which provides a complete map of the network. This LSDB helps routers calculate the least-cost paths to all other nodes.

- The link-state routing algorithm is known as **Dijkstra's algorithm**. Dijkstra's algorithm computes the least-cost path from one node to all other nodes in the network.
- The following notations are used:
  - D(v): cost of the least-cost path from the source node to destination v as of this iteration of the algorithm.
  - p(v): previous node (neighbor of v) along the current least-cost path from the source to v.
  - N' : subset of nodes; v is in N ' if the least-cost path from the source to v is definitively known.

```
Initialization:
  N' = {u}
  for all nodes v
    if v is a neighbor of u
      then D(v) = c(u,v)
    else D(v) = ∞

Loop
  find w not in N' such that D(w) is a minimum
  add w to N'
  update D(v) for each neighbor v of w and not in N':
        D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   least path cost to w plus cost from w to v */
until N'= N
```

step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



Destination	Link
v	(u, v)
w	(u, x)
x	(u, x)
y	(u, x)
z	(u, x)

Least cost path and forwarding table for nodule u

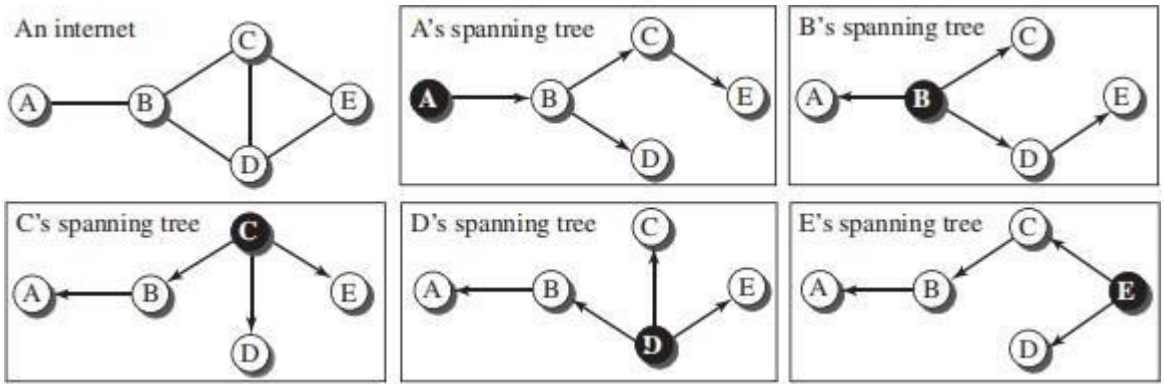
Path-Vector Routing

The **link-state (LS)** and **distance-vector (DV)** routing focus on finding the least-cost path, but there are situations where this goal isn't the priority. For example, a sender may want to avoid certain routers due to security concerns or competition, but least-cost routing doesn't allow for such policies. These methods only focus on minimizing cost, meaning packets could still pass through undesired routers if they are part of the least-cost path. In some cases, such as for security or policy reasons, the sender may prioritize **reachability** over cost, ensuring that packets reach their destination efficiently without considering the route's cost.

Path-vector (PV) routing is a routing algorithm designed to address the limitations of **link-state (LS)** and **distance-vector (DV)** routing, which focus on least-cost paths. Unlike those methods, PV routing allows the source to control the route based on policies, such as avoiding certain routers for security or business reasons. While primarily used for routing between ISPs, PV routing offers flexibility by letting the source determine the best route according to its own policies rather than cost.

Spanning Trees

In **path-vector routing**, the route from a source to all destinations is determined by the **best spanning tree**, but unlike the least-cost tree, the best spanning tree is chosen based on the source's policies. If multiple routes exist, the source can pick the route that aligns with its preferences, which may include avoiding certain nodes or minimizing the number of nodes visited. For example, in a network of five nodes, sources like **A** and **E** may create spanning trees that avoid node **D** as an intermediary, while **B** may avoid node **C**. These policies allow the source to shape the path according to its specific needs.



Creation of Spanning Trees

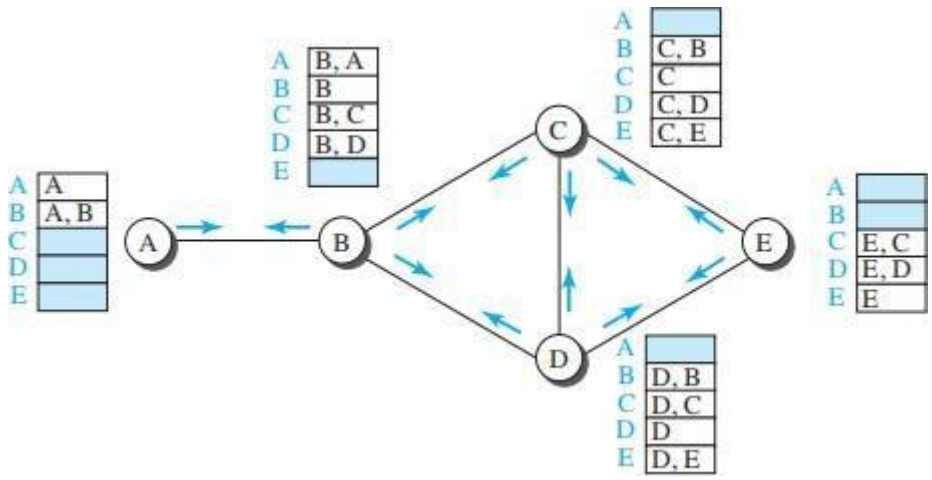
Path-vector routing, like distance-vector routing, is an asynchronous and distributed routing algorithm. The spanning trees are made, gradually and asynchronously, by each node. When a node is booted, it creates a path vector based on the information it can obtain about its immediate

neighbour. A node sends greeting messages to its immediate neighbours to collect these pieces of information.

Each node, after the creation of the initial path vector, sends it to all its immediate neighbours. Each node, when it receives a path vector from a neighbour, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost. We can define this equation as

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

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.



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***.

```
Path_Vector_Routing ( )
{
    // Initialization
    for (y = 1 to N)
    {
        if (y is myself)
            Path[y] = myself
        else if (y is a neighbor)
            Path[y] = myself + neighbor node
        else
            Path[y] = empty
    }
    Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    // Update
    repeat (forever)
    {
        wait (for a vector Path_w from a neighbor w)
        for (y = 1 to N)
        {
            if (Path_w includes myself)
                discard the path                    // Avoid any loop
            else
                Path[y] = best {Path[y], (myself + Path_w[y])}
        }
        If (there is a change in the vector)
            Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    }
} // End of Path Vector
```

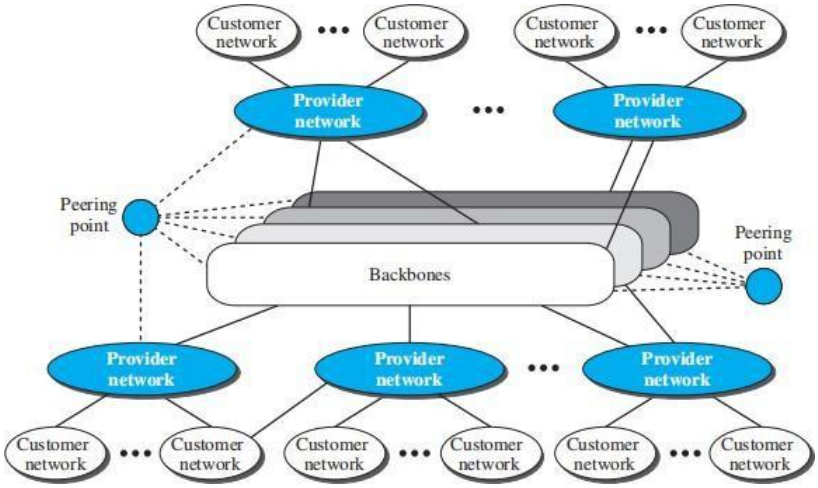
UNICAST ROUTING PROTOCOLS

Routing protocols used in the Internet are based on unicast routing algorithms, but protocols involve more than just algorithms. They also define the domain of operation, the messages exchanged, communication between routers, and interaction with protocols across different domains. The three main Internet routing protocols discussed are 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. These protocols handle different aspects of routing within and between networks.

Internet Structure

The Internet has changed from a tree-like structure, with a single backbone, to a multi-backbone structure run by different private corporations today. Although it is difficult to give a general view of the Internet today, we can say that the Internet has a structure similar to what is shown in Figure.



There are several *backbones* run by private communication companies that provide global connectivity. These backbones are connected by some *peering points* that allow connectivity between backbones. At a lower level, there are some *provider networks* that use the backbones for global connectivity but provide services to Internet customers.

Finally, there are some customer *networks* that use the services provided by the provider networks. Any of these three entities (backbone, provider network, or customer network) can be called an Internet Service Provider or ISP. They provide services, but at different levels.

Hierarchical Routing

- The Internet is vast, and routing cannot rely on a single protocol due to scalability and administrative challenges:
  - **Scalability problem:** Large forwarding tables make searching slow and traffic updates overwhelming.
  - **Administrative issue:** Each ISP is managed independently and may have unique needs like using specific routing algorithms or hardware.
- Hierarchical routing divides the Internet into **Autonomous Systems (AS)**, where each ISP operates as an AS with control over its routing protocol (intra-AS routing).
- **Intra-AS Routing Protocols (Interior Gateway Protocols or IGPs):** Examples include RIP and OSPF, used within an AS.

- **Inter-AS Routing Protocol (Exterior Gateway Protocol or EGP):** BGP is currently the only inter-AS routing protocol, connecting multiple ASs globally.

Types of Autonomous Systems:

1. **Stub AS:**
   - Connected to only one other AS. Traffic can originate or terminate here, but it does not pass through.
   - Example: A customer network.
2. **Multihomed AS:**
   - Connected to more than one AS but doesn't allow traffic to pass through.
   - Example: A customer AS using services from multiple providers but blocking transit traffic.
3. **Transient AS:**
   - Connected to multiple ASs and allows traffic to pass through.
   - Example: Backbone or provider networks.

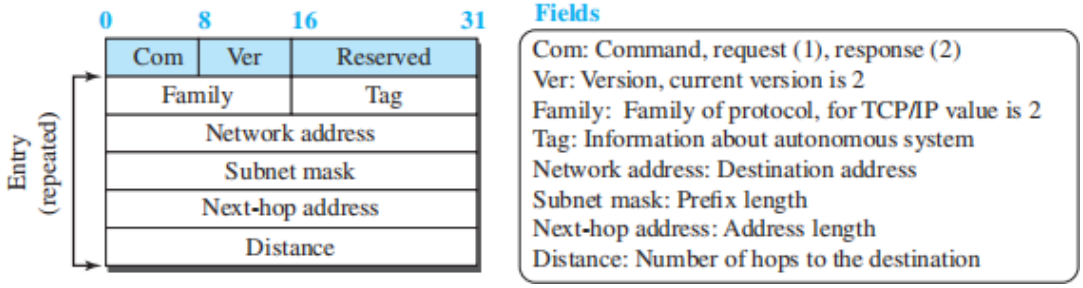
Each AS is assigned a unique Autonomous System Number (ASN) to identify it in the Internet structure.

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 we described earlier. RIP was started as part of the Xerox Network System (XNS), but it was the Berkeley Software Distribution (BSD) version of UNIX that helped make the use of RIP widespread.

RIP Messages

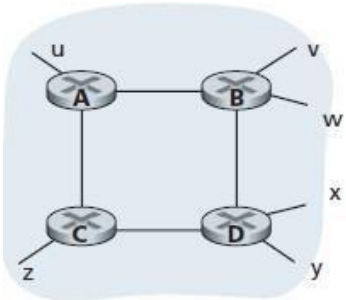
Two RIP processes, a client and a server, like any other processes, need to exchange messages. RIP defines the format of the message. Part of the message, which we call *entry*, can be repeated as needed in a message. Each entry carries the information related to one line in the forwarding table of the router that sends the message.



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

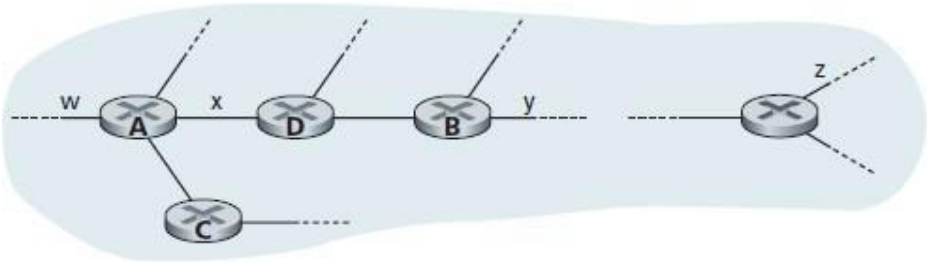
- ➔ RIP is a distance-vector protocol. Figure illustrates an AS with six leaf subnets. The table in the figure indicates the number of hops from the source A to each of the leaf subnets.
- ➔ In RIP, routing updates are exchanged between neighbours approximately every 30 seconds using a RIP response message.



Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2

Number of hops from source router A to various subnets

- ➔ Response messages are also known as RIP advertisements. Consider the portion of an AS shown in Figure.



A portion of an autonomous system

- ➔ Figure shows the routing table for router D. suppose that 30 seconds later, router D receives from router A the advertisement shown in next Figure.

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	B	7
x	–	1
....	....	....

Routing table in router *D* before receiving advertisement from router *A*

Destination Subnet	Next Router	Number of Hops to Destination
z	C	4
w	–	1
x	–	1
....	....	....

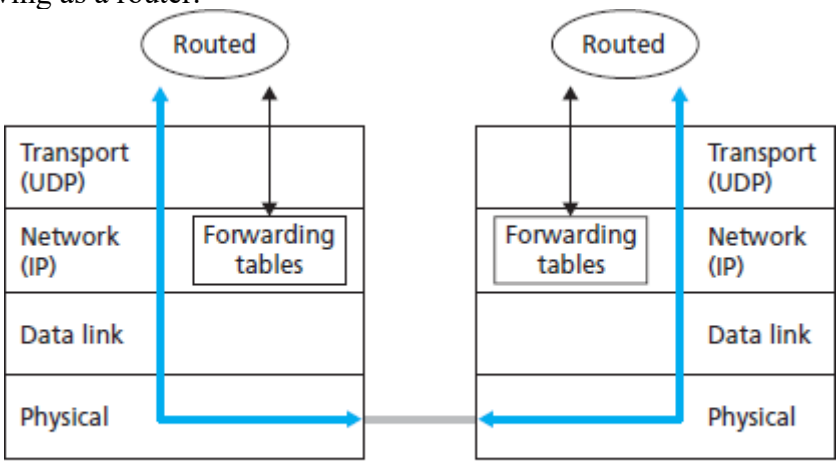
Advertisement from router A

- ➔ Subnet z is only four hops away from router A. Router D, upon receiving the advertisement, merges the advertisement with the old routing table.
- ➔ Router D learns that there is a path through router A to subnet z that is shorter than the path through router B.
- ➔ Thus, router D updates its routing table to account for the shorter shortest path, as shown in Figure.

Destination Subnet	Next Router	Number of Hops to Destination
w	A	2
y	B	2
z	A	5
....	....	....

Routing table in router *D* after receiving advertisement from router *A*

Figure below shows how RIP is implemented in a UNIX system, for example, a UNIX workstation serving as a router.



Implementation of RIP as the *routed* daemon

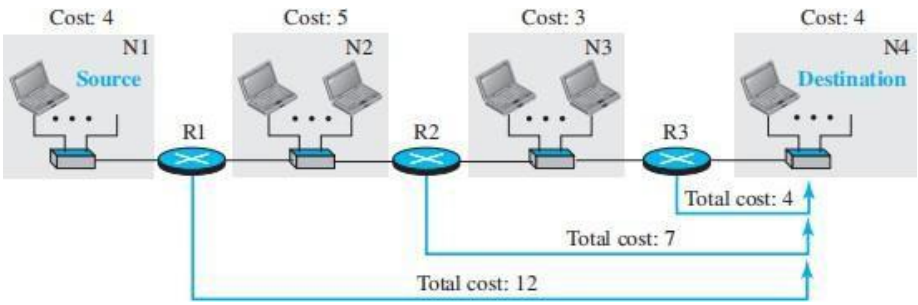
- ➔ A process called **routed** executes RIP, that is, maintains routing information and exchanges messages with routed processes running in neighbouring routers.
- ➔ Because RIP is implemented as an application-layer process, it can send and receive messages over a standard socket and use a standard transport protocol.
- ➔ As shown, RIP is implemented as an application-layer protocol running over UDP.

Open Shortest Path First (OSPF)

Open Shortest Path First (OSPF) is an intradomain routing protocol like RIP but is based on the link-state routing protocol. It is an open protocol, meaning the specification is publicly available. Unlike RIP, OSPF allows each link to be assigned a weight based on factors such as throughput, round-trip time, or reliability, though administrators can use hop count as a cost. Different Types of Service (TOS) can have different weights for cost calculation.

Metric:

OSPF calculates the cost of reaching a destination from a source by considering link weights, which can vary based on the type of service. This is shown in Figure below, where the total cost to the destination is calculated by summing the costs of individual links.



Forwarding Tables:

OSPF routers use Dijkstra's algorithm to create forwarding tables by building the shortest-path tree to destinations. The difference between OSPF and RIP forwarding tables is mainly in the cost values. If OSPF uses hop count as its metric, its forwarding tables would be identical to those of RIP. Both protocols determine the best route using shortest-path trees.

Forwarding table for R1

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

Forwarding table for R2

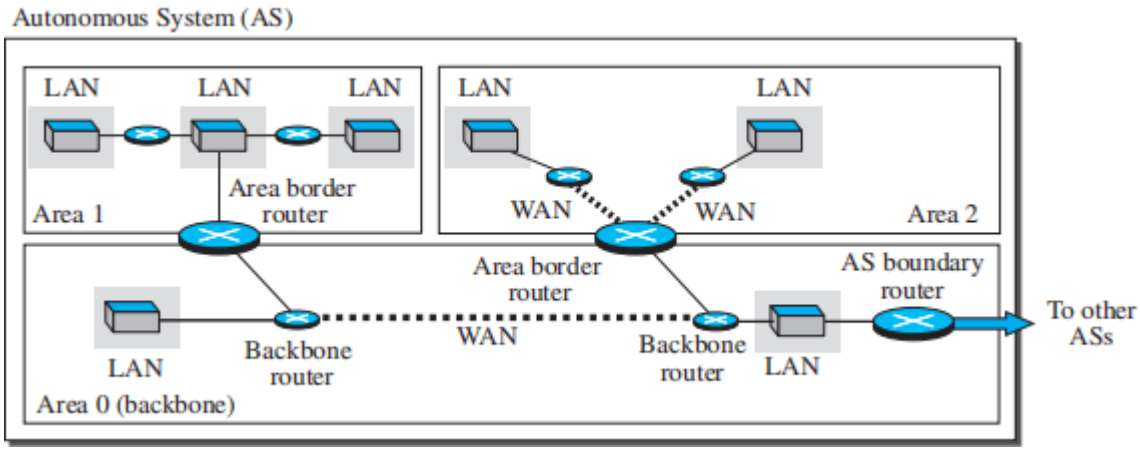
Destination network	Next router	Cost
N1	R1	9
N2	—	5
N3	—	3
N4	R3	7

Forwarding table for R3

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

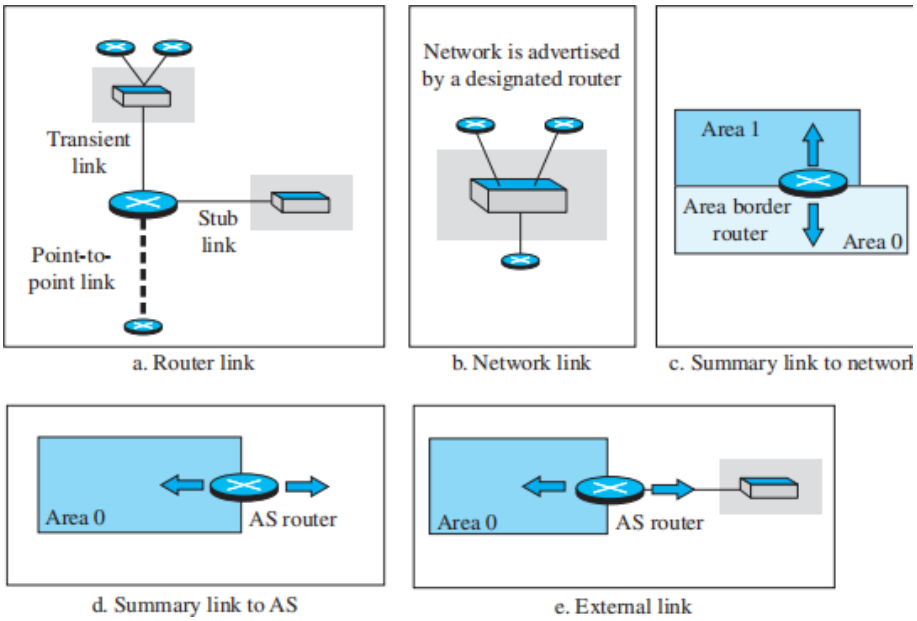
Areas:

OSPF is designed for both small and large autonomous systems (AS). In large ASs, flooding link-state packets (LSPs) across the entire network can cause congestion, so OSPF introduces areas to localize LSP flooding. The AS is divided into smaller sections called areas, with one backbone area (Area 0) responsible for inter-area communication.



Link-State Advertisement (LSA):
OSPF routers advertise their link states to neighbors for forming a global link-state database (LSDB). Unlike the simple graph model, OSPF distinguishes between different types of nodes and links, requiring various types of advertisements:

Router Link: Announces router existence and its connection to other entities.
Network Link: Advertises the existence of a network, but with no associated cost.
Summary Link to Network: Advertised by area border routers to summarize links between areas.
Summary Link to AS: Announced by AS boundary routers to inform other areas of external AS links.
External Link: Advertises external network routes to the AS.

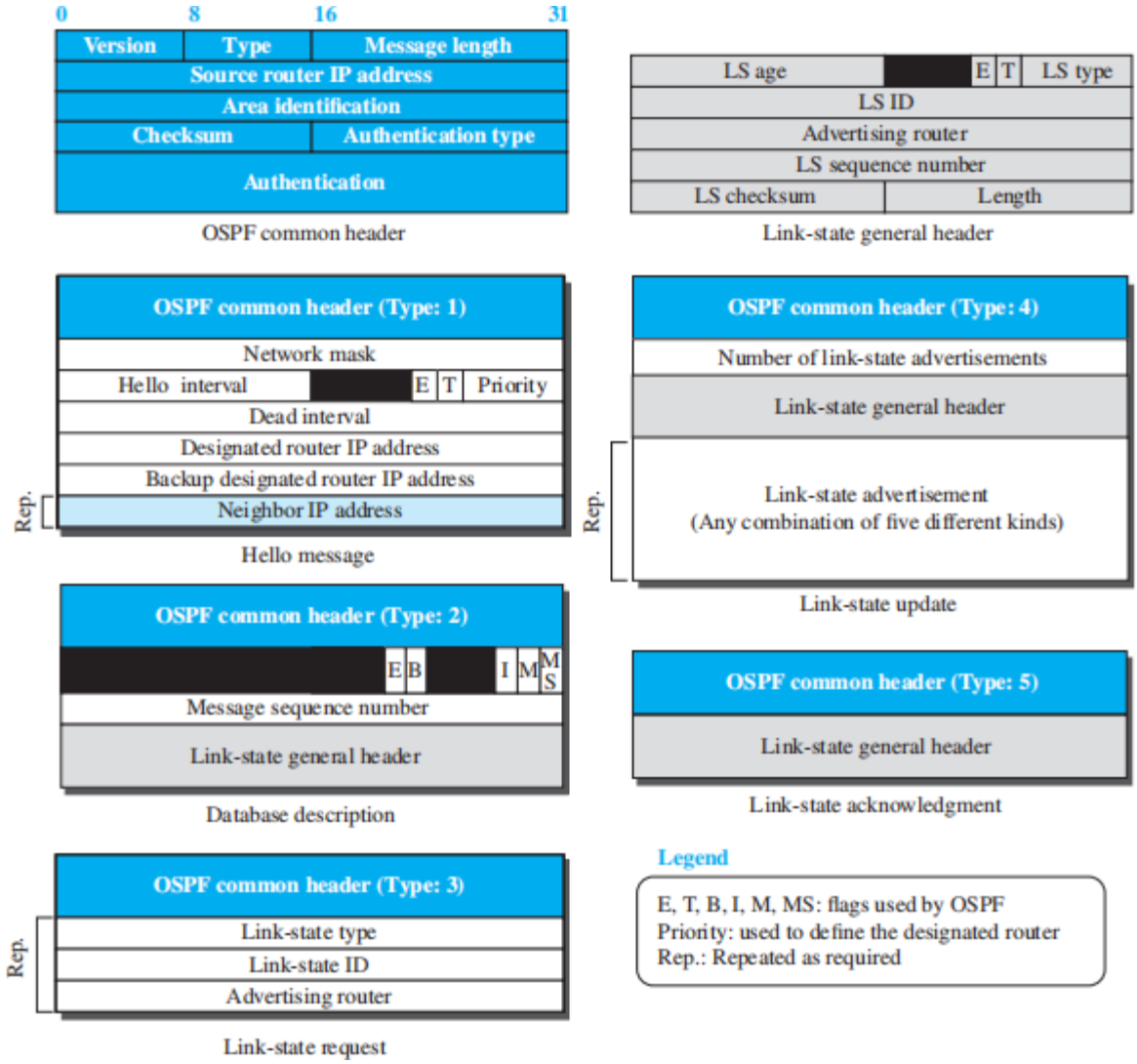


OSPF Implementation:

OSPF operates at the network layer and uses IP for message propagation. OSPF messages are encapsulated in IP datagrams with a protocol field value of 89. OSPF has two versions, with version 2 being the most widely implemented.

OSPF Messages:

OSPF uses five message types, each with a distinct format :



Hello Message (Type 1): Used by routers to introduce themselves and announce their known neighbors.

Database Description Message (Type 2): Sent in response to Hello messages, allowing routers to acquire the full LSDB.

Link-State Request (Type 3): Sent when a router requires specific LS information.

Link-State Update (Type 4): The primary message used to build the LSDB, with versions for each type of link.
Link-State Acknowledgment (Type 5): Provides reliability by confirming receipt of link-state updates.
Authentication: OSPF supports message authentication to prevent unauthorized routers from joining the routing system. This is critical for network security and preventing malicious interference.

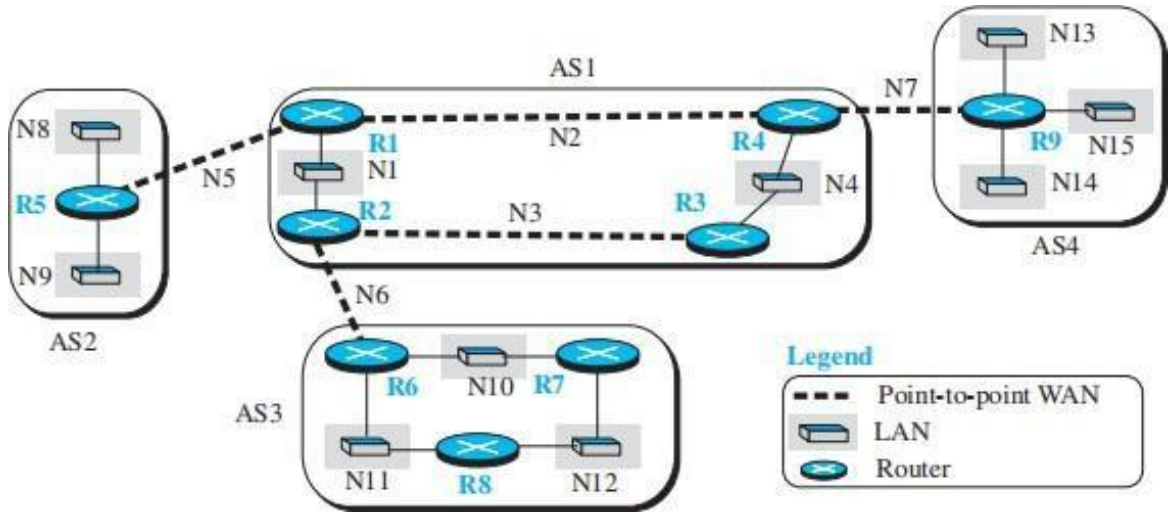
OSPF Algorithm:
OSPF uses a modified link-state routing algorithm. After routers form their shortest-path trees, they create corresponding routing tables. The algorithm also handles OSPF message exchange.

Performance:

Update Messages: OSPF's LSPs are complex and can create heavy traffic in large areas, using considerable bandwidth.
Convergence of Forwarding Tables: OSPF converges relatively quickly once flooding is complete, although Dijkstra's algorithm can take time to run.
Robustness: OSPF is more robust than RIP since routers operate independently after constructing their LSDBs. A failure in one router has less impact on the overall network.

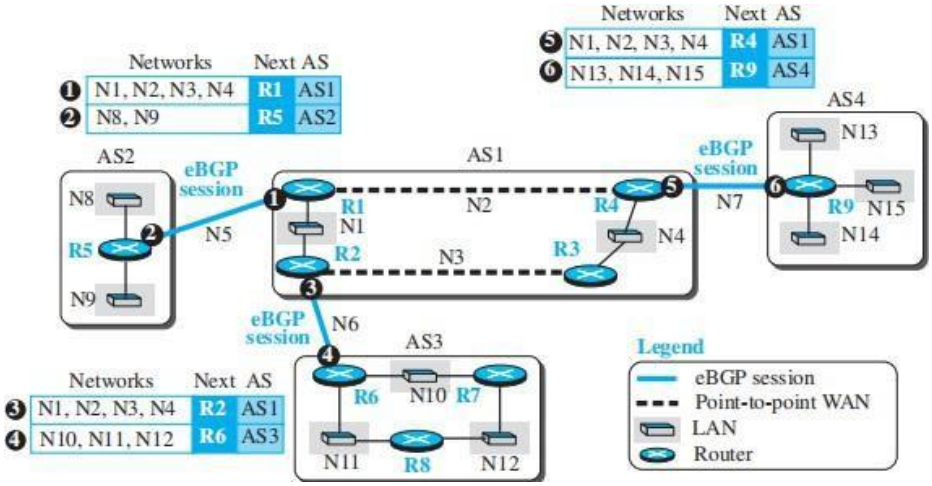
Border Gateway Protocol Version 4 (BGP4)

BGP4 is the predominant interdomain routing protocol used across the internet. It is a path-vector protocol, which means it provides the entire path to the destination instead of just the next hop. This path information is useful for avoiding routing loops and ensuring stable paths between autonomous systems (AS). Figure illustrates a network with four ASs: AS1 is a transient AS, and AS2, AS3, and AS4 are stub ASs. Data exchanges between the stub ASs occur through AS1.



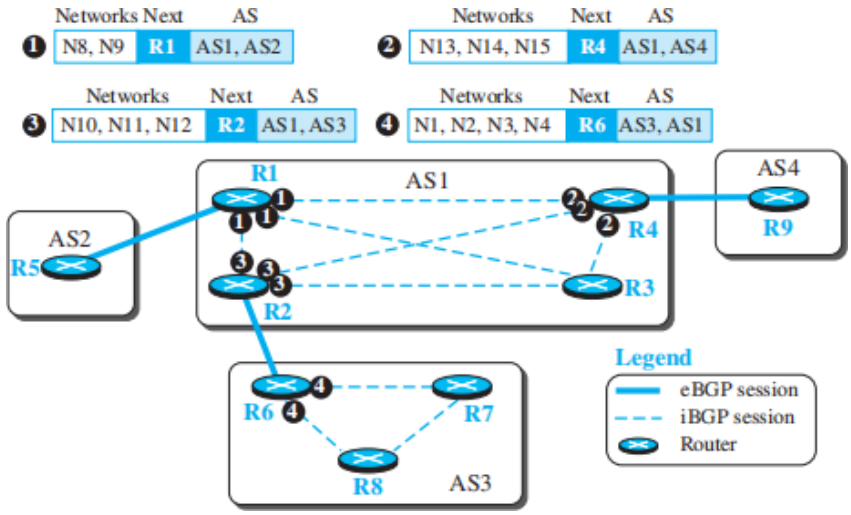
External BGP (eBGP) Operation

BGP operates over Transmission Control Protocol (TCP) connections, ensuring reliable communication between routers. The protocol defines two types of operations: external (eBGP) and internal (iBGP). eBGP sessions are established between routers belonging to different ASs, as seen in Figure. In these sessions, routers share routing information to other ASs by exchanging reachability information about network prefixes. However, this only provides visibility between the border routers, so internal routers need iBGP for full route dissemination.



Internal BGP (iBGP) Operation

Within the same AS, routers establish iBGP sessions to disseminate the routes learned from eBGP sessions. A fully connected mesh of iBGP sessions is required to prevent routing loops. Each iBGP router must have a direct session with every other iBGP router, as depicted in Figure.



This ensures that all routers inside an AS have complete knowledge of external networks. After iBGP sessions converge, all routers will have consistent routing information about the external networks, allowing them to build accurate path tables, as demonstrated in Figure.

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

Path table for R1

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

Path table for R2

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

Path table for R3

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

Path table for R4

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 R5

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 R6

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 R7

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

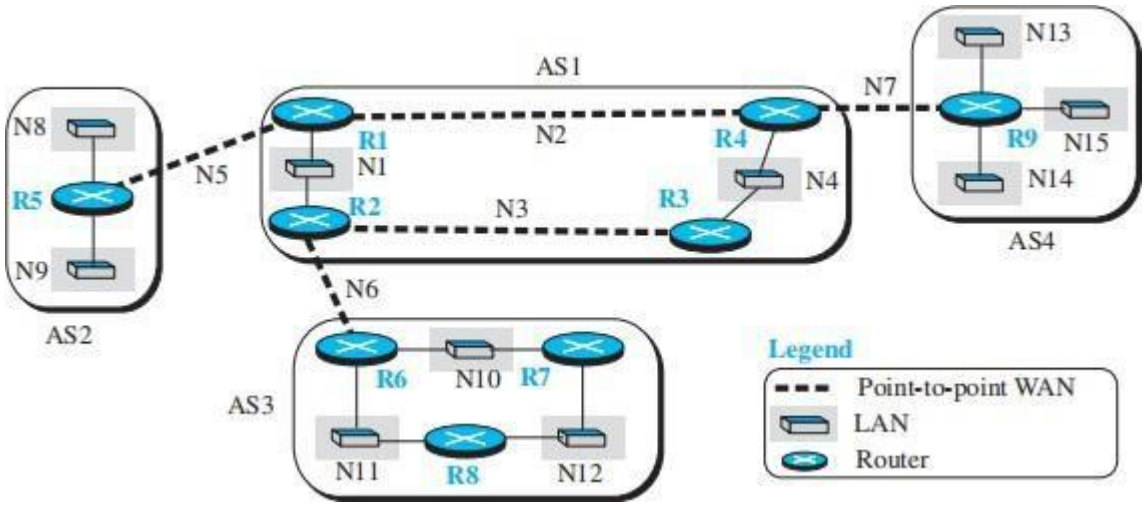
Path table for R8

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

Path table for R9

Injection into Intradomain Routing

BGP injects routes learned from external sources into the intradomain routing protocol (such as RIP or OSPF). In a stub AS like AS2, the border router typically injects a default route into the internal network, pointing to external destinations. In contrast, in a transient AS such as AS1, more detailed route information is injected into the intradomain routing protocol. This ensures that all routers within the AS have the necessary information to forward packets outside their local AS. Figure shows an example of how the forwarding tables are updated with routes from external ASs.

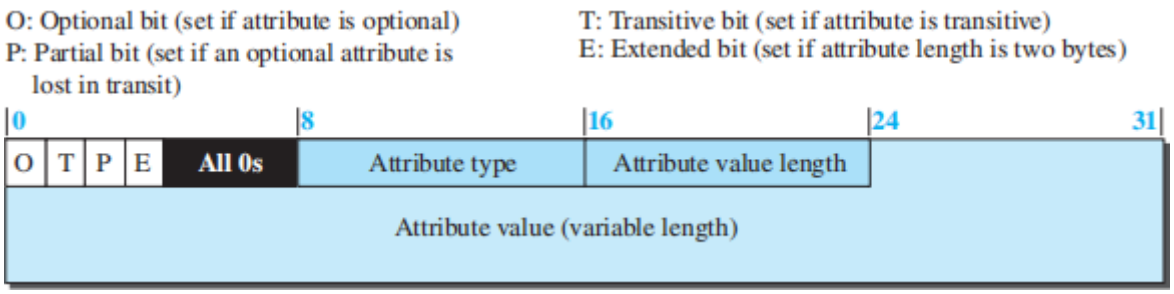


Address Aggregation

BGP employs techniques such as address aggregation to reduce the size of routing tables by summarizing multiple IP prefixes into a single route advertisement. This approach minimizes the number of entries in the forwarding table, optimizing router performance.

Path Attributes

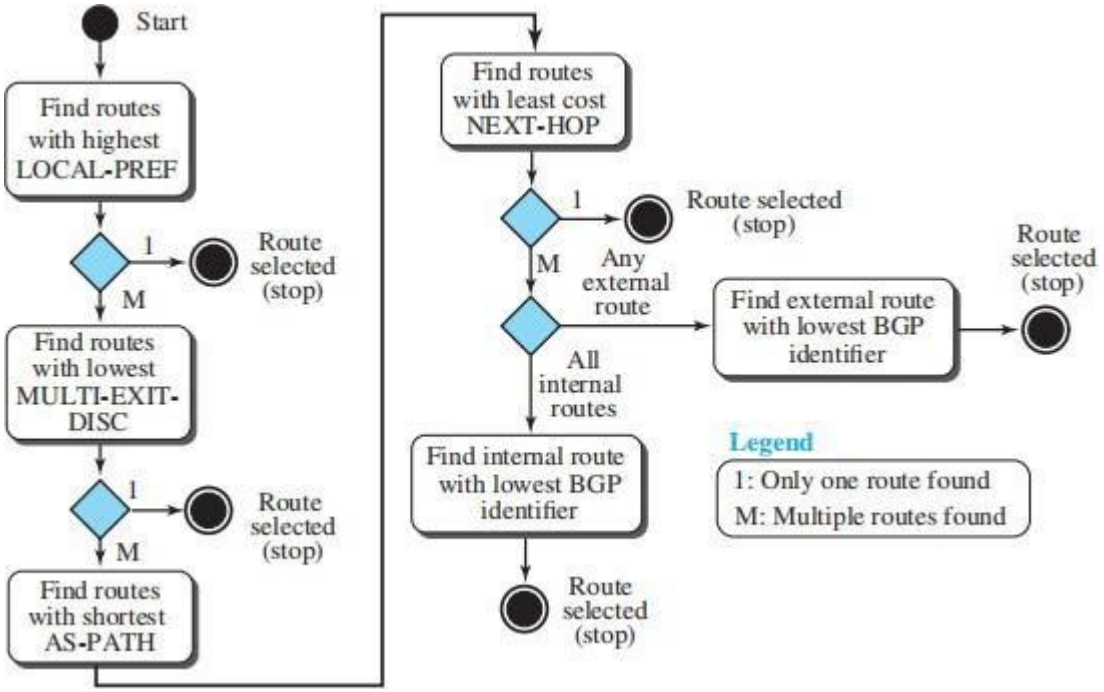
Additionally, BGP uses several path attributes to influence route selection and ensure routing stability. Important attributes include the AS-PATH, which lists all ASs a route has traversed, and the ORIGIN, which indicates how the route was learned. These attributes help routers make intelligent decisions about which paths to choose and avoid loops, as illustrated in Figure.



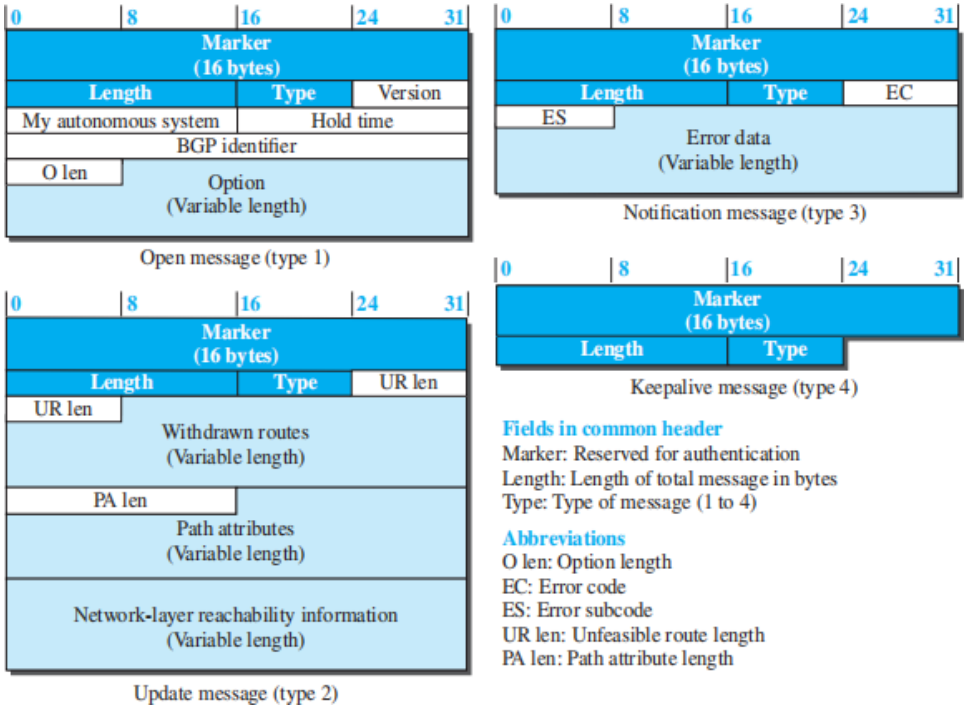
- **ORIGIN (type 1)**: A well-known mandatory attribute defining the source of routing information. Value 1 indicates an intradomain protocol (RIP or OSPF), Value 2 indicates BGP, and Value 3 indicates an unknown source.
- **AS-PATH (type 2)**: A well-known mandatory attribute listing the autonomous systems through which the destination can be reached. It helps prevent loops and assists in route selection.
- **NEXT-HOP (type 3)**: A well-known mandatory attribute defining the next router to forward the data packet. It injects path information into intradomain protocols such as RIP or OSPF.
- **MULT-EXIT-DISC (type 4)**: An optional intransitive attribute, used to select among multiple exit paths to a destination. Its value is a 4-byte unsigned integer, typically defined by the metric in intradomain protocols. The path with the lowest value is selected.
- **LOCAL-PREF (type 5)**: A well-known discretionary attribute used by administrators to prioritize routes. Routes with higher preference values are chosen based on organizational policy.
- **ATOMIC-AGGREGATE (type 6)**: A well-known discretionary attribute with no value field, indicating that the destination prefix is not aggregated.
- **AGGREGATOR (type 7)**: An optional transitive attribute, indicating that the destination prefix is an aggregate. The value specifies the AS number and IP address of the last router that performed the aggregation.

Route Selection in BGP

BGP route selection is complex compared to intradomain routing protocols, as it relies on attributes such as LOCAL-PREF and others. The process begins by extracting routes that meet specific criteria in each step. If one route remains, it is selected; otherwise, the next criteria are applied until one route is chosen.

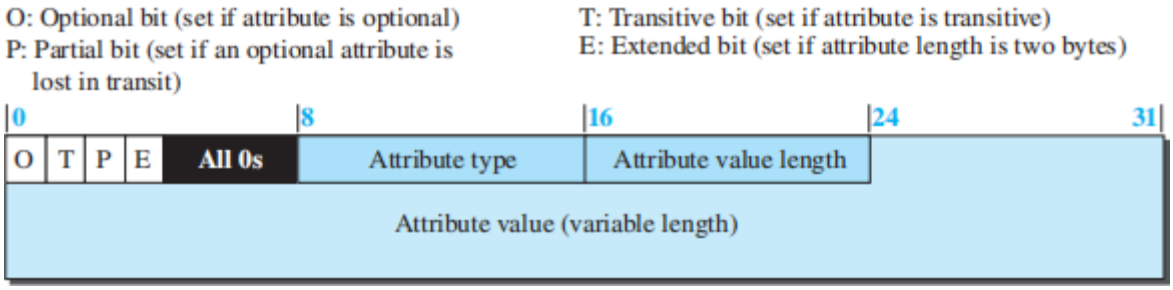


BGP Messages



- **Open Message**: Establishes a neighborhood relationship by initiating a TCP connection.
- **Update Message**: Announces new routes or withdraws previously advertised routes.
- **Keepalive Message**: Regularly exchanged between BGP peers to confirm they are still active.
- **Notification**: Sent when an error is detected or when a router wants to close the session.

Format of Path Attribute



- **Flags**:
  - O: Optional bit (set if the attribute is optional)
  - P: Partial bit (set if an optional attribute is lost in transit)
  - T: Transitive bit (set if the attribute is transitive)
  - E: Extended bit (set if the attribute length is two bytes)
- **Fields**:
  - Attribute type
  - Attribute value length
  - Attribute value (variable)

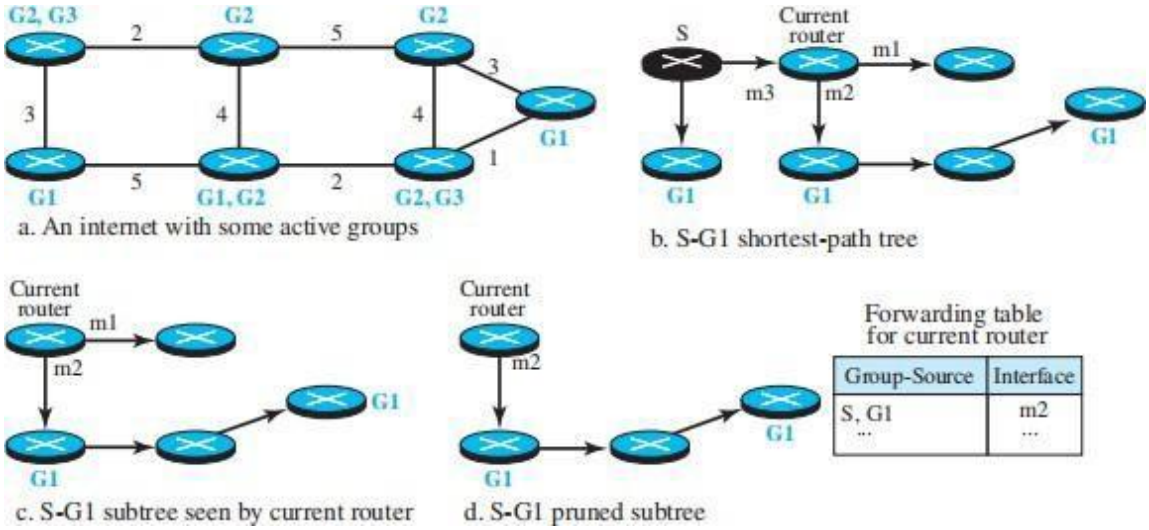
Multicast Link State (MOSPF)

Path First (OSPF) protocol, which is used in unicast routing. It also uses the source-based 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.

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 shows an example of using the steps to change a graph to a multicast tree. For simplicity, we have not shown the network, but we added the groups to each router. The figure shows how a source-based tree is made with the source as the root and changed to a multicast subtree with the root at the current router.



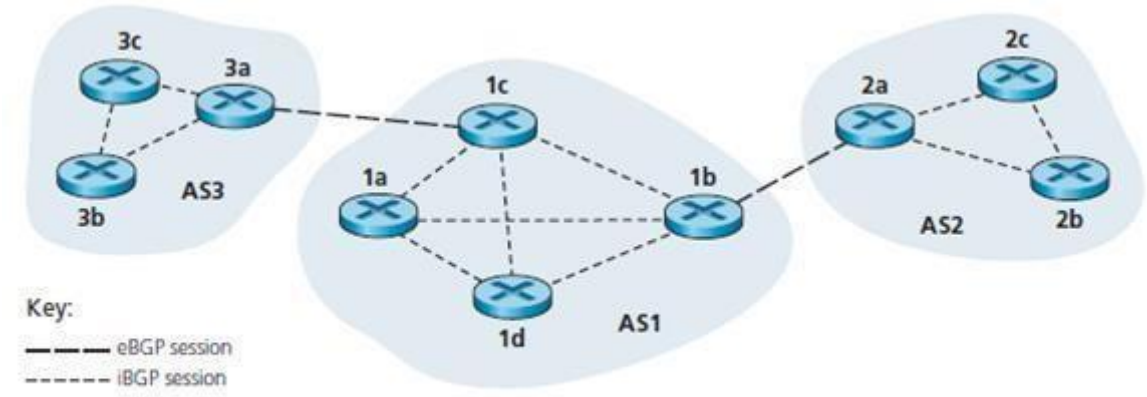
NOTE: Simpler Version of BGP(Not in Prescribed text book)

The Border Gateway Protocol version 4 is the standard inter-AS routing protocol in today's Internet.

- *It is commonly referred to as BGP4 or as BGP. It provides each AS a means to*

1. Obtain subnet reachability information from neighboring ASs.
2. Propagate the reachability information to all routers internal to the AS.
3. Determine "good" routes to subnets based on the reachability information and on AS policy.

- *In BGP, pairs of routers exchange routing information over semi-permanent TCP connections.*
- *In Figure 31, there is a TCP connection between gateway routers 3a and 1c and another TCP connection between gateway routers 1b and 2a.*
- *There are also semi-permanent BGP TCP connections between routers within an AS.*
- *For each TCP connection, the two routers at the end of the connection are called BGP peers, and the TCP connection along with all the BGP messages sent over the connection is called a BGP session.*
- *A BGP session that spans two Ass is called an external BGP (eBGP) session, and a BGP session between routers in the same AS is called an internal BGP (iBGP) session.*
- *Suppose there are four subnets attached to AS2: 138.16.64/24, 138.16.65/24, 138.16.66/24, and 138.16.67/24. Then AS2 could aggregate the prefixes for these four subnets and use BGP to advertise the single prefix to 138.16.64/22 to AS1.*



eBGP and iBGP sessions

Path Attributes and BGP Routes

When a router advertises a prefix across a BGP session, it includes with the prefix a number of BGP attributes. A prefix along with its attributes is called a route. Two important attributes are ***AS-PATH and NEXT-HOP:***

1. *AS-PATH. This attribute contains the ASs through which the advertisement for the prefix has passed. When a prefix is passed into an AS, the AS adds its ASN to the ASPATH attribute. For example in Figure 31, suppose the prefix 138.16.64/24 is first advertised from AS2 to AS1and then to AS3, AS-PATH would be AS2 AS1. Routers use the AS-PATH attribute to detect and prevent looping advertisements.*

2. Providing the critical link between inter-AS and intra-AS routing protocols. The NEXT-HOP is the router interface that begins the AS-PATH. In Figure 31, suppose the gateway router 3a in AS3 advertises a route to gateway router 1c in AS1 using eBGP. The route includes the advertised prefix, say x, and an AS-PATH to the prefix. This advertisement also includes the NEXT-HOP, which is the IP address of the router 3a interface that leads to 1c.

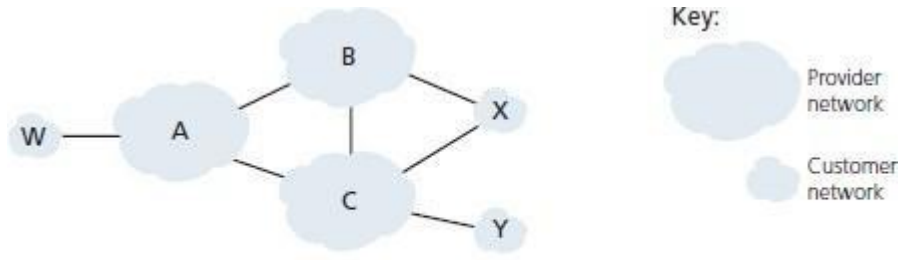
BGP Route Selection

If there are two or more routes to the same prefix, then BGP invokes the following elimination rules until one route remains:

- *Routes are assigned a local preference value as one of their attributes. The policy decision is left to the AS's network administrator. The routes with the highest local preference values are selected.*
- *From the remaining routes, the route with the shortest AS-PATH is selected. If this is the only rule for route selection, then BGP would use a DV algorithm for path determination, where the distance metric uses the number of AS hops.*
- *From the remaining routes, the route with the closest NEXT-HOP router is selected. It determined by the intra-AS algorithm by hot potato routing process.*

If more than one route still remains, the router uses BGP identifiers to select the route.

Routing Policy



A simple BGP scenario

- *Figure shows six interconnected autonomous systems: A, B, C, W, X, and Y.*
- *Assume that autonomous systems W, X, and Yare stub networks and that A, B, and C are backbone provider networks.*
- *Also assume that A, B, and C, all peer with each other, and provide full BGP information to their customer networks.*
- *All traffic entering a stub network must be destined for that network, and all traffic leaving a stub network must have originated in that network. W and Y are clearly stub networks.*
- *X is a multi-homed stub network, since it is connected to the rest of the network via two different providers.*
- *X is prevented from forwarding traffic between B and C by controlling the manner in which BGP routes are advertised.*
- *That is, even though X may know of a path, say XCY, that reaches network Y, it will not advertise this path to B.*
- *Since B is unaware that X has a path to Y, B would never forward traffic destined to Y (or C) via X.*

o This simple example shows how a selective route advertisement policy can be used to implement customer/provider routing relationships.