and routing principles shown in Figure 8.39 to explain how frames are relayed/routed between all the network sites. Include in your explanation the role of the following fields in each frame header:

(i) DLCI,
(ii) FECN, BECN and DE bits.

Explain the use of a CCLM frame and the actions carried out if a frame is corrupted.

8.42 With the aid of the network schematic shown in Figure 8.40, explain how both telephony (via a PBX) and data (via a remote bridge or router) are extended to cover an entire enterprise. Include in your explanation the role and operation of:

(i) a subrate multiplexer,
(ii) a frame relay adapter.

Can the latter also be used for telephony?

The Internet

9.1 Introduction

As we saw in Chapter 1, the Internet is a global network that supports a variety of interpersonal and interactive multimedia applications. A user gains access to these applications by means of an end system – normally referred to as a host – which, typically, is a multimedia PC, a network computer, or a workstation. As we showed in Figure 1.2, the Internet comprises a large number of different access networks which are interconnected together by means of a global internetwork. Associated with each access network – ISP network, intranet, enterprise network, site/campus LAN, and so on – is a **gateway** and the global internetwork consists of an interconnected set of regional, national, and international, networks all of which are interconnected together using high bit rate leased lines and devices known as **routing gateways** or simply **routers**.

The Internet operates in a packet-switched mode and Figure 9.1 shows the protocol stack associated with it. In the figure, we assume the network interface card in all hosts that are attached to an access network communicate with other hosts using the TCP/IP protocol stack. As we showed in Figure 5.11 (and explained in the accompanying text) this is not always the

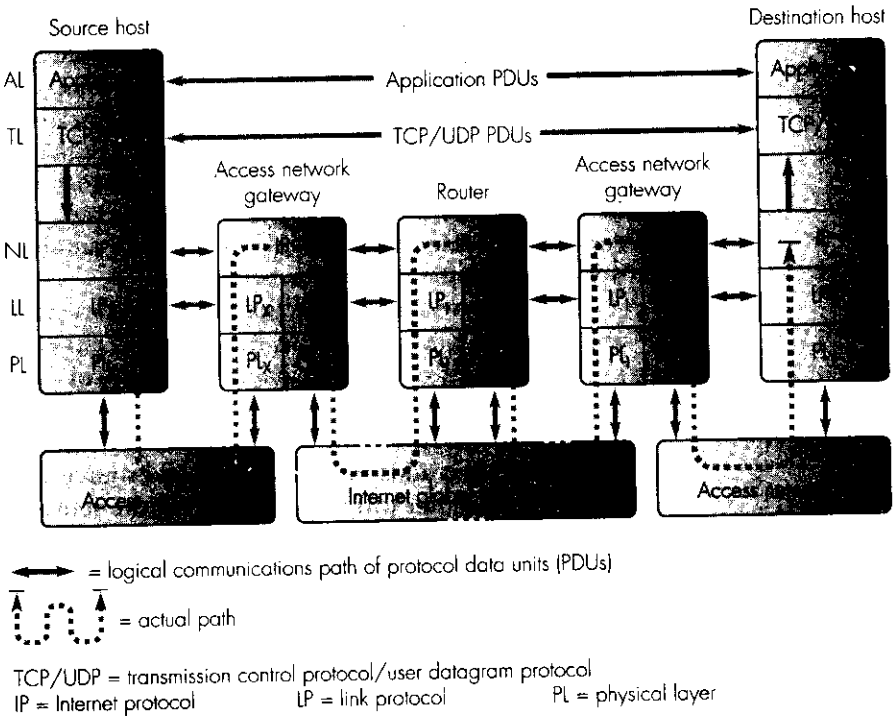


Figure 9.1 Internet networking components and protocols.

case. Nevertheless, any end system (host) that communicates directly over the Internet – the email server in Figure 5.11 for example – does so using the TCP/IP protocol stack.

In general, the various access networks have different operational parameters associated with them in terms of their bit rate, frame format, maximum frame size, and type of addresses that are used. For example, in the case of a site/campus LAN, as we saw in the last chapter, a token ring LAN uses a different bit rate, frame format, and maximum frame size from an Ethernet LAN. This means, therefore, that since bridges can only be used to interconnect LAN segments of the same type, they cannot be used to perform the network interconnection function. Hence instead, the routing and forwarding operations associated with a gateway are performed at the network layer. In the TCP/IP protocol stack the network layer protocol is the **Internet protocol** (**IP**) and, as we show in Figure 9.1, in order to transfer packets of information from one host to another, it is the IP in the two hosts, together with the IP in each Internet gateway and router involved, that perform the routing and other harmonization functions necessary.

The IP in each host (that communicates directly over the Internet) has a unique Internet-wide address assigned to it. This is known as the host's

Internet address or, more usually, its **IP address**. Each IP address has two parts: a **network identifier** (**netid**) and a **host identifier** (**hostid**). The allocation of netids is centrally managed by the **Internet Network Information Center** (**InterNIC**) and each access network has a unique netid assigned to it. For example, each campus/site LAN is assigned a single netid. The IP address of a host attached to an access network then contains the unique netid of the access network and a unique hostid. As with netids, hostids are centrally allocated but this time by the local administrator of the access network to which the host is attached.

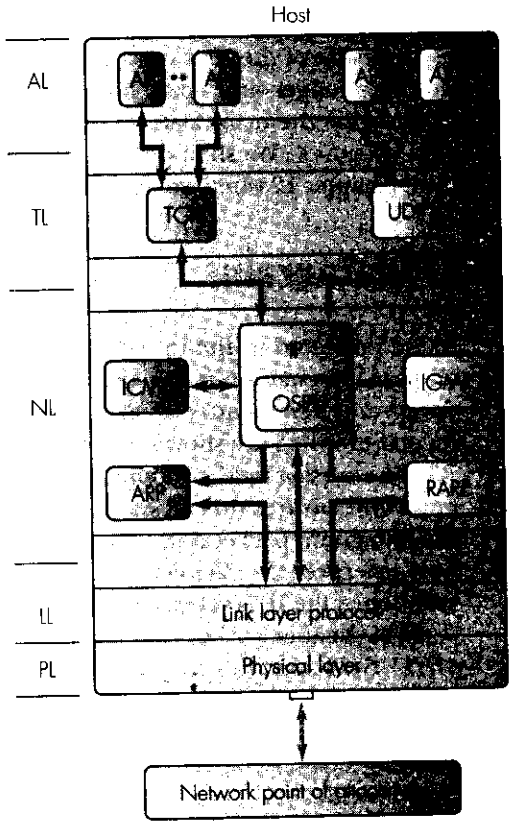
The IP provides a connectionless best-effort service to the transport layer above it which, as we show in the figure, is either the transmission control protocol (TCP) or the user datagram protocol (UDP). Hence when either protocol has a block of information to transfer, it simply passes the block to its local IP together with the IP address of the intended recipient. The (source) IP first adds the destination and source IP addresses to the head of the block, together with an indication of the source protocol (TCP or UDP), to form what is known as an **IP datagram**. The IP then forwards the datagram to its local gateway. At this point the datagram is often referred to as a **packet** and hence the two terms are used interchangeably.

Each access gateway is attached to an internetwork router and, at regular intervals, the IP in these routers exchange routing information. When this is complete, each router has built up a **routing table** which enables it to route a packet/datagram to any of the other networks/netids that make up the Internet. Hence, on receipt of a packet, the router simply reads the destination netid from the packet header and uses the contents of its routing table to forward the packet on the path/route through the global internetwork first to the destination internetwork router and, from there, to the destination access gateway. Assuming the size of the packet is equal to or less than the maximum frame size of the destination access network, on receipt of the packet, the destination gateway reads the hostid part of the destination IP address and forwards the packet to the local host identified by the hostid part. The IP in the host then strips off the header from the packet and passes the block of information contained within it – known as the **payload** – to the peer transport layer protocol indicated in the packet header.

If the size of the packet is greater than the maximum frame size – that is, the **maximum transmission unit** (**MTU**) – of the destination access network, the IP in the destination gateway proceeds to divide the block of information contained in the packet into a number of smaller blocks each known as a **fragment**. Each fragment is then forwarded to the IP in the destination host in a separate packet the length of which is determined by the MTU of the access network. The destination IP then reassembles the fragments of information from each received packet to form the original submitted block of information and passes this to the peer transport layer protocol indicated in the packet header.

As we shall see in the following sections, the above is just a summary of the operation of the IP and, in practice, in order to perform the various functions we have just described, the IP uses a number of what are known as **adjunct protocols.** These are identified in Figure 9.2 and a summary of the role of each protocol is as follows:

- **The address resolution protocol (ARP)** and the **reverse ARP (RARP)** are used by the IP in hosts that are attached to a broadcast LAN (such as an Ethernet or token ring) in order to determine the physical (MAC) address of a host or gateway given its IP address (ARP) and, in the case of the RARP, the reverse function.



AP = application protocol/process
IP = Internet protocol
ARP = address resolution protocol
RARP = reverse ARP
ICMP = Internet control message protocol
IGMP = Internet group message protocol
OSPF = open shortest path first

Figure 9.2 IP adjunct protocols.

- **The open shortest path first (OSPF)** protocol is an example of a routing protocol used in the global internetwork. Such protocols are present in each internetwork router and are utilized to build up the contents of the routing table that is used to route packets across the global internetwork.
- The **Internet control message protocol (ICMP)** is used by the IP in a host or gateway to exchange error and other control messages with the IP in another host or gateway.
- The **Internet group management protocol (IGMP)** is used with multicasting to enable a host to send a copy of a datagram to the other hosts that are part of the same multicast group.

In this chapter we explain the operation of the different parts of the IP and each of the adjunct protocols in some detail. Also, we describe the structure of the Internet global internetwork in more detail as this influences the overall routing strategy and routing protocols that are used. Currently, the most widely used version of the IP is version 4 and hence most of the chapter is devoted to this. In a longer time span, however, this will be replaced by version 6. Hence we describe the main features of this too and how it differs and interoperates with version 4. We shall defer discussion of the two transport layer protocols until Chapter 12 when we describe application protocols and related issues.

9.2 IP datagrams

As we indicated in the introduction, the IP is a connectionless protocol and all user information is transferred in the payload part of what is known as a datagram. The header of each datagram contains a number of fields the format of which are shown in Figure 9.3.

The *version* field contains the version of the IP used to create the datagram and ensures that all systems – gateways, routers, and hosts – that process the datagram/packet during its transfer across the Internet to the destination host interpret the various fields correctly. The current version number is 4 and hence the IP is referred to as **IP version 4** or simply **IPv4**.

The header can be of variable length and the *intermediate header length (IHL)* field specifies the actual length of the header in multiples of 32-bit words. The minimum length (without options) is 5. If the datagram contains options, these are in multiples of 32 bits with any unused bytes filled with **padding** bytes. Also, since the IHL field is 4 bits, the maximum permissible length is 15.

The *type of service (TOS)* field allows an application protocol/process to specify the relative priority (precedence) of the application data and the preferred attributes associated with the path to be followed. It is used by each gateway and router during the transmission and routing of the packet to transmit packets of higher priority first and to select a line/route that has the specified attributes should a choice be available. For example, if a route with

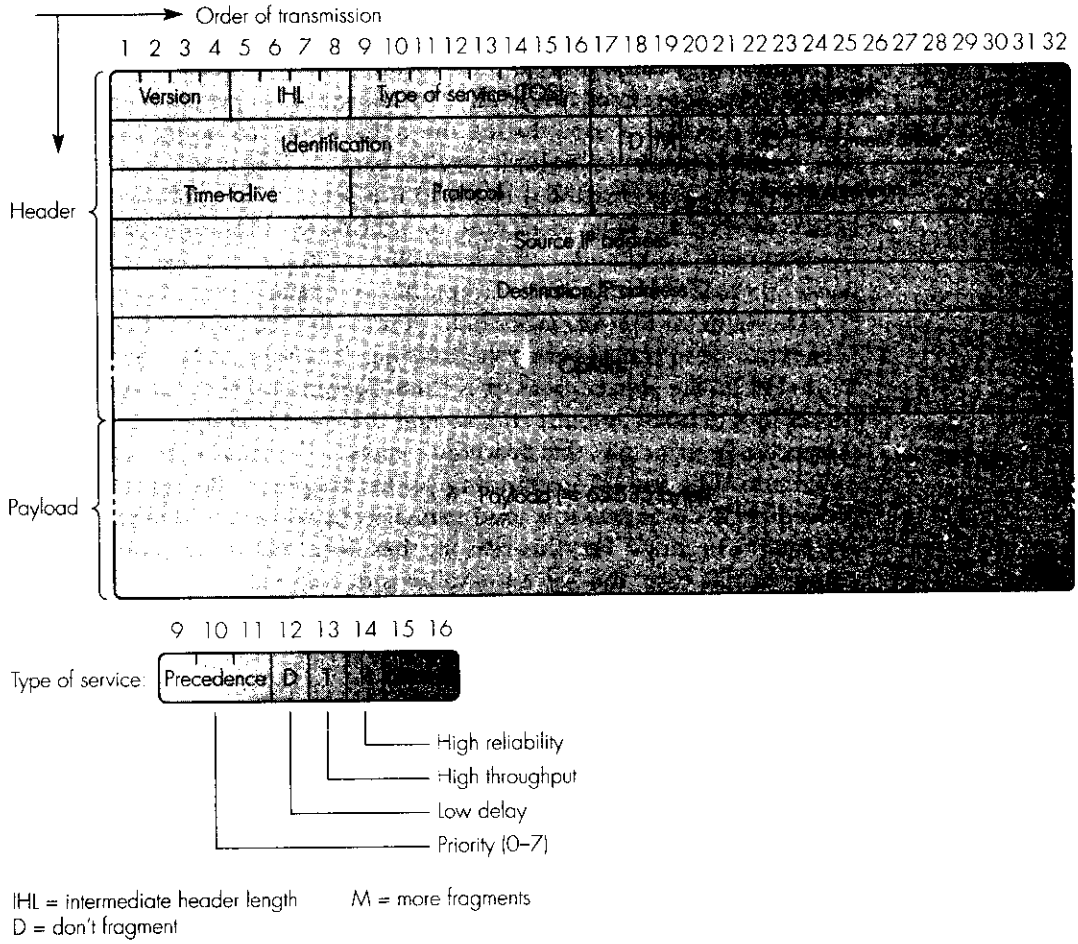


Figure 9.3 IP datagram/packet format and header fields.

a minimum delay is specified then, given a choice of routes, the line with the smallest delay associated with it should be chosen. We shall discuss the use of the TOS field further in Section 9.8.

The *total length* field defines the total length of the initial datagram including the header and payload parts. This is a 16-bit field and hence the maximum length is 65 535 (64K – 1) bytes and, as we explain in the next section, should the contents of the initial datagram need to be transferred in multiple (smaller) packets, then the value in *total length* is used by the destination host to reassemble the payload contained within each smaller packet – known as a fragment – into the original payload. In addition, each smaller packet contains the same value in the *identification* field to enable the destination host to relate each received packet fragment to the same original datagram.

The next three bits are known as *flag bits* of which two are currently used. The first is known as the *don't fragment* or *D-bit*. It is set by a source host and is examined by routers. A set D-bit indicates that the packet should be transferred in its entirety or not at all. The second is known as *more fragments* or *M-bit* and this also is used during the reassembly procedure associated with data transfers involving multiple smaller packets/fragments. It is set to 1 in all but the last packet/fragment in which it is set to 0. In addition, the *fragment offset* is used by the same procedure to indicate the position of the first byte of the fragment contained within a smaller packet in relation to the original packet payload. All fragments except the last one are in multiples of 8 bytes.

The value in the *time-to-live* field defines the maximum time for which a packet can be in transit across the Internet. The value is in seconds and is set by the IP in the source host. It is then decremented by each gateway and router by a defined amount and, should the value become zero, the packet is discarded. In principle, this procedure allows a destination IP to wait a known maximum time for an outstanding packet fragment during the reassembly procedure. In practice, it is used primarily by routers to detect packets that are caught in loops. For this reason, therefore, the value is normally a hop count. In this case, the *hop count* value is decremented by one by each gateway/router visited and, should the value become zero, the packet is discarded. We shall identify how looping can occur in Section 9.6 when we discuss the subject of routing.

The value in the *protocol* field is used to enable the destination IP to pass the payload within each received packet to the same (peer) protocol that sent the data. As we showed in Figure 9.2, this can be an internal network layer protocol such as the ICMP, or a higher-layer protocol such as TCP or UDP.

The *header checksum* applies just to the header part of the datagram and is a safeguard against corrupted packets being routed to incorrect destinations. It is computed by treating each block of 16 bits as an integer and adding them all together using 1s complement arithmetic. As we showed in the example in Figure 6.20(b), the checksum is then the complement (inverse) of the 1s complement sum.

The *source address* and *destination address* are the Internetwide IP addresses of the source and destination host respectively.

Finally, the *options* field is used in selected datagrams to carry additional information relating to:

- *security:* the payload may be encrypted for example, or be made accessible only to a specified user group. The *security* field then contains fields to enable the destination to decrypt the payload and authenticate the sender;
- *source routing:* if known, the actual path/route to be followed through the Internet may be specified in this field as a list of gateway/router addresses;
- *loose source routing:* this can be used to specify preferred routers in a path;

- *route recording:* this field is used by each gateway/router visited during the passage of a packet through the Internet to record its address. The resulting list of addresses can then be used, for example, in the source routing field of subsequent packets;
- *stream identification:* this, together with the source and destination addresses in the datagram header, enables each gateway/router along the path followed by the packet to identify the stream/flow to which the packet belongs and, if necessary, give the packet precedence over other packets. Examples include streams containing samples of speech or compressed video;
- *time-stamp:* if present, this is used by each gateway/router along the path followed by the packet to record the time it processed the packet.

9.3 Fragmentation and reassembly

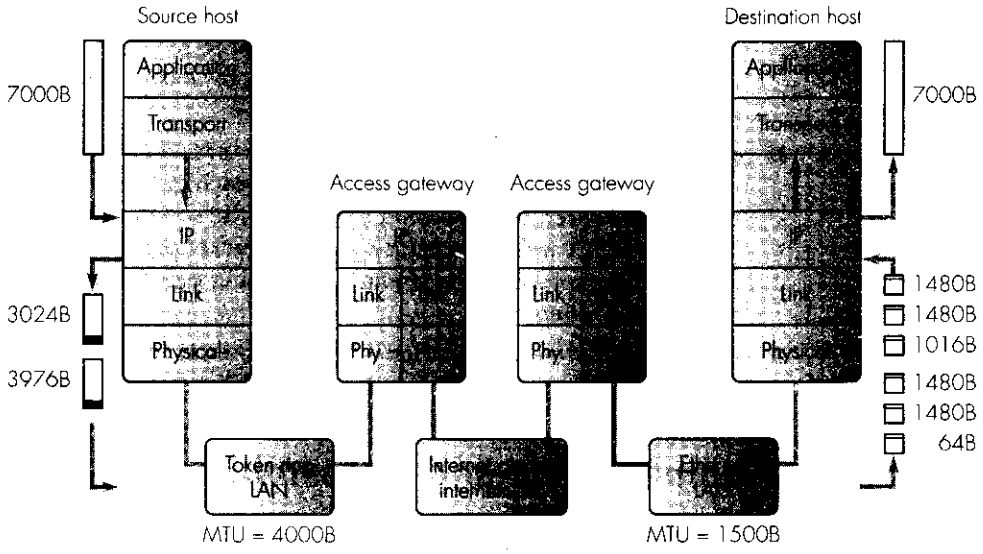
As we explained in Section 9.1, if the size of a packet is greater than the MTU of the destination access network – or an intermediate network in the global internetwork – the IP in the destination gateway – or intermediate router – divides the information received in the packet into a number of smaller blocks known as fragments. Each fragment is then forwarded to the IP in the destination host in a separate packet the length of which is determined by the MTU of the access/intermediate network. The IP in the destination host then reassembles the fragments of information from each received packet to form the original submitted block of information. It then passes this to the peer transport layer protocol indicated in the protocol field of the packet header.

To see how the various fields in each packet header are used to perform this function, consider the transport protocol in a host that is attached to a token ring LAN transferring a block of 7000 bytes – including the transport protocol header – over the Internet to the transport protocol in a host that is attached to an Ethernet LAN. Let us assume that the MTU associated with the token ring LAN is 4000 bytes and that of the Ethernet LAN 1500 bytes. Also that the header of each IP datagram requires 20 bytes. The steps taken to transfer the complete block of 7000 bytes are shown in Figure 9.4(a).

Since the header of each datagram requires 20 bytes, the maximum usable data in each token ring frame is 4000 – 20 = 3980 bytes. Similarly, that in each Ethernet frame is 1500 – 20 = 1480 bytes. However, all fragments of user data (except the last one) must be in multiples of 8 bytes. So we shall have to limit the maximum user data in each packet transferred over the token ring to 3976 bytes. In the case of the Ethernet, 1480 is divisible by 8 and so this value can be used unchanged.

To transfer the block of 7000 bytes over the token ring LAN requires two datagrams, one with 3976 bytes of user data and the second 7000 – 3976 = 3024 bytes. The values for the various fields associated with the fragmentation and reassembly procedures in each datagram header are given in Figure 9.4(b).

(a)



Note: All values shown are the amounts of user data in each packet/frame in bytes

(b) *Token ring LAN:*

	(i)	*(ii)*
Identification	20	20
Total length	7000	7000
Fragment offset	0	497
(User data)	3976	3024
M-bit	1	0

(c) *Ethernet LAN:*

	(i)	*(ii)*	*(iii)*	*(iv)*	*(v)*	*(vi)*
Identification	20	20	20	20	20	20
Total length	7000	7000	7000	7000	7000	7000
Fragment offset	0	185	370	497	682	867
(User data)	1480	1480	1016	1480	1480	64
M-bit	1	1	1	1	1	0

Figure 9.4 Fragmentation and reassembly example: (a) Internet schematic; (b) packet header fields for token ring LAN; (c) Ethernet LAN.

The value in the *identification* field is the same in all fragments and is used by the destination IP to relate each fragment to the same original block of information. In the example, we assume a value of 20 has been allocated by the IP in the source host.

The *total length* is the number of bytes in the initial datagram including the 20-byte header. However, since we have subtracted 20 from the maximum user data value associated with each LAN, we have shown this as 7000. Note that this is the same in all the datagram fragments and hence the destination IP can readily determine when all fragments have been received. The

fragment offset then indicates the position of the user data in each fragment – in multiples of 8 bytes – relative to the start of the initial datagram. Finally, the *more fragments* (*M*) *bit* is 1 in each fragment and 0 in the final fragment.

We assume that the two datagrams/packets created by the source IP are transferred over the global internetwork unchanged and, on reaching the access gateway attached to the Ethernet LAN, the smaller maximum user data value of 1480 bytes means that both packets must be further fragmented. As we show in Figure 9.4(c), both packets must be fragmented into three smaller packets. The first into two maximum sized packets of 1480 bytes and a further packet containing 3976 – 2(1480) = 1016 bytes, and the second containing two maximum sized packets and a further packet containing 3024 – 2(1480) = 64 bytes. The IP in the destination host then reassembles the user data in each of the six packets it receives into the original 7000-byte block of information and delivers this to the peer transport protocol.

As we explained, the *time-to-live* field in each packet header – and fragment header – is present to avoid packets endlessly looping around the Internet (normally as a result of routing table inconsistencies) and also to set a maximum limit on the time a host needs to wait for a delayed/corrupted/discarded datagram fragment. Hence although the use of fragmentation would appear to be relatively straightforward, there are drawbacks associated with its use. For example, as we shall explain in Section 12.3.2, with the TCP transport protocol, if an acknowledgment of correct receipt of a submitted block is not received within a defined maximum time limit, the source TCP will retransmit the complete block. Thus, as we can deduce from the example in Figure 9.4, it only needs one of the six datagram fragments to be delayed or discarded to trigger the retransmission of the complete 7000-byte block. As a result, therefore, most TCP implementations avoid the possibility of fragmentation occurring by limiting the maximum submitted block size – including transport protocol header – to 1048 bytes or, in some instances, 556 bytes. Alternatively, as we shall expand upon in Section 9.7, it is possible for the source IP, prior to sending any transport protocol (user) data, to determine the MTU of the path to be followed through the Internet. Then, if this is smaller than the submitted user data, the source IP fragments the data using this MTU. In this way, no further fragmentation should be necessary during the transfer of the packets through the global internetwork.

9.4 IP addresses

Each host, gateway, and router has a unique Internetwide IP address assigned to it that comprises a netid and a hostid part. Hence in the case of gateways and routers that interconnect two (or more) networks together, each gateway/router port – also referred to as an interface – has a different netid associated with it. In order to give the Internet Network Information Center some flexibility when assigning netids, one of three different address formats

can be used. Each format is known by an **address class** and, as we show in Figure 9.5, classes A, B, and C are all different types of unicast addresses.

Each of these classes is intended for use with a different size of network. For example, at one extreme a large national network and at the other a small site LAN. The class to which an address belongs can be determined from the position of the first zero bit in the first four bits. The remaining bits then specify the netid and hostid parts with the boundary separating the two parts located on byte boundaries to simplify decoding.

Class A addresses have 7 bits for the netid and 24 bits for the hostid; class B addresses have 14 bits for the netid and 16 bits for the hostid; and class C addresses have 21 bits for the netid and 8 bits for the hostid. Class A addresses are intended for use with networks that have a large number of attached hosts (up to 2^{24}) while class C addresses allow for a large number of networks each with a small number of attached hosts (up to 256). An example of a class A network is a large national network and an example of a class C network is a small site LAN.

Netids and hostids comprising either all 0s or all 1s have special meaning:

- An address with a hostid of all 0s is used to refer to the network in the netid part rather than a host.
- An address with a netid of all 0s implies the same network as the source network/netid.

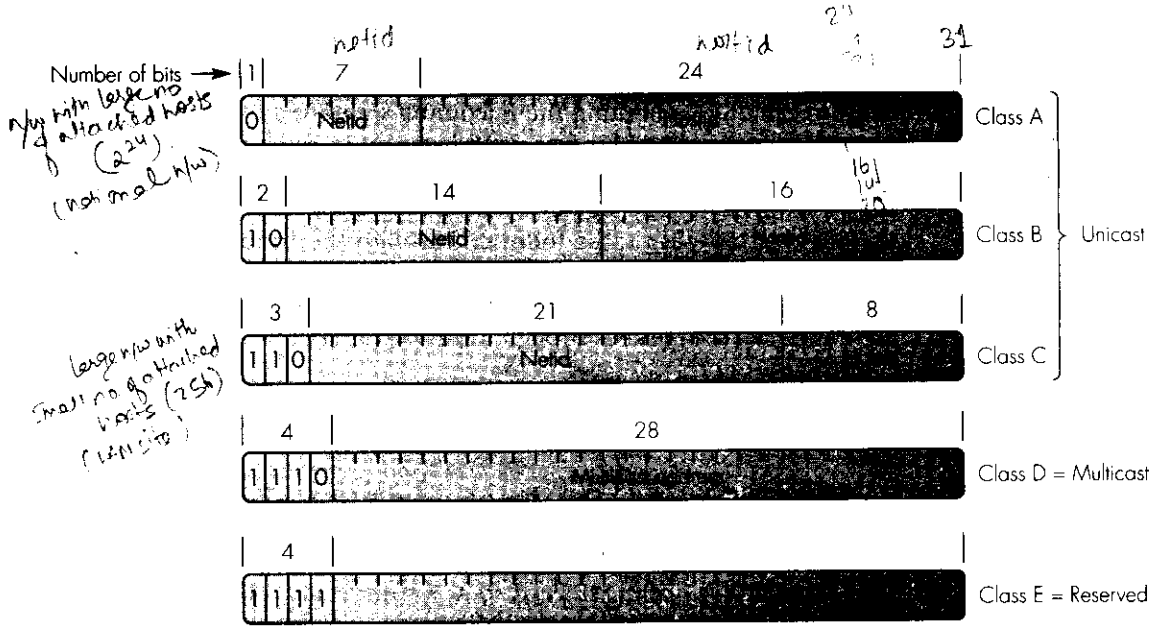


Figure 9.5 IP address formats.

- An address of all 1s means broadcast the packet over the source network.
- An address with a hostid of all 1s means broadcast the packet over the destination network in the netid part.
- A class A address with a netid of all 1s is used for test purposes within the protocol stack of the source host. It is known, therefore, as the **loopback address.**

To make it easier to communicate IP addresses, the 32 bits are first broken into four bytes. Each byte is then converted into its equivalent decimal form and the total IP address is represented as the four decimal numbers with a dot (period) between each. This is known as the **dotted decimal notation** and some examples are as follows:

00001010 00000000 00000000 00000000 = 10.0.0.0.
= class A, netid 10

10000000 00000011 00000010 00000011 = 128.3.2.3
= class B, netid 128.3, hostid 2.3

11000000 00000000 00000001 11111111 = 192.0.1.255
= class C, all hosts broadcast on netid 192.0.1

Example 9.1

Assuming the IP address formats shown in Figure 9.5, derive the range of host addresses for classes A, B, and C. Give your answer in dotted decimal notation and also straight decimal.

Answer:

Class A:
Netid = 1 to 127 Hostid = 0.0.0 to 255.255.255
= 126 networks = 16 777 214 hosts

Class B:
Netid = 128.0 to 191.255 Hostid = 0.0 to 255.255
= 16 382 networks = 65 534 hosts

Class C:
Netid = 192.0.0 to 223.255.255 Hostid = 0 to 255
= 2 097 152 networks = 254 hosts

Note that we have not used hostids of all 0s or all 1s.

Class D addresses are reserved for multicasting. As we explained in Section 8.2, in a LAN a frame can be sent to an individual, broadcast, or group address. The group address is used by a station to send a copy of the frame to all stations that are members of the same multicast group (and hence have the same multicast address). In the case of LANs, the group address is a MAC address and the class D IP address format is provided to enable this mode of working to be extended over the complete Internet.

As we can see, unlike the three unicast address classes, the 28-bit multicast group address has no further structure. As we shall expand upon in Section 9.6.9, multicast group addresses are assigned by the **Internet assigned numbers authority** (*IANA*). Although most of these are assigned dynamically (for conferences and so on), some are reserved to identify specific groups of hosts and/or routers. These are known as **permanent multicast group addresses**. Examples are 224.0.0.1 which means all hosts (and routers) on the same broadcast network, and 224.0.0.2 which means all routers on the same site network.

9.4.1 Subnets

Although the basic structure shown in Figure 9.5 is adequate for most addressing purposes, the introduction of multiple LANs at each site can mean unacceptably high overheads in terms of routing. As we described in Section 8.5, MAC bridges are used to interconnect LAN segments of the same type. This solution is attractive for routing purposes since the combined LAN then behaves like a single network. When interconnecting dissimilar LAN types, as we explained in Section 8.8.4, the differences in frame format and, more importantly, frame length, mean that routers are normally used since the fragmentation and reassembly of packets/frames is a function of the network layer rather than the MAC sublayer. However, the use of routers means that, with the basic address formats, each LAN must have its own netid. In the case of large sites, there may be a significant number of such LANs.

This means that with the basic addressing scheme, all the routers relating to a site need to take part in the overall Internet routing function. As we shall expand upon in Section 9.5, the efficiency of any routing scheme is strongly influenced by the number of routing nodes that make up the Internet. The concept of **subnets** has been introduced to decouple the routers – and hence routing – associated with a single site from the overall routing function in the global internetwork. Essentially, instead of each LAN associated with a site having its own netid, only the site is allocated a netid. Each LAN is then known as a **subnet** and the identity of each (LAN) subnet then forms part of the hostid field. This refined address format is shown in Figure 9.6(a).

The same address classes and associated structure are used, but the netid now relates to a complete site rather than to a single subnet. Hence, since only a single gateway/router attached to a site performs internetwide routing, the netid is considered as the **Internet part.** For a single netid with a

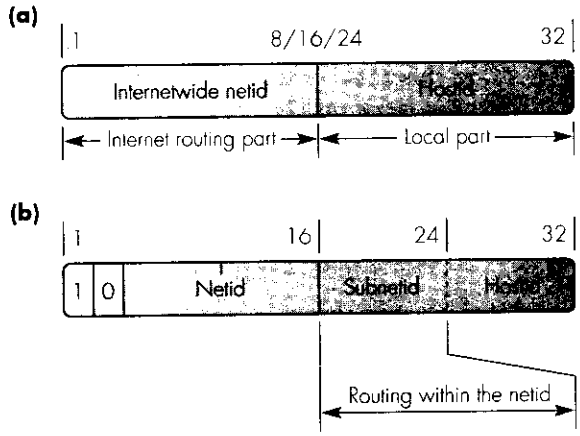


Figure 9.6 Subnet addressing: (a) address structure; (b) example.

number of associated subnets, the hostid part consists of two subfields: a **subnetid part** and a **local hostid part.** Because these have only local significance, they are known collectively as the **local part.** Also, to discriminate between the routers in the global internetwork and those in a local site network, the latter are known as **subnet routers.**

Because of the possibly wide range of subnets associated with different site networks, no attempt has been made to define rigid subaddress boundaries for the local address part. Instead, an **address mask** is used to define the subaddress boundary for a particular network (and hence netid). The address mask is kept by the site gateway and all the subnet routers at the site. It consists of binary 1s in those bit positions that contain a network address – including the netid and subnetid – and binary 0s in positions that contain the hostid. Hence an address mask of

11111111 11111111 11111111 00000000

means that the first three bytes (octets) contain a network/subnet identifier and the fourth byte contains the host identifier.

For example, if the mask relates to a class B address – a zero bit in the second bit position – this is readily interpreted as: the first two bytes are the internetwide netid, the next byte the subnetid, and the last byte the hostid on the subnet. Such an address is shown in Figure 9.6(b).

Normally, dotted decimal is used to define address masks and hence the above mask is written:

255.255.255.0

Byte boundaries are normally chosen to simplify address decoding. So with this mask, and assuming the netid was, say, 128.10, then all the hosts attached to this network would have this same netid. In this way, the presence of a possibly large number of subnets and associated (subnet) routers is transparent to all the other Internet gateways and routers for routing purposes.

Example 9.2

The administrator of a campus LAN is assigned a single class B IP address of 150.10.0.0. Assuming that the LAN comprises 100 subnets, each of which is connected to an FDDI backbone network using a subnet router, define a suitable address mask for the LAN if the maximum number of hosts connected to each subnet is 70.

Answer:

A class B IP address means that both the netid and the local part are each 16 bits. Hence the simplest way of meeting this requirement is to divide the local part into two: 8 bits for the subnet and 8 bits for the hostid.

This will allow for up to 254 subnets and 254 hosts, excluding all 1s and all 0s.

The address mask, therefore, is 255.255.255.0

9.5 ARP and RARP

As we outlined in Section 9.1, the address resolution protocol (ARP) and the reverse ARP (RARP) are used by the IP in hosts that are attached to a broadcast LAN. The ARP is used to determine the MAC address of another host or gateway that is attached to the same LAN given the IP address of the host gateway. It is defined in **RFC 826**. The RARP performs the reverse function and is defined in **RFC 903**. We explain the operation of each separately.

9.5.1 ARP

Associated with each host are two addresses: its IP address and its MAC address which, since it is assigned to the MAC integrated circuit when it is manufactured, is known also as the host's hardware (or physical) address. Normally, both addresses are stored in the configuration file of the host on the hard disk. In order to describe the operation of the ARP, we shall use the LAN topology shown in Figure 9.7.

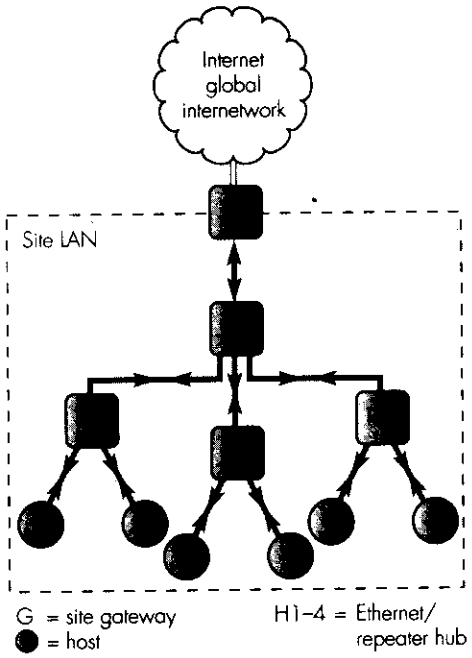


Figure 9.7 Example topology for describing the operation of the ARP.

As we can see, this comprises three Ethernet hubs (H1, H2, and H3) that are interconnected by means of a fourth hub (H4). There is also a connection between H4 and the site gateway (G). We assume that all the hubs are simple repeater hubs and that all hosts have just been switched on and hence have sent no frames. Associated with each ARP is a routing table known as the **ARP cache**. This contains a list of the IP/MAC address-pairs of those hosts with which host A has recently communicated and, when the host is first switched on, it contains no entries. First we shall explain the steps taken by the ARP in host A to send a datagram to another host on the same LAN – host B – and then to a host on a different LAN via the gateway.

On receipt of the first datagram from the IP in host A, the ARP in A reads the destination IP address of B contained in the datagram header and determines it is not in its cache. Hence it broadcasts an **ARP request message** in a broadcast frame over the LAN and waits for a reply. The request message contains both its own IP/MAC address-pair and the (target) IP address of the destination, host B. Being a broadcast frame this is received by the ARP in all hosts attached to the LAN.

The ARP in host B recognizes its own IP address in the request message and proceeds to process it. It first checks to see whether the address-pair of the source is within its own cache and, if not, enters them. This is done since it is highly probable that the destination host will require the MAC address of

the source when the higher-layer protocol responds to the message contained within the datagram payload. The ARP in host B then responds by returning an **ARP reply** message (containing its own MAC address) to the ARP in host A using the latter's MAC address contained in the request message. On receipt of the reply message, the ARP in host A first makes an entry of the requested IP/MAC address-pair in its own cache and then passes the waiting datagram to either the LLC sublayer (if one is present) or (if not) to the MAC sublayer together with the MAC address of host B which indicates where it should be sent. At B the datagram is passed directly to the IP for processing.

Being on the same broadcast network as all the site hosts, the LAN port of the gateway receives a copy of all broadcast frames containing ARP request and reply messages. On receipt of each message, the ARP first checks to see if it has the IP/MAC address-pair(s) contained in the message in its cache and, if not, adds them to the cache. In this way, the site gateway learns the address-pair of all the hosts that are attached to the site LAN.

To send a datagram from, say, host A to a host on a different LAN – and hence netid – the ARP in A broadcasts the request message as before. In this case, however, on receipt of the message, the gateway determines that the netid part of the destination IP address relates to a different network and responds by returning a reply message containing its own address-pair. Hence A makes an entry of this in its cache and proceeds to forward the datagram to the gateway as if it was the destination host. The gateway then forwards the datagram/packet over the Internet using one of the global internetwork routing protocols we shall describe later in Section 9.6. The ARP in the gateway is known as a **proxy ARP** since it is acting as an agent for the ARP in the destination host.

When the gateway receives the response packet from the destination host, it reads the destination IP address from the header – host A – and obtains from its cache the MAC address of A. It then transfers the packet to the IP in A using the services provided by the MAC sublayer. A similar procedure is followed for bridged LANs and for router-based LANs except that with the latter, the ARP in the subnet router that is connected to the same subnet as the source host acts as the proxy ARP. Also, in order to allow for hosts to change their network point of attachments, entries in the ARP cache timeout after a predefined time interval.

9.5.2 RARP

As we indicated at the start of the last section, normally the IP/MAC address-pair of a host is stored in the configuration file of the host on its hard disk. With diskless hosts, clearly this is not possible and hence the only address that is known is the MAC address of the MAC chipset. In such cases, therefore, the alternative reverse address resolution protocol (RARP) is used.

The server associated with a set of diskless hosts has a copy of the IP/MAC address-pair of all the hosts it serves in a configuration file. When a

diskless host first comes into service, it broadcasts a **RARP request** message containing the MAC address of the host onto its local LAN segment. Being a broadcast, the server receives this and, on determining it is a RARP message, the MAC/LLC sublayer passes the message to the RARP. The latter first uses the MAC address within it to obtain the related IP address from the configuration file and then proceeds to create a **RARP reply** message containing the IP address of the host and also its own address-pair. The server then sends the reply message back to the host and, once its own address-pair is known, the ARP in the diskless host can proceed as before.

9.5.3 ARP/RARP message formats and transmission

As we show in Figure 9.8(a), the format of the ARP and RARP request and reply messages are the same both having a fixed length of 28 bytes. Note that the term "hardware address" is used to refer to a MAC address and "target" the recipient of a request.

The *hardware type* field specifies the type of hardware (MAC) address contained within the message, for example 0001 (hex) in the case of an Ethernet. The ARP and RARP can be used with other network protocols (as well as the IP) and the *protocol type* indicates the type of network address being resolved; for example, 0800 (hex) is used for IP addresses. The *HLEN* and *PLEN* fields specify the size in bytes of the hardware (MAC) and protocol (IP) address lengths respectively, for example 06 (hex) for an Ethernet and 04 (hex) for the IP. The *operation* field indicates whether the message (resolution operation) is an ARP request (0001) or reply (0002) or a RARP request (0003) or reply (0004).

The next four fields specify the hardware (MAC)/IP address-pair of the sender (source) and the target (destination). For example, in an ARP request message just the address-pair of the sender is used while in the reply message all four addresses are used.

As we indicated in the introduction, both the ARP and RARP are integral parts of IP inasmuch as, once the MAC address relating to an IP address is present in the ARP cache, the IP can use this to initiate the transmission of a datagram to its intended recipient directly. Hence, as we can deduce from Figure 9.2, on receipt of a frame, the receiving MAC/LLC sublayer must be able to determine to which protocol the frame contents should be sent: IP, ARP, or RARP. To achieve this, when each of the protocols passes a message/datagram to the LLC/MAC sublayer for transmission, in addition to the MAC address of the intended recipient, it specifies the name of the (peer) protocol in the destination (IP/ARP/RARP) to which the message/datagram should be passed.

As we show in Figure 9.8(b), this is specified in a two-byte *type* field which immediately precedes the message/datagram in the user data field of the MAC frame. As we explained in Section 8.8.3, normally, the LLC sublayer is not used with the original Ethernet MAC standard and hence the *type* field

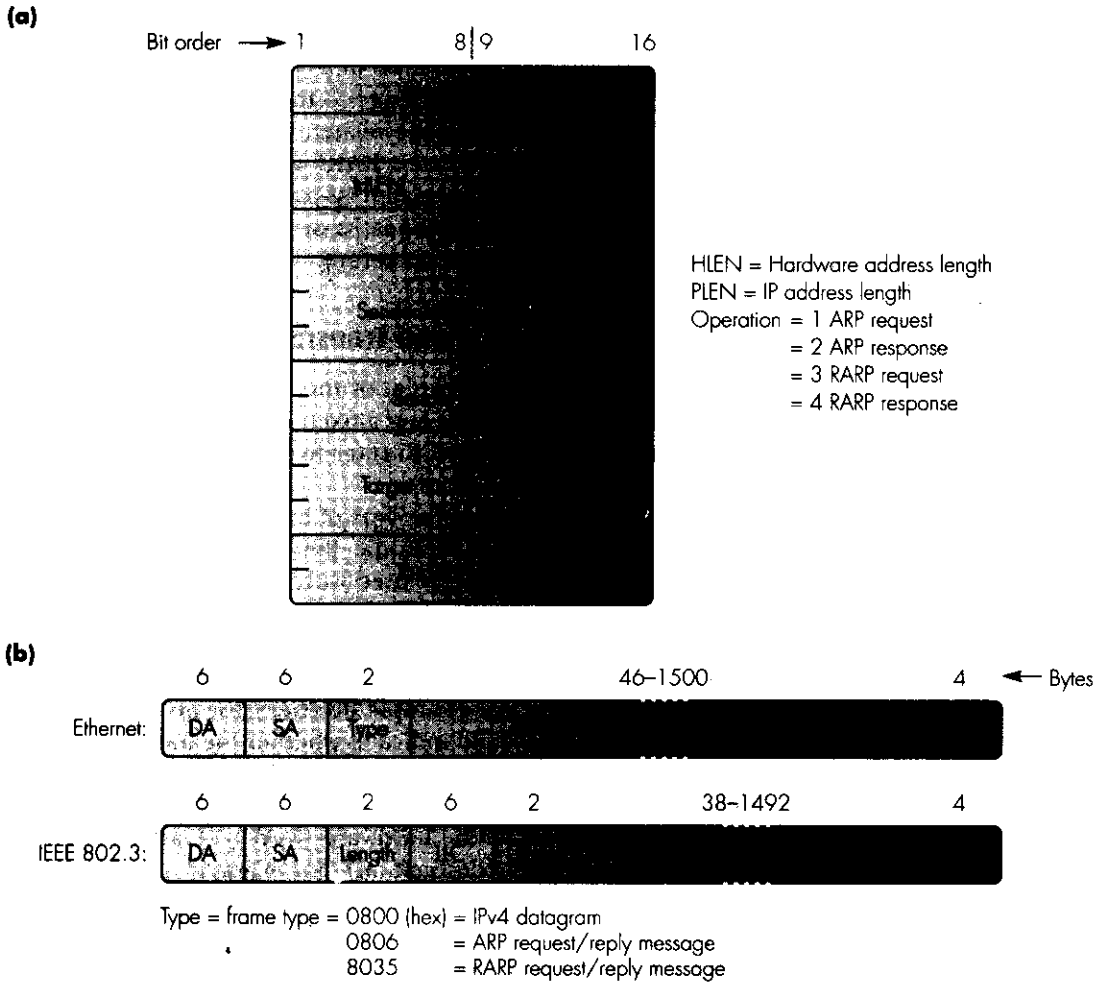
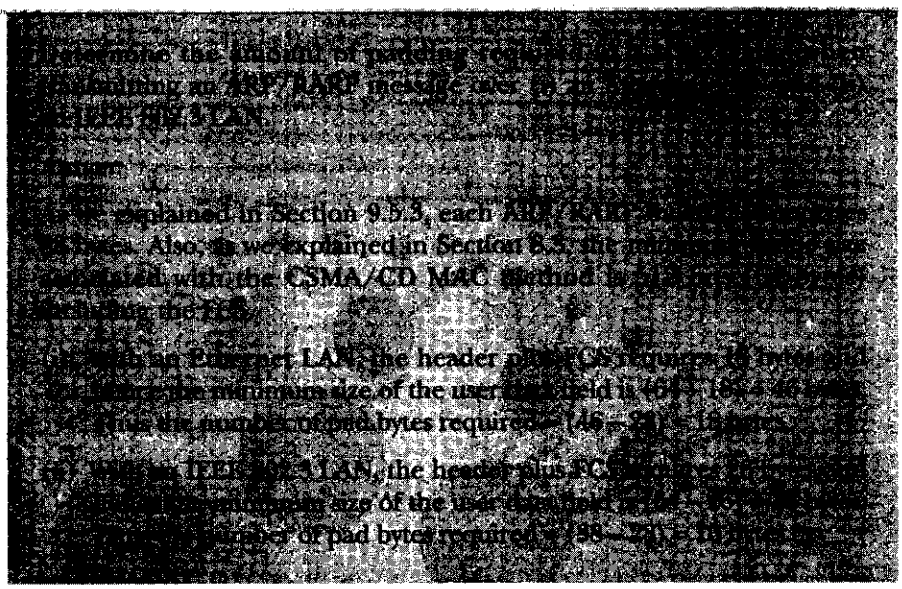


Figure 9.8 ARP and RARP message formats and transmission: (a) ARP and RARP message formats; (b) MAC frame format with Ethernet and IEEE802.3.

immediately follows the MAC *source address* (*SA*). With the more recent IEEE 802.3 MAC standard, however, the LLC sublayer is present and hence the *type* field immediately follows the 6 bytes required for the LLC protocol. However, the maximum frame length associated with the 802.3 standard (1500 bytes) means that the length indicator value in the header is always different from the three *type* field values. Hence the receiving MAC sublayer can readily determine which of the two standards is being used and, therefore, where the *type* field is located. With a token ring LAN, since there is only one standard, this problem does not arise.

Example 9.3



9.6 Routing algorithms

The Internet comprises many thousands of access networks that are geographically distributed around the world. As a result, the global internetwork used to interconnect the gateways associated with these (access) networks is not a single network as such but rather a collection of different types of network that have evolved over the lifetime of the Internet. The general architecture of the Internet is shown in Figure 9.9.

As we can see, the various networks making up the Internet global internetwork we showed earlier in Figure 9.1 are in a hierarchical structure. At the highest level, each continent has its own **continental backbone network.** This comprises a geographically distributed set of very high throughput routers which are interconnected by very high bit rate lines leased from one or more network providers. The continental backbones are then interconnected together by means of a **core backbone network** consisting of a small number of very high throughput (intercontinental) routers and leased lines.

At the second level, attached to all of the other routers in each continental backbone are a number of regional and national networks and, in some instances, large internetworks. In general, these also consist of a geographically distributed set of high throughput routers interconnected by high bit rate leased lines. Because of the historical evolution of the Internet, however, in addition, there are a number of legacy networks that operate using a different protocol stack from TCP/IP.

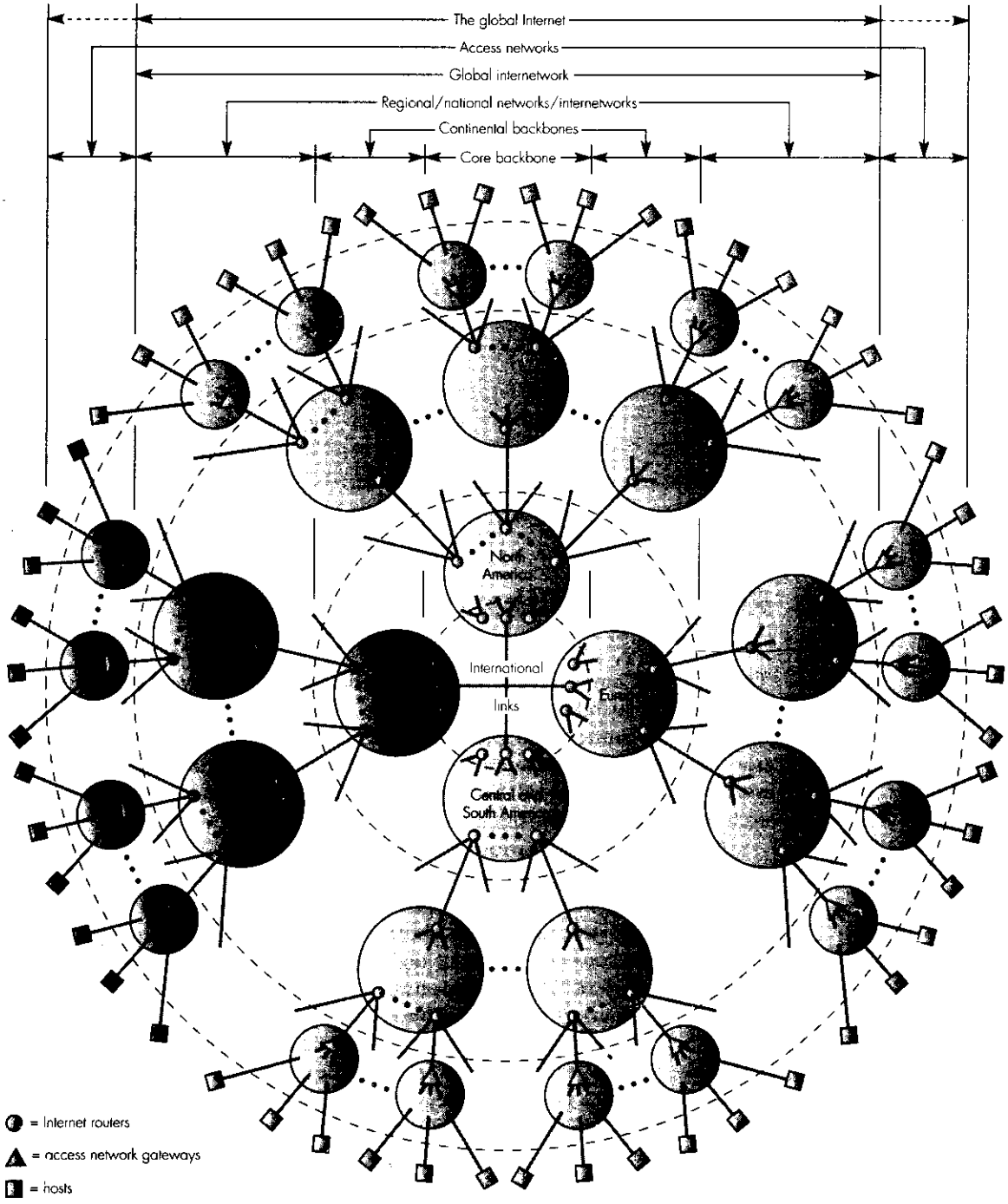


Figure 9.9 General architecture of the global Internet.

The various types of access networks – site/campus LANs, ISP networks, enterprise networks – are then attached to one of the regional or national networks/internetworks by means of either an **IP access network gateway** or, if the access network uses a different protocol stack from TCP/IP, a **multiprotocol gateway.**

All routing within the total global internetwork is carried out by the IP in each router using the netid part of the destination IP address in each datagram header. In practice, as we shall see in the following subsections, there are only a small number of different routing algorithms used and, in order to explain their principle of operation, we shall use the simple internetwork topology shown in Figure 9.10.

As we can see, each of the routers in the interconnection network has a number of access networks attached to it by means of (access) gateways. We assume that each access network is a site/campus LAN with a single netid and that the ARP in each host is used to carry out routing within the access network. Hence, as we explained in Section 9.5.1, the ARP in each gateway acts as an agent (proxy ARP) on behalf of all hosts at the site/campus to relay packets to and from the interconnection network.

The interconnection network itself comprises four routers (R1 – R4) that are interconnected by, say, leased lines. For description purposes, each line has a pair of numbers associated with it. The first we shall use as a line identifier and the second is what is referred to as the **cost** of the line. For example, the cost could be based on the line bit rate and, normally, the higher the line

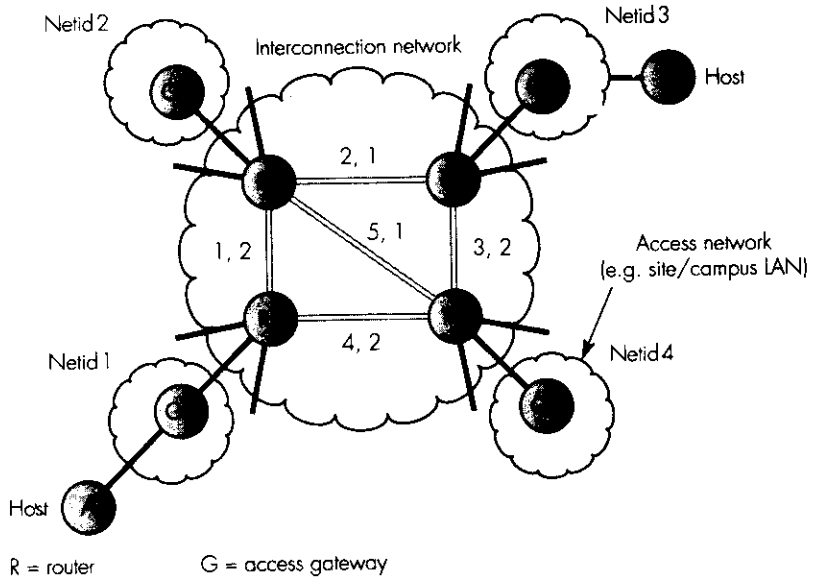


Figure 9.10 Example internetwork topology.

bit rate the lower the cost value. As we shall see, the cost value associated with each line is used during the routing of datagrams/packets and hence is also known as a **routing metric.** The cost of a route/path through the interconnection network is determined by summing together the cost value associated with each line that makes up the path. This is known as the **path cost** and, when different paths between two routers are available, the path with the least path cost value is known as the **shortest path.** Other metrics used in relation to the computation of the shortest paths are based on the physical length – and hence propagation delay – of each line, the number of lines (**hops**) in the route (**hop count**), and the mean queuing delay within each router associated with a line.

Let us assume that host A on netid 1 wants to send a datagram to host B on netid 3. As we saw in Section 9.5.1, on determining that the destination netid in the datagram header is for a different netid from its own, gateway G1 forwards the datagram to router R1 over the connecting line/link. On receipt of the datagram, R1 proceeds to forward it first to R3 over the interconnection network and then to G3. At this point, the IP in G3 knows how to route the datagram to host B using the hostid part of the destination IP address and the related MAC address of B in its ARP cache. What we do not know, is how the datagram is routed across the interconnection network.

There are three unanswered questions involved:

(1) How does R1 know from the netid contained within the destination IP address that the destination router is R3?

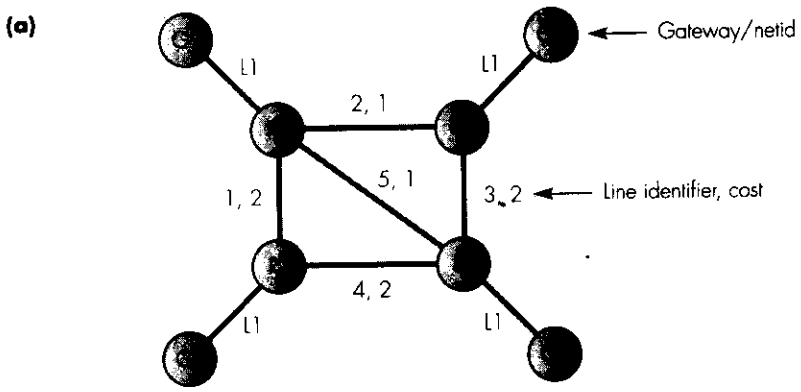
(2) How does R1 know the shortest path route to be followed through the interconnection network to R3?

(3) How does R3 know how to relay the datagram to G3 instead of one of the other gateways that is attached to it?

In relation to the last point, we can accept that a simple protocol can be used to enable each gateway to inform the router to which it is attached, the netid of the access network. The first two points, however, are both parts of the routing algorithm associated with the interconnection network. There are a number of different algorithms that can be used and we shall describe a selection of them in the following subsections. Note that when discussing routing algorithms, the more general term "packet" is used.

9.6.1 Static routing

With this type of routing, the outgoing line to be used to reach all netids is loaded into the routing table of each router when it is first brought into service. As an example, we show the routing table for each of the four routers in the example internet in Figure 9.11(b). To avoid unnecessary repetition, we assume only one gateway/netid is attached to each router.



(b) R1:

Netid	Line
1	L1
2	1
3	1
4	4

R2:

Netid	Line
1	1
2	L1
3	2
4	5

R3:

Netid	Line
1	2
2	2
3	L1
4	3, 2

R4:

Netid	Line
1	4
2	5
3	3, 5
4	L1

Figure 9.11 Static routing: (a) internet topology; (b) routing table entries.

To route a packet from a host attached to, say, netid 1 to another that is attached to netid 3, on receipt of the packet, R1 consults its routing table and determines it should forward the packet on line 1. Similarly, on receipt of the packet, R2 determines it should be forwarded on line 2. Finally, R3 forwards the packet to G3 and from there it is forwarded by G3 to the host specified in the hostid part of the IP address.

As we show in the routing tables for routers R3 and R4, two alternative lines are given for routing packets between these two routers. As we can deduce from the cost values, both routes have the same path cost of 2, one using only line 3 and the other going via R2 and lines 2 and 5. Clearly, however, if a second routing metric of, say, distance was used, then the second path would be longer and hence only a single path would be present. For this reason, more than one metric is sometimes used and the choice of path is then based on the information contained within the related set of routing tables.

We can also make a second observation from this set of routing tables; that is, to go from R1 to R3, the shortest path is via R2 using lines 1 and 2. Also, when we look at the routing table for R2, the shortest path from R2 to R3 is also line 2. More generally, if the shortest path between two routers, A and C, is via an intermediate router B, then the shortest path from B to C is along the same path. This is known as the **optimality principle** and it follows from this that each router along the shortest path need only know the identity of its immediate neighbor along the path. The routing operation is known, therefore, as **next-hop routing** or **hop-by-hop routing**.

The disadvantage of static routing is that all the routing table entries may need to be changed whenever a line is upgraded or a new line is added. Also, should a line or router develop a fault, when the fault is reported, the routing tables in all affected routers need to be changed. For these reasons, static routing is inappropriate for a large, continuously-changing network like the Internet.

9.6.2 Flooding

To explain the operation of the flooding algorithm, we return to Figure 9.10 and again assume we are sending a datagram from host A to host B. The steps followed are summarized in Figure 9.12.

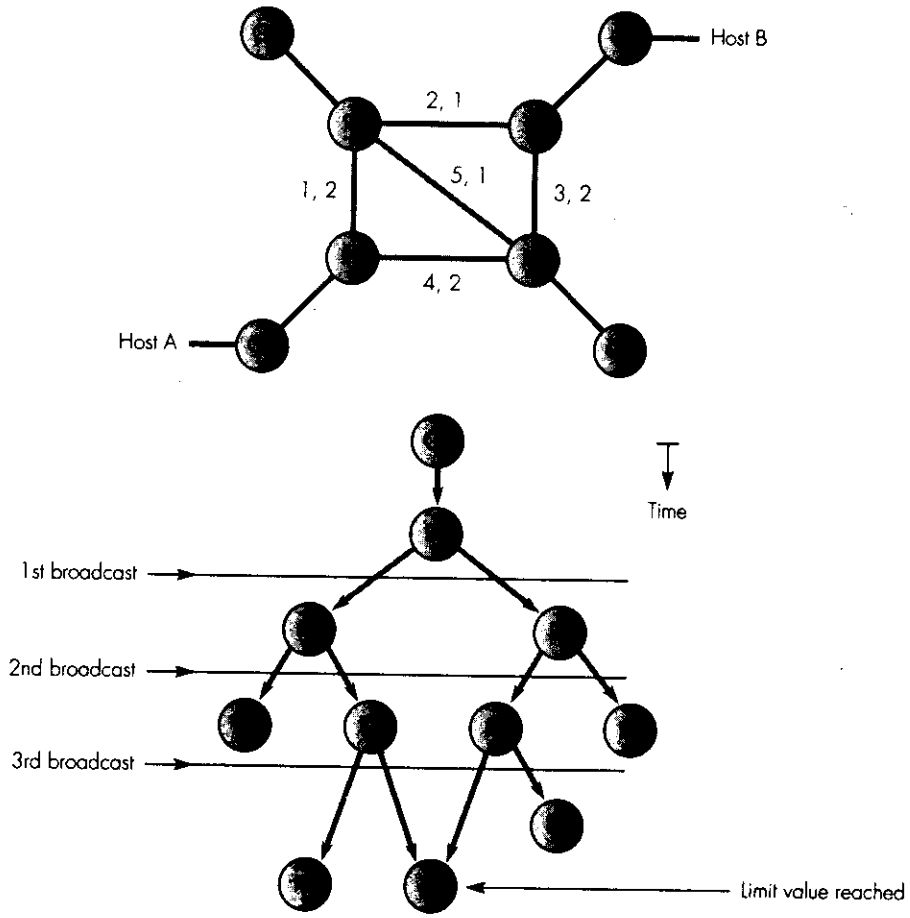


Figure 9.12 Flooding example.

On receipt of the packet from G1, R1 sends a copy of it over both lines 1 and 4. Similarly, on receipt of their copy of the packet, routers R2 and R4 determine from the netid within it that the packet is not for one of their own networks and hence proceed to forward a copy of the packet onto lines 2 and 5 (R2) and lines 5 and 3 (R4). However, since line 2 has a higher bit rate – lower cost value – than line 3, the copy of the packet from R2 will arrive at R3 first and, on determining it is addressed to one of its own netids, R3 forwards the packet to G3. Additional copies of the packet will then be received by R3 but, remembering that each copy will have the same value in the *identifier* field, these can be detected as duplicates and discarded by R3.

In order to limit the number of copies of the packet that are produced, a limit is set on the number of times each copy of the packet is forwarded. In the example, it is assumed that a limit value of 3 has been placed in the *time-to-live* field of the packet header by R1. Prior to forwarding copies of the packet, the limit value is decremented by 1 by the recipient router and only if this is above zero are further copies forwarded.

As we can deduce from the figure, the flooding algorithm ensures that the first copy of the packet flows along the shortest path and hence is received in the shortest time. Also, should a line or router fail, providing an alternative path is available, a copy of each packet should always be received. Flooding, therefore, is an example of an **adaptive** – also known as **dynamic** – routing algorithm. Nevertheless, as we can see from this simple example, even with a limit of three hops, the packet is transmitted 10 times. This compares with just two transmissions using the shortest path. Hence the very heavy bandwidth overheads associated with flooding means that it is inappropriate for use with the large networks that make up the Internet.

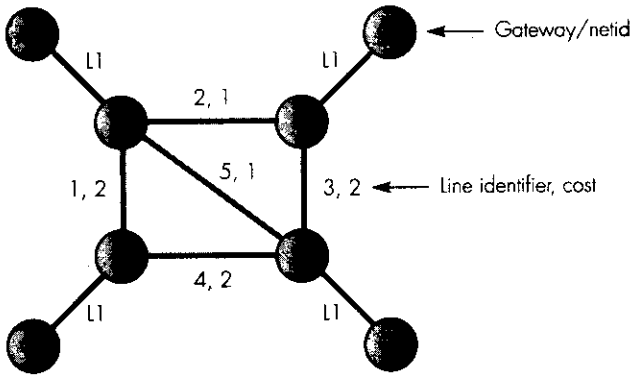
9.6.3 Distance vector routing

The distance vector algorithm is a distributed algorithm that enables each router to build up a routing table (the vector) that contains the path cost (the distance) to reach all the netids in the internetwork.

Initially, each router knows only the identity of, firstly, the netids of the networks that are attached to it – through gateways – and their related local line numbers, and secondly, the identity of the lines – and their cost – that form direct links to other routers. Normally, this information is either entered by network management or by the exchange of configuration messages with the other routers when each router is first brought into service. The information is held in a table known as a *connectivity* or *adjacency* table and the contents of the four tables for our example internetwork, together with the contents of the initial routing table for each router, are shown in Figure 9.13(a).

In order for each router to build up its complete routing table – containing the minimum distance (shortest path) to reach all netids – at predefined time intervals, each router first adds the known cost of the lines that connect

(a)



Connectivity/adjacency tables:

R1:

R	L, C
G1/Netid1	L1, 0
R2	1, 2
R4	4, 2

R2:

R	L, C
R1	1, 2
G2/Netid2	L1, 0
R3	2, 1
R4	5, 1

R3:

R	L, C
R2	2, 1
G3/Netid3	L1, 0
R4	3, 2

R4:

R	L, C
R1	4, 2
R2	5, 1
R3	3, 2
G4/Netid4	L1, 0

Initial routing tables:

R1:

Netid	R, D
1	R1, 0

R2:

Netid	R, D
2	R2, 0

R3:

Netid	R, D
3	R3, 0

R4:

Netid	R, D
4	R4, 0

(b)

R2 R4

R1:

Netid	R, D
1	R1, 0
2	R2, 2
4	R4, 2

R1 R3 R4

R2:

Netid	R, D
1	R1, 2
2	R2, 0
3	R3, 1
4	R4, 1

R2 R4

R3:

Netid	R, D
2	R2, 1
3	R3, 0
4	R4, 2

R1 R2 R3

R4:

Netid	R, D
1	R1, 2
2	R2, 1
3	R3, 2
4	R4, 0

R2 R4

R1:

Netid	R, D
1	R1, 0
2	R2, 2
3	R2, 3
4	R4, 2

R1 R3 R4

R2:

Netid	R, D
1	R1, 2
2	R2, 0
3	R3, 1
4	R4, 1

R2 R4

R3:

Netid	R, D
1	R2, 3
2	R2, 1
3	R3, 0
4	R4, 2

R1 R2 R3

R4:

Netid	R, D
1	R1, 2
2	R2, 1
3	R3, 2
4	R4, 0

Figure 9.13 Distance vector algorithm: (a) internet topology and initial tables; (b) derivation of final routing tables.

the router to its neighbors to the current distance values in its own routing table and forwards a copy of the related updated table to each of its neighbors. Then, based on the information received, if a reported distance is less than a current entry, each router proceeds to update its own routing table with the reported distance. The same procedure then repeats with the updated table contents. This procedure repeats for a defined number of iterations after which, each router has determined the path with the minimum distance to be followed to reach all netids.

As an example, the build-up of the final routing table for each of the four routers in our example internetwork is shown in Figure 9.13(b). To avoid repetition, we assume that only a single gateway/netid is attached to each router and, as we can see, for this simple internet the contents of each routing table are complete after just two routing table updates.

In the case of R1, this receives the updated contents of the routing tables held by R2 and R4. Hence after R1 receives the first set of updated tables from them, it determines that the shortest path to reach netid 2 has a distance of 2 via R2 and, to reach netid 4, the distance is 2 via R4. At the same time, R2 and R4 have themselves received update information from their own neighbors and, as a result, on receipt of the second set of updated tables from them, R1 determines that the shortest path to reach netid 3 has a distance of 3 via R2. Note that with the distance vector algorithm an entry is updated only if a new distance value is less than the current value. Also, that routes with equal path cost values are discarded.

The final routing table of each router contains the next-hop router and the corresponding distance (path cost) value to reach all of the netids in the internetwork. Hence to route a packet, the netid is first obtained from the destination IP address in the packet header and the identity of the next-hop router read from the routing table. The corresponding line number on which the packet is forwarded is then obtained from the connectivity table.

To ensure that each table entry reflects the current active topology of the internet, each entry has an associated timer and, if an entry is not confirmed within a defined time, then it is timed-out. This means that each router transmits the contents of its complete routing table at regular intervals which, typically, is every 30 seconds. Again, for a small internet this is not a problem but for a large internet like the Internet, the bandwidth and processing overheads associated with the distance vector algorithm can become very high. Also, since entries are updated in the order in which they are received and paths of equal distance/cost are discarded, routers may have dissimilar routes to the same destination. As a result, packets addressed to certain destinations may loop rather than going directly to the desired router/gateway. Nevertheless, the **routing information protocol** (**RIP**) which uses the distance vector routing algorithm is still widely used in many of the individual networks that make up the Internet.

9.6.4 Link-state shortest-path-first routing

As the name implies, this type of routing is based on two algorithms: link-state (LS) and shortest-path-first (SPF). The link-state algorithm is used to enable each router to determine the current (active) topology of the internet and the cost associated with each line/link. Then, once the topology is known, each router runs (independently) the shortest-path-first algorithm to determine the shortest path from itself to all the other routers in the internet.

Link-state algorithm

As with the distance vector algorithm, initially, each router knows only its own connectivity/adjacency information and, as an example, the table entries for our example internet are repeated in Figure 9.14(a). The link-state algorithm is then run and the build-up of the internet topology by R1 is shown in Figure 9.14(b).

Initially, based on the information R1 has in its own connectivity table, the (incomplete) topology is as shown in (i). At regular intervals, each router broadcasts a **link-state message**, containing the router's identity and its associated connectivity information, to each of its immediate neighbors. Hence in the example, we assume that R2 is the first to send its own connectivity information to R1 and this enables R1 to expand its knowledge of the topology to that shown in (ii). This is followed by the connectivity information of R4 which enables R1 to expand its knowledge of the topology to that shown in (iii). Concurrently with this happening, the same procedure will have been carried out by all of the other routers. Hence in our example internet, R2 and R4 will have received the connectivity information of R3. After this has been received, therefore, R2 and R4 relay this information on to R1 in a second set of link-state messages and this enables R1 to complete the picture of the active topology (iv). Also, since each router has carried out the same procedure, each will have derived the current active topology and, in addition, determined the identity of the router to which each netid is attached. At this point, each router runs the shortest-path-first algorithm to determine the shortest path from itself to all the other routers. In practice, there are a number of algorithms that can be used to find the shortest path but we shall restrict our discussion to the Dijkstra algorithm.

Dijkstra shortest-path-first algorithm

We shall explain the Dijkstra algorithm in relation to our example internet topology. This is shown in Figure 9.15(a) together with the cost of the lines that link the routers together. The sequence of steps followed by R1 to derive the shortest paths to reach the other three routers is shown in Figure 9.15(b).

Shown in parentheses alongside each of the other routers is the aggregate cost from that router back to the source via the router indicated. Hence an entry of (4,R4) means that the cost of the path back to R1 is 4 via R4. Initially, only the path cost of those routers that are directly connected to R1 are known (R2 and R4) and those not directly connected (R3) are marked

(a)

Connectivity/adjacency tables:

R1:

R	L, C
G1/Netid1	L1, 0
R2	1, 2
R4	4, 2

R2:

R	L, C
R1	1, 2
G2/Netid2	L1, 0
R3	2, 1
R4	5, 1

R3:

R	L, C
R2	2, 1
G3/Netid3	L1, 0
R4	3, 2

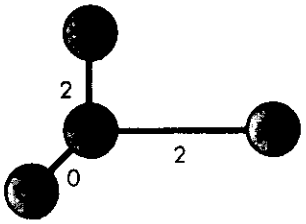
R4:

R	L, C
R1	4, 2
R2	5, 1
R3	3, 2
G4/Netid4	L1, 0

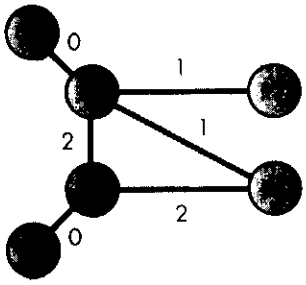
(b)

Topology build-up by R1:

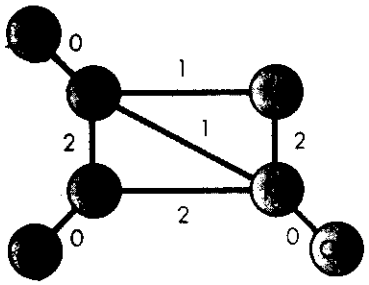
(i) Initial:



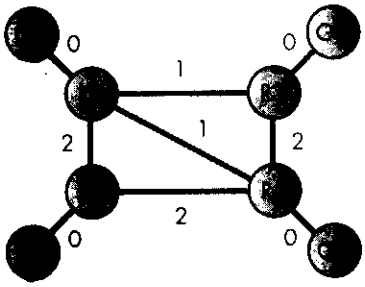
(ii) After connectivity information from R2:



(iii) After connectivity information from R4:



(iv) After connectivity information from R3 via R2:



G/Netid	R
G1/1	R1
G2/2	R2
G3/3	R3
G4/4	R4

Figure 9.14 Link state algorithm: (a) initial connectivity/adjacency tables; (b) derivation of active topology and netid location.

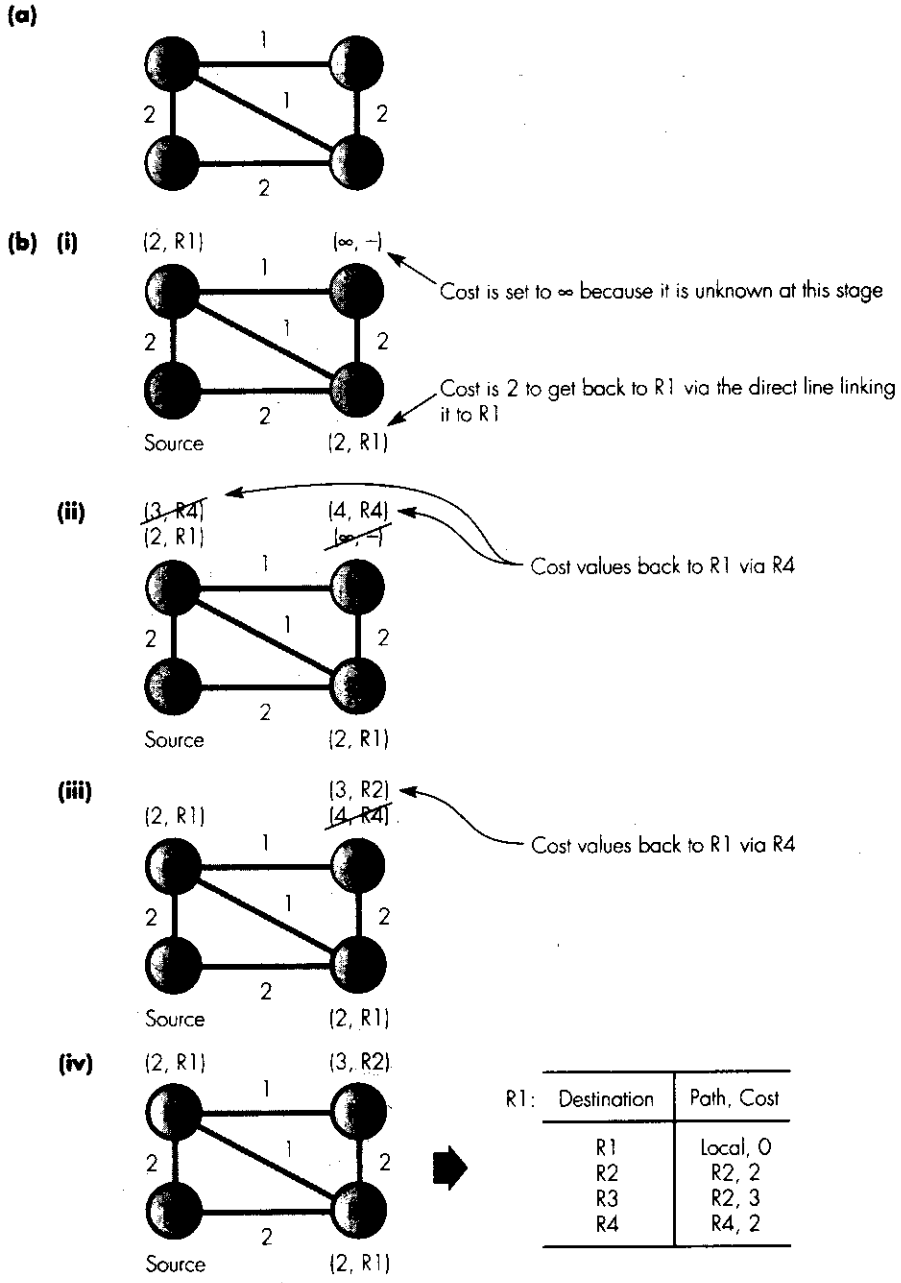


Figure 9.15 Dijkstra algorithm: (a) initial topology; (b) derivation of shortest paths from R1 to each other router.

with an infinite path cost value. Also, until a cost value is known to be the minimum cost, it is said to be **tentative** and only when the cost value is confirmed as the minimum value is it said to be **permanent**. The router is then shown in bold.

Initially, since R1 is the source it is shown in bold and the path costs back to R1 from the two directly connected routers (R2 and R4) are shown equal to the respective line costs (i). Hence R2, for example, has an entry of (2,R1) indicating the cost is 2 to get back to R1 via the direct line linking it to R1. Also, since R3 is not connected directly to R1, it is shown with a path cost of infinity.

Once this has been done, the next step (ii) is to choose the router with the minimum path cost value from all the remaining routers that are still tentative. Hence in our example, the choice is between R2 and R4 – since both are tentative and have a path cost value of 2 – and, arbitrarily, we have chosen R4. This is now marked permanent and the new set of aggregate path cost values via R4 are computed. For example, the cost of the path from R2 to R1 via R4 is 3 (1 from R2 to R4 plus 2 from R4 to R1) but, since this is greater than the current cost of 2, this is ignored. In the case of R3, however, the cost of 4 via R4 is less than the current value of infinity and hence (4,R4) replaces the current entry.

The router with the minimum path cost value is again chosen from those that remain tentative and, since R2 has a path cost of 2, this is marked permanent and the new path costs to R1 via R2 are computed (iii). As we can see, the path cost from R3 to R1 via R2 is only 3 and hence an entry of (3,R2) replaces the current entry of (4,R4). Finally (iv), R3 is made permanent as it is the only remaining router that is still tentative and, now that the minimum path costs from each of the other routers back to R1 are known, the routing table for R1 is complete.

In Figure 9.16 we show the same procedure applied first with R2 as the source – part (a) – then with R3 – part (b) – and finally with R4 – part (c). From these derivations we can make some observations about the algorithm:

- The derived shortest path routes adhere to the optimality principle.
- If the computed path costs associated with two or more tentative routers are the same, then an arbitrary selection can be made as to which is made permanent.
- If the computed aggregate path cost from a (tentative) router to the source via a different router is the same as that via another router, then both can be retained. The choice of route is then arbitrary and load sharing becomes possible.

Datagram routing procedures

The routing of a datagram involves a combination of both link-state tables – one containing the location of all netids and the other the connectivity information – and the derived set of shortest-path routing tables. These are used in slightly different ways depending on the choice of routing method,

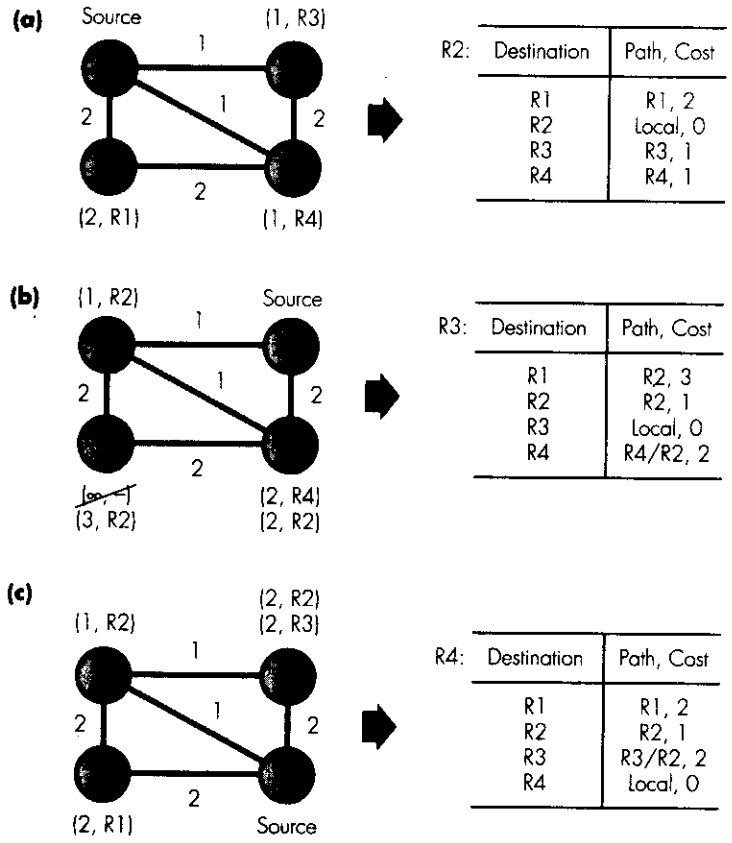
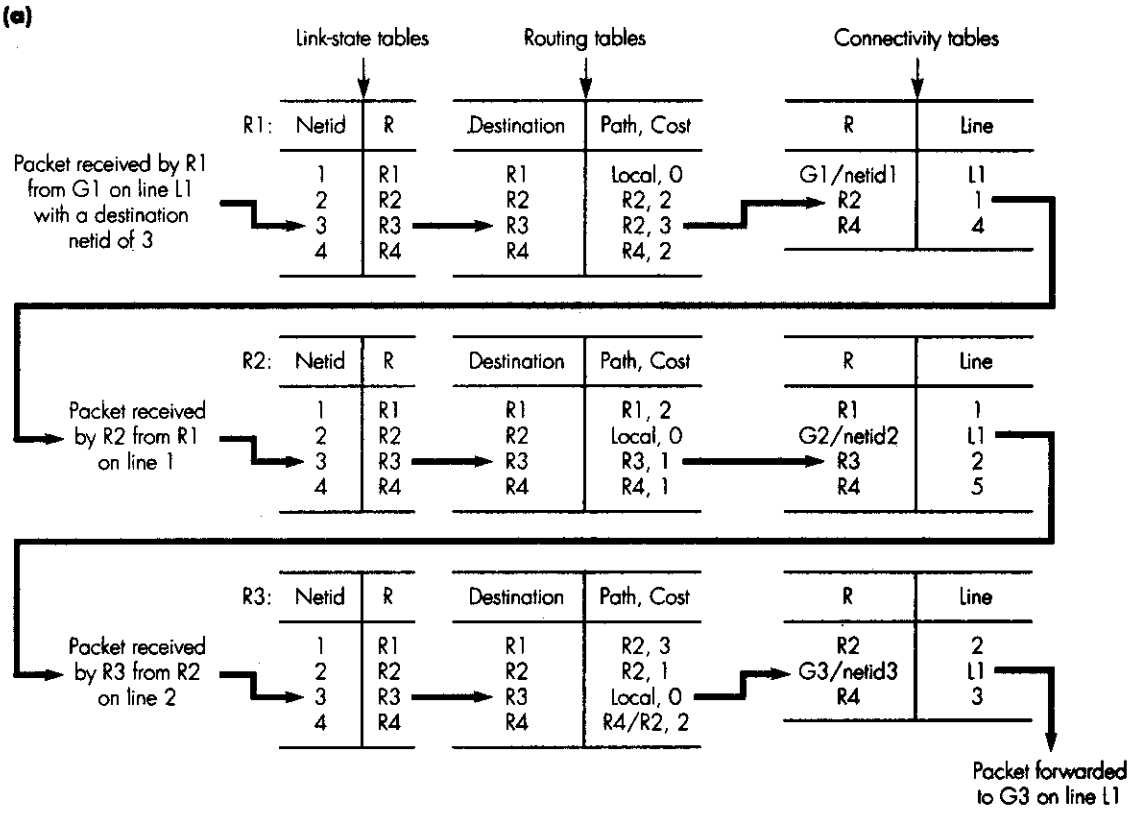


Figure 9.16 Shortest path derivations: (a) by R2; (b) by R3; (c) by R4.

hop-by-hop routing or source routing. We shall explain the procedure followed with each method using the example of a host attached to netid 1 sending a datagram/packet to a host attached to netid 3.

The procedure followed with hop-by-hop routing is summarized in Figure 9.17(a). Using this method, each router computes only its own routing table contents and uses this together with the contents of its own connectivity table. On receipt of the packet from gateway G1, router R1 obtains the netid from the destination IP address in the packet header – netid 3 – and uses its copy of the link-state table to determine that this is reached via router R3. It then determines from the contents of its routing table that the next-hop router on the shortest path to R3 is R2 and hence proceeds to forward the packet to R2 over the line indicated in its connectivity table, line 1.

The same procedure is repeated by router R2 – using its own link-state, routing, and connectivity tables – to forward the packet to R3 over line 2. Finally, on receipt of the packet, R3 determines from its own tables that the packet is addressed to netid 3 and that this is attached to one of its local lines,



(b)

1. Datagram received by R1 from G1 on line L1
2. R1 uses its set of routing tables to determine the least path cost is via routers R2, R3, and writes this list into an options field in the datagram header.
3. R1 uses its own connectivity table to forward the packet to the first router in the list, R2, using line 1.
4. On receipt of the packet, R2 reads the second router from the list, R3, and uses its own connectivity table to forward the packet on line 2.
5. On receipt of the packet, R3 determines it is for one of its local gateways and uses its own connectivity table to forward the packet to G3 on line L1.

Figure 9.17 LS-SPF routing examples: (a) hop-by-hop routing; (b) source routing.

L1. The packet is then forwarded to the attached gateway and from there to the destination host.

The procedure followed with source routing is summarized in Figure 9.17(b). Using this method, once all the routers have built up a picture of the current active topology using the link-state algorithm, they each

compute the complete set of four routing tables. Then, on receipt of a packet from one of its attached gateways – G1 in the example – the source router – R1 – uses the set of tables to determine the list of routers that form the shortest path to the intended destination – R2 and R3. The list is then inserted into an *options* field of the datagram header by R1 and the packet forwarded to the first router in the path, R2, using the corresponding line number obtained from R1's own connectivity table, line 1.

On receipt of the packet, R2 reads from the *options* field the identity of the next router along the path, R3, and uses its own connectivity table to determine the line the packet should be forwarded on, line 2. On receipt of the packet, R3 determines it is intended for one of its local gateways and uses its own connectivity table to determine the packet should be forwarded to gateway G3 on line L1.

Additional comments

Although in the various examples, internet-wide identifiers have been used to identify each of the lines in the example internet topology, this has been done to simplify the related descriptions. In practice, as we can deduce from the description of the LS-SPF algorithm the line identifiers associated with each router have only local significance and, since these are part of the router's configuration information, normally, a different set of line identifiers is used by each router.

In our discussion of the link-state algorithm, we assumed that the transmission of the link-state messages was reliable and that none was lost as a result of transmission errors. Clearly, should a link-state message be corrupted, then the routing tables in each router may be inconsistent and, amongst other things, cause packets to loop. To overcome this, each link-state message, in addition to the identity of the router that created the message and its associated connectivity information, also contains a sequence number and a timeout value. As we have mentioned, link-state information is distributed by each router relaying a copy of the messages it receives from each of its neighbors on to its other neighbors. Hence to avoid messages being relayed unnecessarily, when each new message it created – at defined time intervals – it is assigned a sequence number equal to the previous number plus one. Each router then keeps a record of the sequence number contained within the last message it received from each of the other routers and only if a new message is received – that is, one with a higher sequence number – is a copy forwarded to its other neighbors. In addition, the associated timeout value in each message is decremented by each router and, should this reach zero, the received message is discarded.

Although in the simple example we used to describe the LS-SPF algorithm only a single routing metric (cost value) was used, multiple metrics can be used. In such cases, therefore, there may be a different shortest path between each pair of routers for each metric. Although this leads to

additional computation overheads, the choice of path can then be made dependent on the type of information contained within the datagram being routed. For example, for real-time information such as digitized speech, the choice of path may be based on minimum delay rather than bit rate.

As we can deduce from the description of the distance vector algorithm in Section 9.6.3, for large internets, the amount of routing information passed between routers is substantial and, in the limit, involves each router transferring the contents of its complete routing table at regular intervals. In contrast, the link-state shortest-path-first algorithm involves only the transfer of the link-state information of each router. Hence it is far more efficient in terms of the amount of bandwidth that is utilized for routing updates. It is for this reason, that the LS-SPF algorithm is now the preferred algorithm. The protocol based on this is known as the open shortest path first (OSPF) routing protocol and, as we showed earlier in Figure 9.2, this forms an integral part of the IP.

9.6.5 Hierarchical routing

As we showed earlier in Figure 9.9 and explained in the accompanying text, the Internet is made up of a large number of separately managed networks/internetworks that are interconnected together by means of a global backbone network. Each separate network/internetwork is treated as an **autonomous system** (**AS**) – with its own internal routing algorithm and management authority – and is assigned a number to identify it within the context of the total Internet. The general architecture is shown in Figure 9.18(a) together with some (very much simplified) autonomous system topologies.

To discriminate between the routers used within the various types of network, the following terminology is used:

- **subnet router:** this is a router that operates entirely within a single network when subnetting is being used;
- **access gateway:** this is used to connect an access network to an interior gateway;
- **interior gateway:** this is a router that is used to interconnect the networks within an autonomous system (typically a regional or national network);
- **exterior gateway:** this is a router that is used to connect an autonomous system to the core backbone network.

The various routing protocols associated with the Internet are identified in Figure 9.18(b). The routing protocol used by the interior gateways within a single autonomous system is known as an **interior gateway protocol (IGP)** and that used by the exterior gateways within the backbone network an **exterior gateway protocol (EGP)**. Since the Internet consists of an interconnected

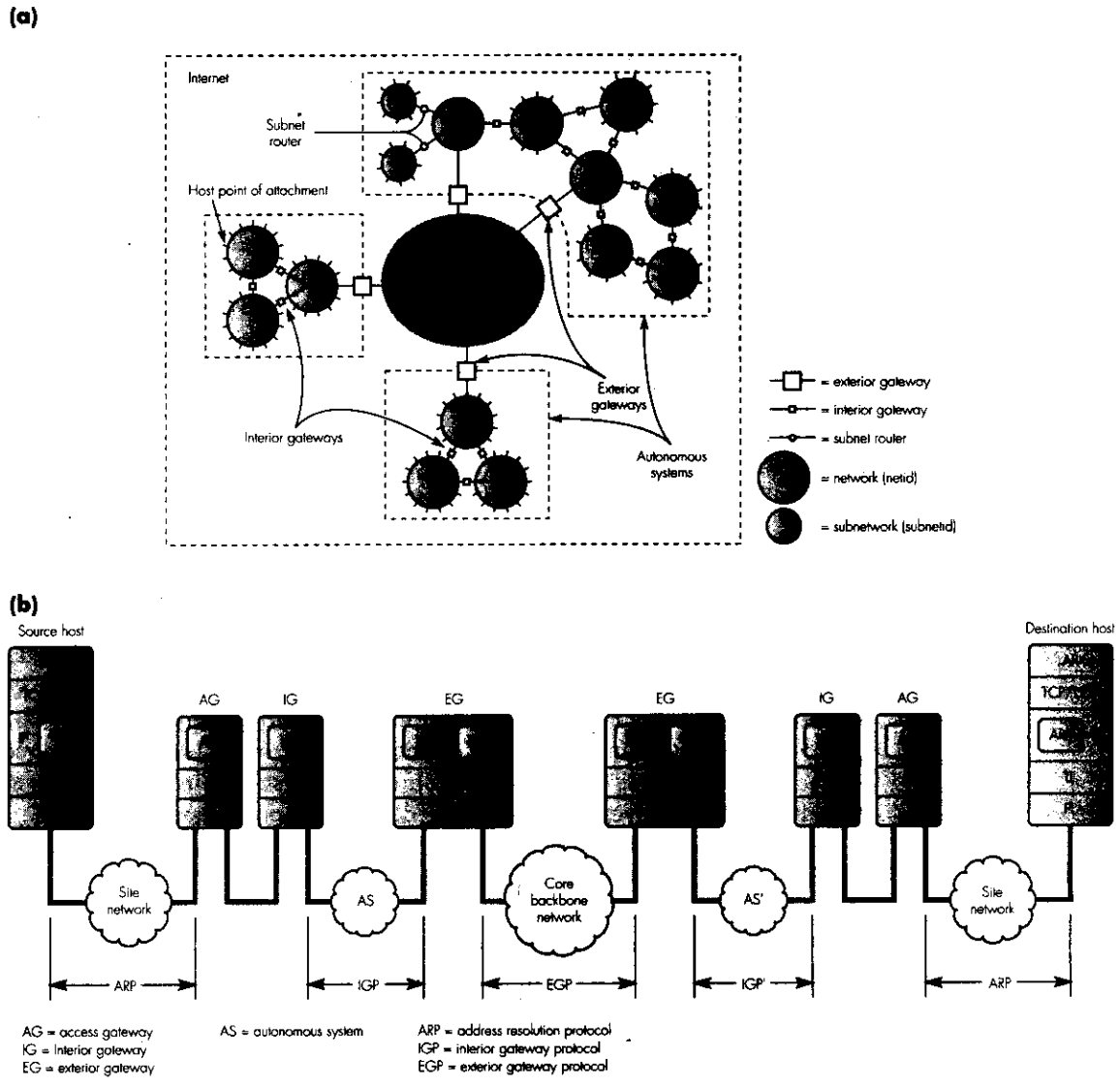


Figure 9.18 Hierarchical routing over the Internet: (a) generalized architecture and terminology; (b) associated routing protocol terminology.

set of networks and internetworks, many of which have evolved over a relatively long period of time, each autonomous system may have a different IGP. Examples are the RIP and the OSPF the principles of which we described in the previous two sections. The EGP, however, as it must be, is an Internet-wide standard.

To reflect the different types of routing protocols involved, the total routing information is organized hierarchically:

- Hosts maintain sufficient routing information to enable them to forward packets directly to other hosts on the same network, a subnet router (if subnetting is being used), and to an access gateway.
- Subnet routers maintain sufficient routing information to enable them to forward packets to other subnet routers (belonging to the same network) or to the network access gateway.
- Interior gateways maintain sufficient routing information to forward packets to their own access gateways and to other interior and exterior gateways within the same autonomous system.
- Exterior gateways maintain sufficient routing information to forward packets to an interior gateway (if the packet is for the autonomous system to which it is attached) or to another exterior gateway if it is not.

An example protocol used by hosts to route packets to other hosts (and an access gateway) that are attached to the same network is the ARP we described earlier in Section 9.5. Subnet routers are used when a network is divided up into subnets to interconnect the various subnets together. Hence they operate in a similar way to interior gateways – and use similar routing protocols – except that all routing within the network is carried out using the subnet address within the hostid part of the destination IP address rather than the netid part. As we indicated earlier, the interior gateway routing protocol can vary from one autonomous system to another. Example protocols are the RIP (based on the distance vector algorithm which we described in Section 9.6.3) and the OSPF (based on the link-state shortest-path-first (LS-SPF) algorithm we described in Section 9.6.4). In this section, we shall describe the standard exterior gateway protocol known as the **border gateway protocol (BGP)**.

Border gateway protocol

The management authority associated with each autonomous system nominates one or more gateways to function as exterior gateways for that system. Within the autonomous system, these communicate with the other interior gateways using the interior gateway protocol for that system. Each exterior gateway, through its local routing table, knows about the netids within that system and their distances from that gateway. The contents of the routing table are built up in the same way as the interior gateways of the autonomous system to which it is attached.

When each exterior gateway is first initialized, it is given the unique identity of the autonomous system to which it is attached and the contents of a connectivity table. This is known as the **reachability table** and enables the gateway to communicate with all the other exterior gateways to which it has a direct link. The BGP in each exterior gateway then makes contact with the

BGP in other selected exterior gateways to exchange routing information with them. This routing information consists of the list of netids within the corresponding autonomous system together with their distances and routes from the reporting exterior gateway. This information is used by a sending gateway to select the best exterior gateway for forwarding datagrams to a particular netid and hence autonomous system.

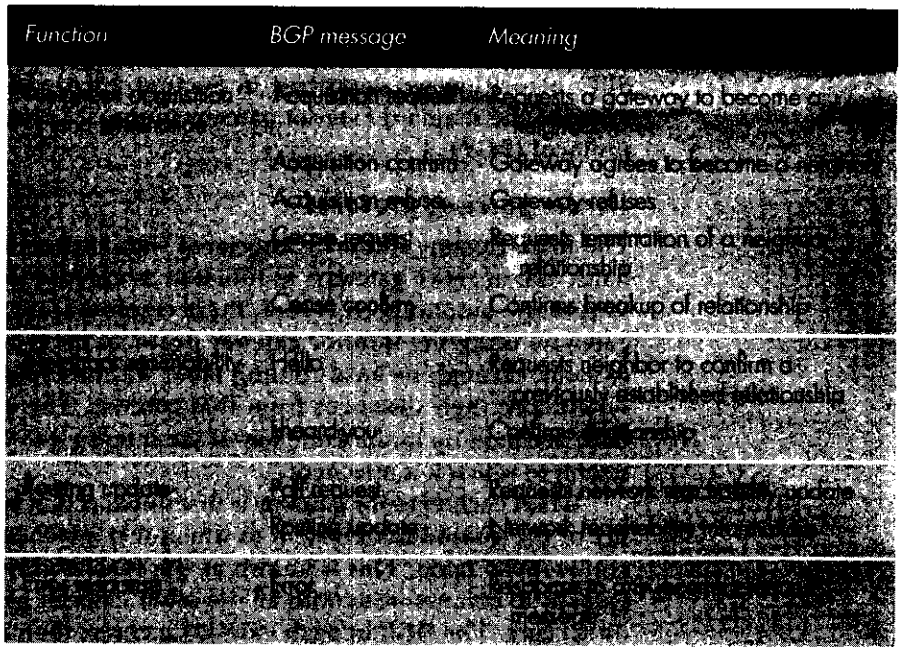
The three main functions associated with the BGP are as follows:

- neighbor acquisition and termination,
- neighbor reachability,
- routing update.

Each function operates using a request-response message exchange. The messages associated with each function are shown in Table 9.1

Since each autonomous system is managed and run by a different authority, before any routing information is exchanged, two exterior gateways attached to different systems must first agree to exchange such information. This is the role of the **neighbor acquisition and termination** procedure. When two gateways agree to such an exchange, they are said to have become **neighbors.** When a gateway first wants to exchange routing information, it sends an

Table 9.1 BGP message types that are exchanged by the BGP in exterior gateways and their meaning



acquisition request message to the BGP in the appropriate gateway which then returns either an *acquisition confirm* message or, if it does not want to accept the request, an *acquisition refuse* message which includes a reason code.

Once a neighbor relationship has been established between two gateways – and hence autonomous systems – they periodically confirm their relationship. This is done either by exchanging specific messages – *hello* and *I-heard-you* – or by embedding confirmation into the header of normal routing information messages.

The actual exchange of routing information is carried out by one of the gateways, which sends a *poll request* message to the other gateway asking it for the list of networks (netids) that are reachable via that gateway and their distances from it. The response is a *routing update* message which contains the requested information. Finally, if any request message is incorrect, an *error message* is returned as a response with an appropriate reason code.

As with the other IP protocols, all the messages (PDUs) associated with the BGP are carried in the user data field of an IP datagram. All BGP messages have the same fixed header, the format of which is shown in Figure 9.19.

The *version* field defines the version number of the BGP. The *type* and *code* fields collectively define the type of message while the *status* field contains message-dependent status information. The *checksum*, which is used as a safeguard against the processing of erroneous messages, is the same as that used with IP. The *autonomous system number* is the assigned number of the autonomous system to which the sending gateway is attached; the *sequence number* is used to synchronize responses to their corresponding request message.

Neighbor reachability messages contain only a header with a *type* field of 5, a code of 0 = hello, and a 1 = I-heard-you.

Neighbor acquisition messages have a *type* field of 3; the code number defines the specific message type. The *hello interval* specifies the frequency with which hello messages should be sent; the *poll interval* performs the same function for poll messages.

A poll message has a *type* field of 2. The *code* field is used to piggyback the neighbor reachability information: a code of 0 = hello and a code of 1 = I-heard-you. The *source network IP address* in both the poll and the routing update response messages indicates the network linking the two exterior gateways. This allows the core network itself to consist of multiple networks with an associated routing protocol.

The *routing update* message contains the list of networks (netids) that are reachable via each gateway within the autonomous system arranged in distance order from the responding exterior gateway. As indicated, this enables the requesting gateway to select the best exterior gateway through which to send a packet for forwarding within an autonomous system. Note that to conserve space, each netid address is sent in three bytes (24 bits) only with the most significant 8-bit hostid field missing. The latter is redundant for all IP address class types.

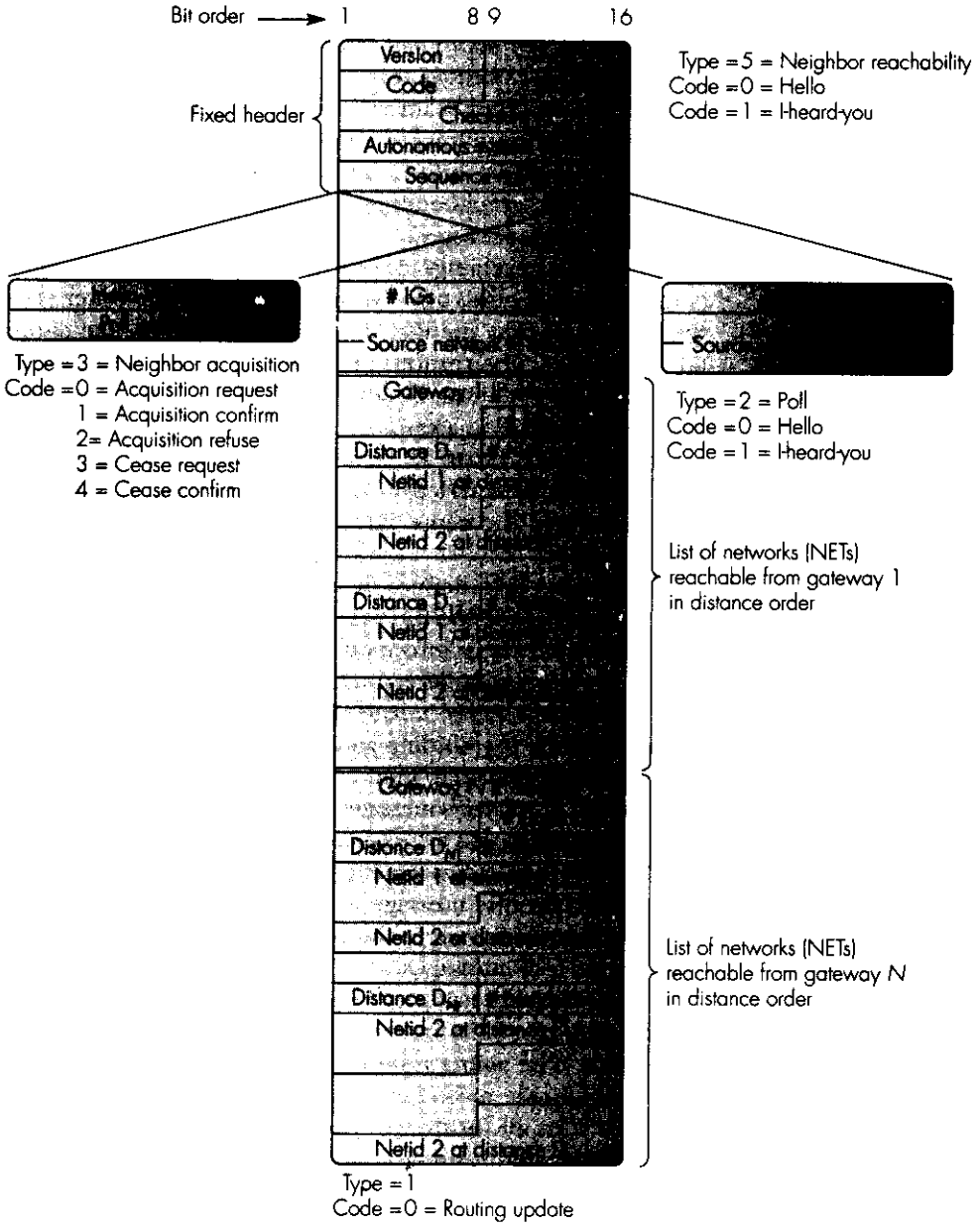


Figure 9.19 Border gateway protocol (BGP) message formats.

9.6.6 Classless inter-domain routing

As we explained earlier in Section 9.4, each IP address is 32 bits in length and is made up of a netid part and a hostid part. The allocation of netids is centrally managed by the Internet Network Information Center and, in order to utilize network addresses efficiently, the number of bits used for the netid part varies. As we showed earlier in Figure 9.5, the number of bits is determined by the address class:

Class A: netid = 7 bits, hostid = 24 bits

Class B: netid = 14 bits, hostid = 16 bits

Class C: netid = 21 bits, hostid = 8 bits

In this way, the manager of a network with many attached hosts can be allocated a class A address, one with a small number of hosts a class C address, and the rest a class B address.

In practice, however, when requesting an IP address, most network managers opted for a class B address since they considered a class C address with just 256 hostids too small and a class A address with 2^{24} hostids too large. As a result, even though there were plenty of netids available with class C addresses, a shortage of class B addresses was predicted. To resolve this problem, an alternative type of routing (which essentially bypasses the fixed divisions associated with each class) known as **classless inter-domain routing** (**CIDR**) was introduced. It is defined in **RFC 1519**.

As we saw in Example 9.1, there are over 2 million class C addresses and hence the primary aim of CIDR was to exploit their usage in a more efficient way. To do this, with CIDR, instead of using the fixed netid/hostid boundary associated with class C addresses, the boundary is made variable and dependent on the number of attached hosts specified by the network manager making a request. For example, if the manager of a new network to be connected to the Internet estimates that the number of hosts may grow to, say, 1000, then a contiguous block of 1024 (2^{10}) class C addresses is allocated.

The alternative CIDR method was introduced relatively recently (post-1996) and, by that time, the general Internet architecture had grown to that we showed earlier in Figure 9.9. Hence in order to reduce the amount of routing information exchanged by exterior gateways, it was decided to abandon the fixed netid/hostid boundary and instead introduce a hierarchical structure that reflected this architecture. Remembering that all class C addresses start with 110, the range of addresses available (in dotted decimal form) is from:

192.0.0.0 through to 223.255.255.255

Clearly, some of these addresses had already been allocated. Hence the allocations to the networks in the different continents are as follows:

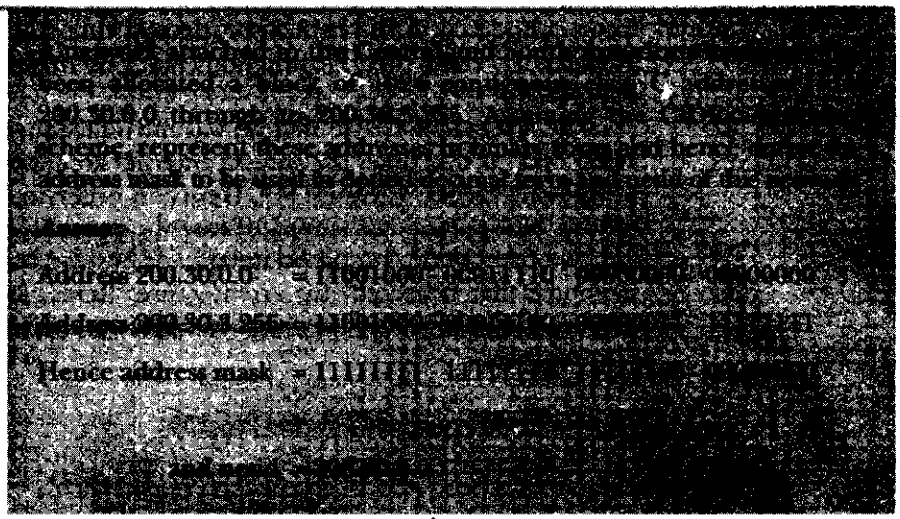
Europe:	194.0.0.0 through to 195.255.255.255
North America:	198.0.0.0 through to 199.255.255.255
Central and South America:	200.0.0.0 through to 201.255.255.255
Asia and the Pacific:	202.0.0.0 through to 203.255.255.255

The remainder of the addresses are held in reserve.

As we can deduce from these allocations, the first byte in the destination IP address indicates the continental backbone to which it should be sent. Hence the routing of packets across the global backbone network with addresses that are in this format is relatively straightforward. For example, any netid that contains 194 or 195 (in dotted decimal) as its first (dotted) decimal number indicates the packet should be sent to the European backbone. However, the absence of a fixed division point in the remaining 24 bits means that each router in the backbone must be informed of the related netid/hostid boundary before it can route the datagram any further.

The approach adopted is similar to that we described earlier in Section 9.4.1 relating to subnetting. As we saw, with subnetting the hostid field is itself divided into a subnetid part and a hostid part with no fixed boundary between them. Instead, the division point is indicated by means of an address mask which contains a binary 1 in all those bit positions that contain the subnetid (and netid) fields. In a similar way, an address mask is used to indicate the boundary between the netid and hostid parts of the new class C addresses. Each exterior gateway then contains a copy of the address mask of each of the networks within the autonomous systems that are attached to it together with the base address – the netid – of the corresponding network.

Example 9.4



In this way, an exterior gateway, on receiving a packet, reads the destination IP address from the packet header and then performs the logical AND operation on this and the list of address masks that it contains. On detecting a match – that is, the resulting netid is the same as that stored with the corresponding mask – the exterior gateway uses the netid and the related interior gateway protocol to route the packet to the appropriate interior gateway. The packet is then passed on to the related access network gateway by this interior gateway.

As we can deduce from this, each interior gateway in the related autonomous system must also contain a copy of the address masks of the networks within that system. Also, each access gateway has a copy of its own address mask and, by using this, it first extracts the hostid from the destination IP address and then uses this to route the packet to that host.

Finally, as we can see from Example 9.4, it is possible for a number of hosts associated with a network which has been allocated a large block of addresses to produce a match with a mask relating to a network with a smaller block of addresses. However, since all network masks are tested, this will be in addition to the match relating to the mask with the smaller block of addresses. Should this happen, then the mask with the smaller block of addresses – and hence larger number of 1s in its address mask – is chosen as the most probable match.

Example 9.5

Two networks that are attached to the Central and [illegible] backbone have been allocated the following block of [illegible]

Network 1: Addresses = 200.64.16.0 through to 200.64.31.255
Mask = 255.192.16.0

Network 2: Addresses = 200.64.17.0 through to 200.64.17.255
Mask = 255.255.255.0

Assuming the CIDR addressing scheme, determine the address of a host attached to network 1 that will produce a match with the mask for network 2.

Answer:

Network 1 Netid = 11001000 01000000 0001/xxxx xxxxxxxx

Network 2 Netid = 11001000 01000000 00010001/ xxxxxxxx

Hence

Network 1 Hostid = 11001000 01000000 0001/0001 xxxxxxxx

9.6.7 Tunneling

In the previous sections we have assumed that all networks within the global internetwork operate in a connectionless mode using the IP and its associated routing protocols. In practice, however, this is not always the case. As we showed earlier in Figure 9.9 and explained in the accompanying text, the Internet is made up of many separately managed networks and internetworks which, in some instances, use a different operational mode and/or protocol from the IP.

For example, consider a small enterprise consisting of two sites, both of which have LANs that operate using the TCP/IP stack, but only one of the sites has an access gateway connected to the Internet. Also, for cost reasons, instead of using a leased line to connect the two site LANs together, a public (or private) data network is used that operates in a connection-oriented mode and with a different protocol from the IP. Clearly, it is not possible to transfer each IP datagram directly over the public data network and instead a technique known as **tunneling** is used. Figure 9.20 illustrates this approach.

As we can see, in order to link the two sites together, a device known as a **multiprotocol router** is connected to each site LAN. As the name implies, a multi-protocol router operates using two different protocol stacks: the IP protocol stack on the site side and the protocol stack associated with the non-IP network on the other. The IP in each host simply treats the multiprotocol

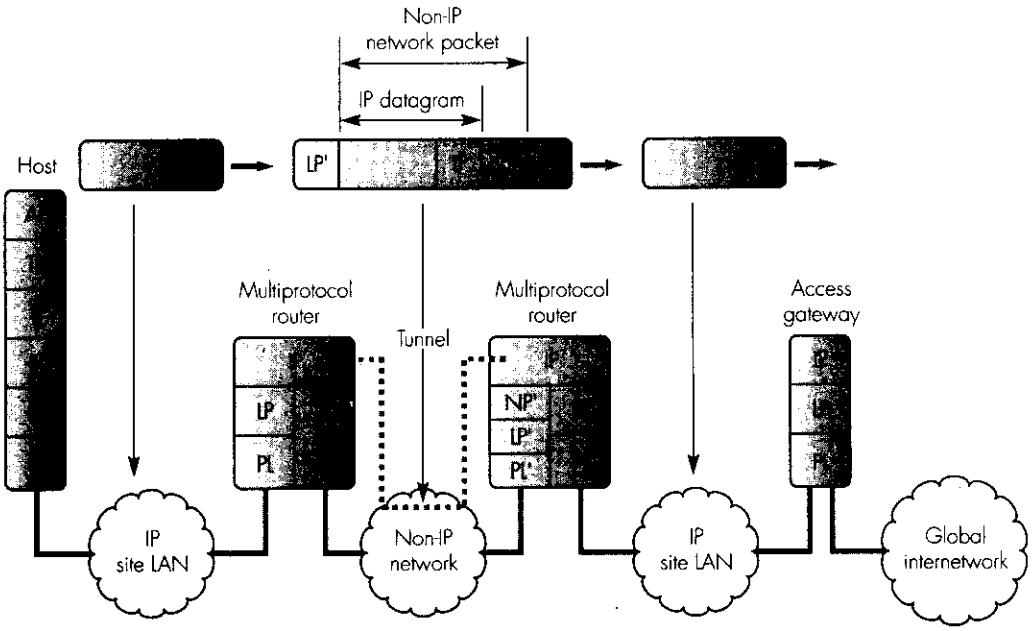


Figure 9.20 Tunneling example.

router as the site Internet access gateway. To send and receive packets to and from a host – a server for example – that is connected to the Internet, the IP simply sends the packet to the multiprotocol router using, for example, the ARP.

Typically, the IP in the source router is given the (non-IP) network address of the multiprotocol router at the remote site by network management. On receipt of the packet, the IP in the source router, on determining that the netid in the destination IP address is not for this site, looks up the non-IP network address of the remote router and passes this, together with the datagram, to the network layer protocol associated with the non-IP network. The latter treats the datagram as user data and proceeds to transfer the datagram to the peer network layer in the remote router using the protocol stack of the non-IP network with the datagram encapsulated in a data packet relating to the network layer protocol.

On receipt of the data packet by the peer network layer protocol in the remote router, the user data – the datagram – contained within it is passed to the IP. The IP first determines from the destination IP address that the required host is not attached to the site LAN and hence proceeds to send the packet to the IP in the Internet access gateway using, for example, the ARP. A similar procedure is followed in the reverse direction to transfer the packets containing the related reply message. Thus, the presence of the non-IP network is transparent to the IP in each host and the access gateway.

In addition to using tunneling to transfer an IP packet over a non-IP network, the same technique is used to send an IP packet over an IP network. As we shall expand upon in Section 9.6.11, tunneling is used by an IP router to relay a packet that contains a multicast destination address to a different router that can handle multicast packets. Normally, the IP address of their nearest multicast router is known by all the other routers and, on receipt of a packet with a multicast address, the source router encapsulates the packet within a new packet with the IP address of the multicast router as the destination IP address.

9.6.8 Broadcast routing

As we explained in the last chapter, LANs such as Ethernet and token ring operate by a station/host broadcasting each frame over the LAN segment to which it is attached. The frame is then received by all the other stations that are attached to the same segment and, by examining the destination (MAC) address in the frame header, the network interface software within each host can decide whether to pass the frame contents on to the IP layer for further processing or to discard the frame. A frame is accepted if the destination MAC address is the same as its own individual address, or is a broadcast address, or is equal to one of the group addresses of which the station is a member. For a bridged LAN, this mode of working is extended to cover the total LAN by each bridge relaying all frames that contain either a broadcast or a multicast address on to all the other LAN segments to which the bridge is

attached. In this section we explain how broadcasting is achieved at the IP layer and, in the next section, how multicasting is achieved.

As we identified in Section 9.4, there are a number of different types of IP broadcast address:

- **limited broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the same LAN segment or bridged LAN. To achieve this, the destination IP address is set to 255.255.255.255.255. Neither subnet routers nor access gateways forward such packets;
- **subnet-directed broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the subnet specified in the destination IP address. To achieve this, the subnet mask associated with the subnet must be known and this is then used to determine the hostid part and set all these bits to 1. Such packets are forwarded by subnet routers but only if the destination netid is different from the source netid are they forwarded by access gateways, interior gateways, and, if necessary, exterior gateways;
- **net-directed broadcast**: this is used to send a copy of a packet to the IP in all the hosts that are attached to the network specified in the netid part of the destination IP address. Such packets are forwarded by subnet routers but only if the destination netid is different from the source netid are they forwarded by access gateways, interior gateways, and, if necessary, exterior gateways.

Thus, a packet with a net-directed broadcast address whose destination netid is different from the source netid may need to be forwarded across the global internetwork. Since the destination netid is known, however, then the datagram can be routed by interior – and if necessary exterior – gateways in the same way as a packet with a unicast address. This also applies to a packet with a subnet-directed broadcast address whose destination netid is different from the source netid. Also, since with a subnet-directed broadcast address all the subnet routers in both the source and destination networks have a copy of the subnet mask, then they too can use the unicast routing algorithm associated with the network to route the packet to the subnet router specified in the IP address. The latter then broadcasts the packet over this subnet. With a net-directed broadcast, however, this is not possible and the unanswered question is how the packet is broadcast to all the subnets belonging to the network specified in the netid part of the address.

One solution is to use flooding but, as we concluded at the end of Section 9.6.2, this has very high bandwidth overheads associated with it. Two alternative approaches are employed, the choice determined by the routing algorithm that is used to route unicast packets over the network. For description purposes we shall use the example of a large site/campus network that comprises a large number of subnets all interconnected by subnet routers.

The aim of both algorithms is then for the arriving packet with a net-directed broadcast address to be broadcast over all the subnets using a minimum amount of bandwidth.

Reverse path forwarding

This algorithm is used primarily with networks that use the distance vector (DV) algorithm to route unicast packets. To explain the operation of the algorithm we use the network topology shown in Figure 9.21(a). We assume that subnet router 1 (SR1) also acts as the (single) access gateway for the network and that all subnets (SNs) are broadcast LANs.

Using the DV algorithm we explained in Section 9.6.3, in addition to each subnet router deriving the shortest paths to each subnet, they can also derive the shortest paths to reach each of the other subnet routers. To see how this is done, the initial and final routing tables built up by each subnet router (based on a routing metric of hop count) are shown in Figure 9.21(b).

Once the routing tables have been created, the reverse path forwarding algorithm used to route (broadcast) packets works as follows. On receipt of a packet/datagram, the IP in each subnet router (SR) consults its routing table and only forwards a copy of it – onto each of the ports of the SR except the port the packet arrived on – if the packet arrived from an SR that is on the shortest path from SR1 to the SR that is processing the packet. If it is not, then the packet is discarded. Based on this simple rule, the path followed by each copy of an incoming packet is shown in Figure 9.21(c).

As we can see, on receipt of a packet, SR1 broadcasts a copy of it out onto subnets SN1 and SN3. Hence a copy of the packet is received by the IP in SR2 and SR4 respectively. On receiving the packet, the IP in SR2 first consults its routing table and determines from the (first) entry in the table that SN1 (from which the packet was received) is on the shortest path back to SR1. Similarly, when the IP in SR4 consults its routing table it also determines that SN3 (from which the packet was received) is also on the shortest path back to SR1. Hence both SR2 and SR4 are shown in bold in the figure and each proceeds to broadcast a copy of the packet, SR2 onto SN2 and SN4, and SR4 onto SN7.

After the second set of broadcasts, on receipt of its copy of the packet, the IP in SR3 determines from its routing table that SN2 is on the shortest path back to SR1. Similarly for the copy SR5 receives from SR2 via SN4. Hence both SR3 and SR5 are shown in bold and proceed to broadcast a copy of the packet, SR3 onto SN5 and SN6, and SR5 onto SN5, SN7, and SN8. However, in the case of the packet received by SR5 from SR4 via SN7, SR5 determines that SN7 is not on the shortest path back to SR1. Hence SR5 along this path is not shown in bold and the arriving packet is discarded.

The same procedure is repeated by SR5 and SR6 after the third set of broadcasts have been received but this time only SR6 determines from its routing table that SN6 is on the shortest path back to SR1 and proceeds to broadcast a copy of the packet onto SN8. The copies of the packet received

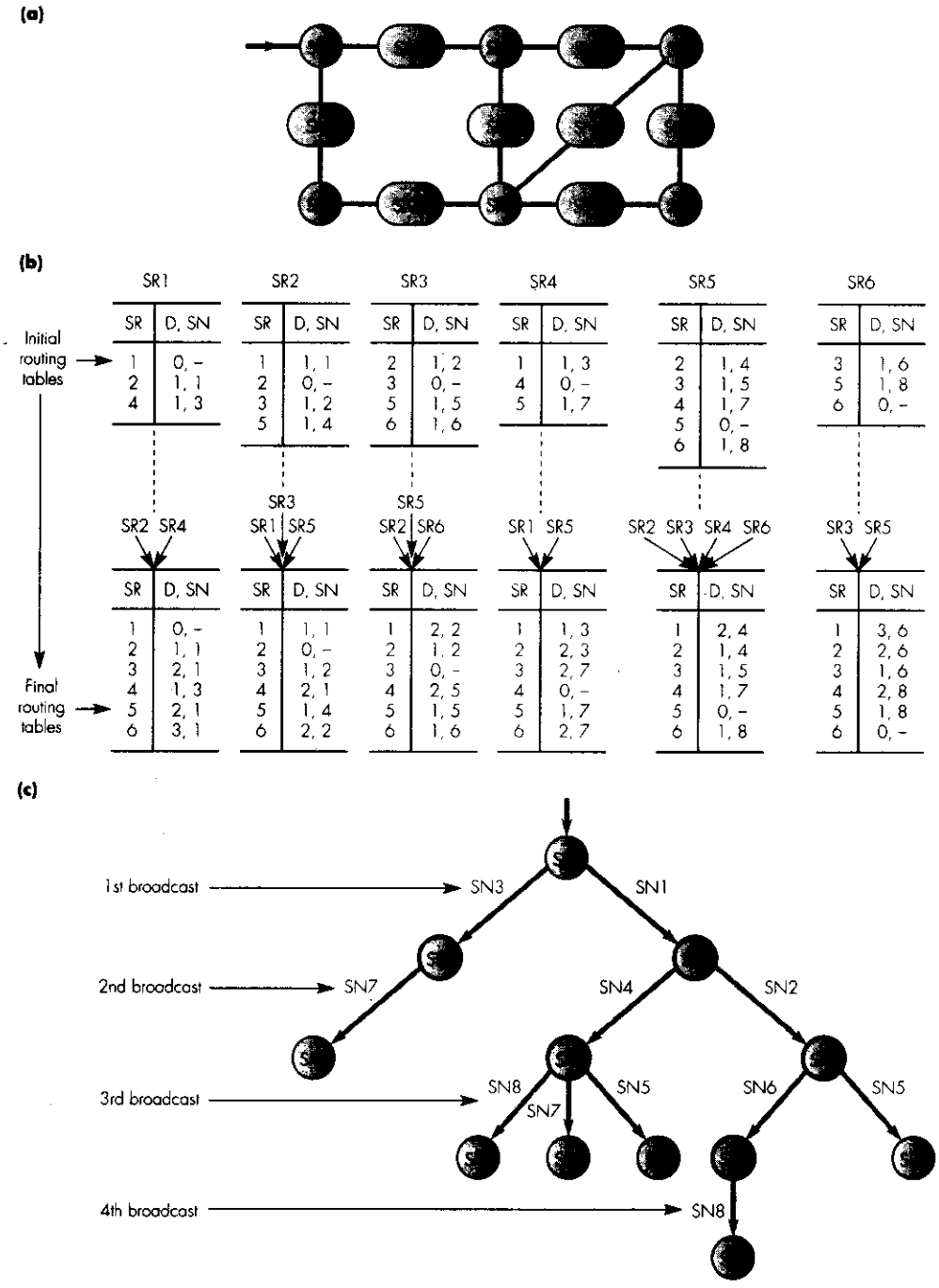


Figure 9.21 Reverse path forwarding: (a) network topology; (b) distance vector routing tables using a hop-count metric; (c) broadcast sequence.

from SR5 by SR3, SR4, and SR6 are all discarded as the related subnets – SN5, SN7, and SN8 – are not on the shortest paths back to SR1. Also the packet received by SR5 from SR6 after the fourth broadcast.

As we can deduce from this example, a copy of the packet is broadcast over all the eight subnets that make up the network and only SN7 and SN8 receive two copies. However, since both copies of the packet have the same value in the identifier field of the packet header, the second copy can be detected by each of the hosts on these subnets as a duplicate and is discarded. Note also that the same set of routing tables can be used to perform the same algorithm if a second (or different) SR acts as an additional (or alternative) access gateway. Also, if the broadcast is over the source network.

Spanning tree broadcast

With networks that use a routing algorithm based on the link-state algorithm, an alternative way of routing broadcast packets/datagrams is for each router to use the link-state information to establish a spanning tree active topology with the access gateway/router as the root node.

As we saw in Section 9.6.4, the information gathered as part of the link-state algorithm is used by each router to derive the current active topology of the internetwork. In a similar way, therefore, with networks that consist of multiple subnets interconnected by subnet routers, each subnet router builds up knowledge of the current active topology of the network and then uses this to compute the shortest path to reach all the subnetids in the network.

Hence with the spanning tree broadcast algorithm, in addition to each subnet router computing the shortest paths, they all derive the (same) spanning tree topology from the current active topology using, for example, the algorithm we described in Section 8.5.1. As we saw, a spanning tree topology is established in order to avoid frames looping within a bridged LAN. This is done by defining the ports associated with each bridge as either root ports or designated ports. All ports that are either root or designated ports are then set into the forwarding state and all the other ports are set into the non-forwarding (blocked) state.

Using the same approach, we can derive a spanning tree by setting selected subnet router ports into the forwarding and blocked state. For example, using the network topology we showed in Figure 9.21(a) and the algorithm we described in Section 8.5.1, assuming each subnet router knows the root SR, each will derive the spanning tree shown in Figure 9.22(b). The resulting broadcast sequence is therefore as shown in Figure 9.22(c).

We can make a number of observations from this example:

- For consistency, the port numbers associated with each subnet router are determined by the (known) identifier of the attached subnet.
- Each SR has a root port (RP) associated with it which is the port with the shortest path back to the root. The path costs in the example are based on hop count and, in the event of a tie, the port with the smallest port number is chosen.

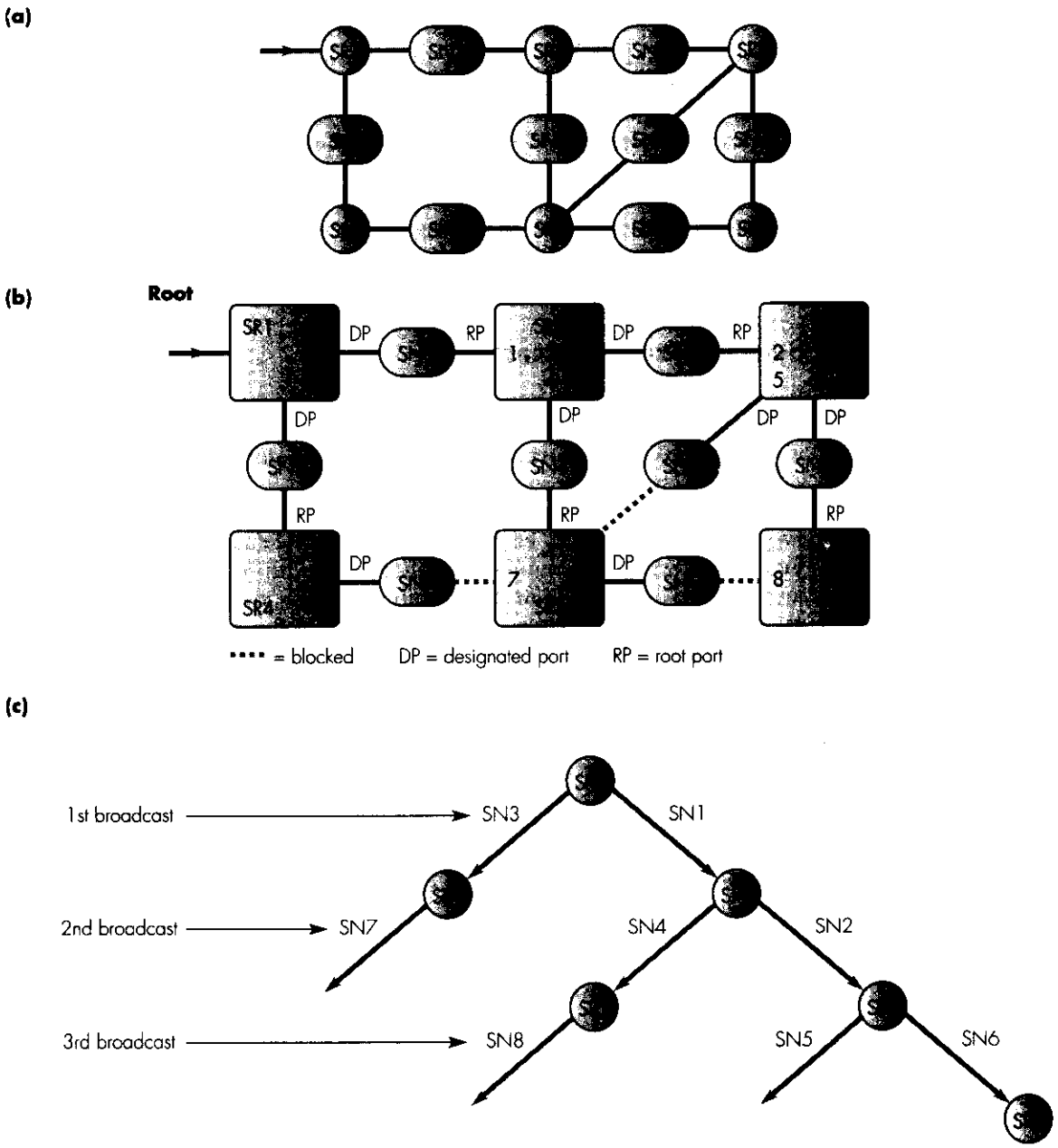


Figure 9.22 Spanning tree broadcast: (a) network topology; (b) spanning tree derived by each subnet router; (c) broadcast sequence.

- For each subnet, there is a designated port (DP) which is the router port on the shortest path from the root to the subnet. In the event of a tie, the SR with the smallest identifier is chosen.
- All router ports that are not root or designated ports (DP) are set into the blocked state.
- Only a single copy of each received packet/datagram is broadcast over each subnet.
- The same spanning tree can be used to broadcast packets if a second (or different) SR acts as an additional (or alternative) access gateway and if the broadcast is over the source network.

9.6.9 Multicast routing

As we explained in Chapter 1, applications such as audio- and videoconferencing require a copy of the information generated by each host participating in a conference to be sent to all the other hosts that belong to the same conference. The term **multicasting** is used to describe the diffusion of a copy of the packets/datagrams generated by each host to all the other hosts and the term **multicast group** is used to identify the hosts that are members of the same conference. Clearly, there can be many conferences in progress concurrently, each involving a different group of hosts. It is to support applications of this type that class D multicast addresses were defined. As we showed in Figure 9.4 and explained in the accompanying text, a single class D address is used to identify all the hosts that are members of the same multicast group. Also, since 28 bits are available, many different multicast groups can be in place concurrently.

A number of multicast addresses are reserved to identify specific groups. For example, all the hosts (and subnet routers) that are attached to a site broadcast network are members of the group with the reserved multicast address of 224.0.0.1. Hence the IP in all hosts that can support multicasting will have this address permanently in their **multicast address table (MAT)** and, should a packet be received with this address, the datagram contents will be passed to (and acted upon) by a related application process. Similarly, all the subnet routers belonging to the same network are members of the multicast address 224.0.0.2.

As we showed in Figure 5.4, with networks that do not support broadcasting such as PSTNs and ISDNs, the diffusion operation is achieved by using a central (conference) server known as a multipoint control unit (MCU). Also, in packet-switching networks that support broadcasting such as LANs, as we showed in Figure 5.6, the H.323 standard uses a similar approach with a networked device known as a gatekeeper performing the multipoint control function of an MCU. Using this approach, the IP and MAC address-pair of the MCU is known by the IP in all the hosts that are participating in a conference and, once the IP in the MCU has obtained the IP/MAC address-pair of all the participating hosts, the IP in the MCU can carry out the diffusion operation using only the individual (unicast) IP address of each participant.

This approach is acceptable providing the number of participating hosts is relatively small and the composition of the conference is static. However, for applications such as a conference or meeting which is being transmitted over a LAN or the Internet, a relatively large number of participants – and hence hosts – may be involved. Moreover, hosts may wish to join and/or leave the conference/meeting whilst it is in progress. It is to meet this type of application that multicasting using IP multicast addresses is used.

With this mode of working, the organizer of the conference/meeting first obtains an IP multicast address for it from the Internet assigned numbers authority (IANA). The allocated address is then made known to all the registered participants together with the conference/meeting start time and its likely duration. Each participating host can then request to join the conference at any time during the time the conference/meeting is in progress. To do this, two operating scenarios are used which depend on whether the participating hosts are all attached to the same LAN/subnet or, as is more usual, are attached to many different networks that are geographically distributed around the Internet. We shall discuss each separately.

Multicasting over a LAN

The IANA has been allocated a block of Ethernet MAC addresses for applications that involve multicasting. As we described in Section 8.3, Ethernet MAC addresses are 48 bits in length and the reserved block of addresses in dotted decimal are from

0.0.94.0.0.0 through to 0.0.94.255.255.255

Half of these addresses are used for multicasting. Hence, remembering that all centrally administered Ethernet group (multicast) addresses must start with 10 binary, the block of Ethernet addresses used for centrally administered multicasting applications are from

128.0.94.0.0.0 through to 128.0.94.127.255.255

Thus for a particular conference/meeting, a 3-byte address in the range

0.0.0 through to 127.255.255

is allocated by the IANA. In the case of an Ethernet LAN, this forms the least significant 24 bits of the 48-bit destination MAC address – starting with the three bytes 128.0.94 – and, in the case of the Internet, the least significant 24 bits of the destination IP multicast address. As we explained earlier, class D IP addresses are 28 bits in length – the first four bits being 1110 – and hence, as we show in Figure 9.23(a), the four remaining bits are all set to 0.

So to join a conference/meeting that is being broadcast over the same LAN, once the multicast address is known, the application process running in

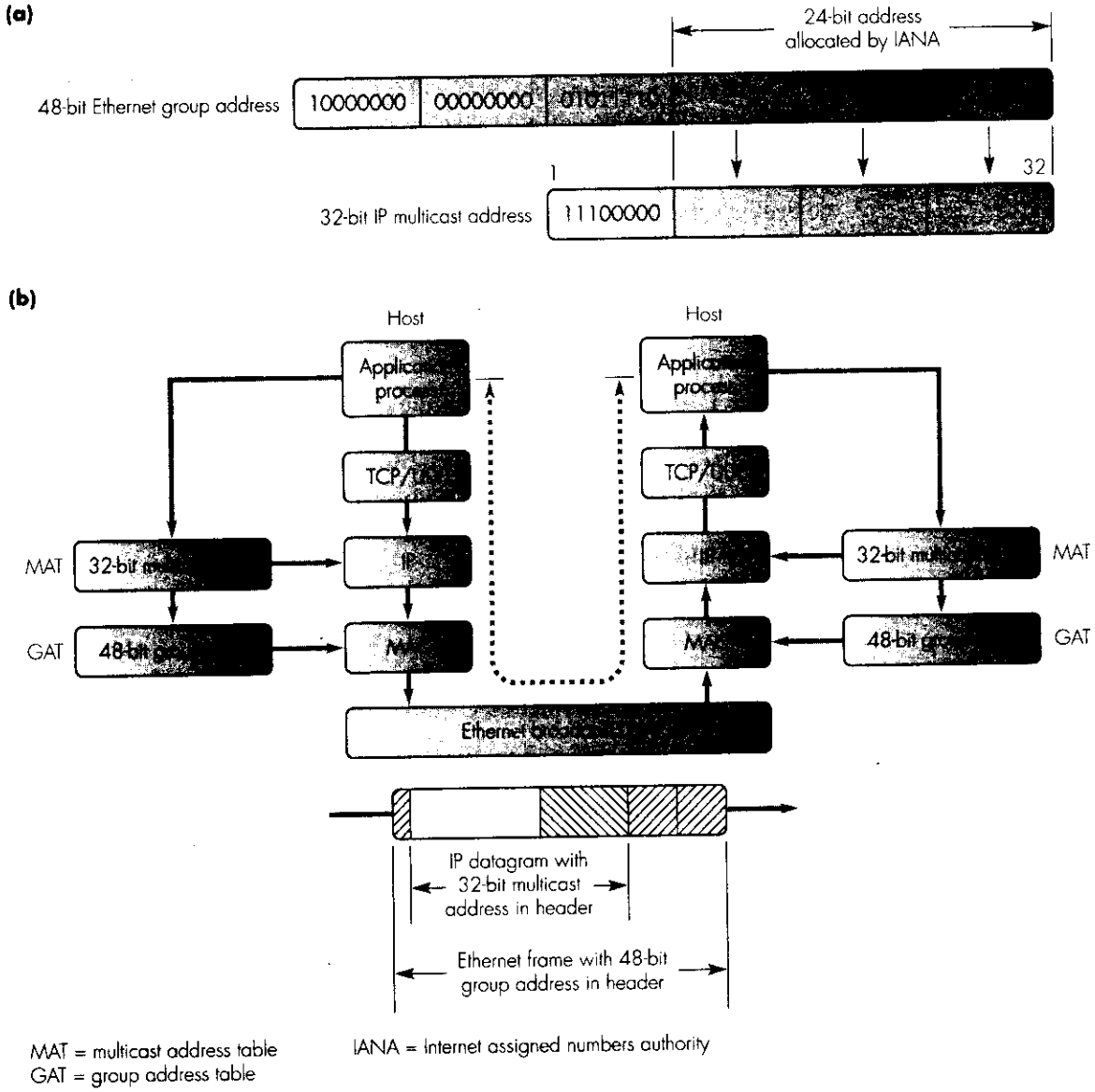


Figure 9.23 Multicasting over a LAN: (a) address allocation principle; (b) address usage.

the host that is managing the information associated with the conference/meeting simply loads the allocated IP multicast address into its MAT and the related 48-bit Ethernet group address derived from this into its **group address table (GAT)**. Each datagram relating to the conference/meeting then has the allocated multicast address in the destination IP address

field of the datagram header. Each packet is then broadcast over the LAN in an Ethernet frame containing the derived 48-bit group address in the destination MAC address field of the frame header.

On receipt of a frame, the MAC sublayer in each host that is a member of the same multicast group, first checks to see whether the destination MAC address in the frame header is present in its group address table and, if it is, it passes the frame contents – the datagram – on to the IP layer. The latter first determines that the destination IP address is a multicast address and also that it is present in its multicast address table. It therefore passes the information contained within the datagram on to the related application process via the TCP or UDP. This procedure is shown in Figure 9.23(b).

Multicasting over the Internet

When the hosts that are part of a multicast group are attached to different networks/subnetworks geographically distributed around the Internet, then intermediate subnet routers and/or interior/exterior gateways may be involved. Thus, since an IP multicast address has no structure – and hence no netid – associated with it, a different type of routing from that used to route unicast packets must be used.

The sequence of steps followed to route a packet with a multicast address is as follows:

- A router that can route packets containing a (destination) IP multicast address is known as a **multicast router** (**mrouter**).
- Normally, in the case of a network that comprises multiple subnets interconnected by subnet routers, a single subnet router also acts as the mrouter for that network.
- Each mrouter learns the set of multicast group addresses of which all the hosts attached to the networks which the mrouter serves are currently members.
- The information gathered by each mrouter is passed on to each of the other mrouters so that each knows the complete list of group addresses that each mrouter has an interest in.
- On receipt of a packet with a destination IP multicast address, each mrouter uses an appropriate routing algorithm to pass the packet only to those mrouters that are attached to a network which has an attached host that is a member of the multicast group indicated in the destination IP address field.

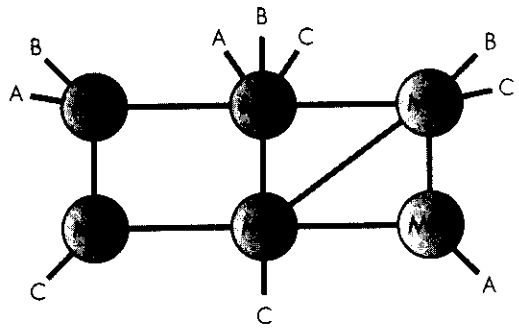
As with broadcast routing, two different algorithms are used, the choice determined by the routing algorithm that is used to route unicast packets. The aim of both algorithms is to minimize the amount of transmission bandwidth required to deliver each multicast packet to those multicast routers that

have an interest in the packet. To explain the two algorithms, we shall use the internetwork topology shown in Figure 9.24(a). The letters by each multicast router (MR) – A, B, and C – indicate the multicast address(es) that each has an interest in and, since all MRs exchange this information, they each have a copy of the multicast address table shown in Figure 9.24(a).

DVMRP

When distance vector routing is being used, an additional set of routing tables (to those used to route unicast packets) based on MR-to-MR distances are derived and, for the six MRs, these are given in Figure 9.24(b). They are based on a routing metric of hop count and have been derived using the same procedure we described earlier in Section 9.6.3. Together with the contents of the MAT in each MR, they are used to route all packets that have a destination multicast address.

(a)



Multicast address table

MR	Multicast address
1	A, B
2	A, B, C
3	B, C
4	C
5	C
6	A

(b)

MR1

MR	MR, D
1	1, 0
2	2, 1
3	2, 2
4	4, 1
5	2, 2
6	2, 3

MR2

MR	MR, D
1	1, 1
2	2, 0
3	3, 1
4	1, 2
5	5, 1
6	3, 2

MR3

MR	MR, D
1	2, 2
2	2, 1
3	3, 0
4	5, 2
5	5, 1
6	6, 1

MR4

MR	MR, D
1	1, 1
2	5, 2
3	5, 2
4	4, 0
5	5, 1
6	5, 2

MR5

MR	MR, D
1	2, 2
2	2, 1
3	3, 1
4	4, 1
5	5, 0
6	6, 1

MR6

MR	MR, D
1	3, 3
2	3, 2
3	3, 1
4	5, 2
5	5, 1
6	6, 0

MR = multicast router D = distance in hops

Figure 9.24 Distance vector multicast routing: (a) example topology and multicast address table; (b) unicast routing tables of MR1–6.

The protocol is known as the **distance vector multicast routing protocol** (**DVMRP**). To explain how it works, assume a packet with a multicast address of A is received by the IP in MR1 from one of its attached networks. The sequence is as follows:

- The IP first consults its MAT and finds that a copy of the packet should be sent to MR2 and MR6. It then proceeds to consult its routing table (RT) and finds that the shortest path to MR6 is also via MR2 and hence sends just a single copy of the packet to MR2.
- On receipt of the packet, MR2 consults its MAT and sees that a copy of the packet should be sent out onto its own network and also to MR1 and MR6. On consulting its RT, however, it finds that the shortest path to MR1 is on the line/port the packet was received and hence it only sends a copy of the packet to MR6. The RT indicates that the shortest path to MR6 is via MR3 and hence it forwards a copy of the packet to MR3.
- On receipt of the packet, MR3 determines from its MAT that MR1, MR2, and MR6 should be sent a copy of the packet. On consulting its RT it finds that the shortest path to both MR1 and MR2 is via MR2. Hence, since this is the line/port the packet arrived on, it only sends a copy of the packet to MR6.
- On receipt of the packet, MR6 determines from its MAT that a copy of the packet should be sent out on its own network and also to MR1 and MR2. However, since the shortest path to both MR1 and MR2 is via the line/port the packet arrived on, it only sends a copy out on its local network.

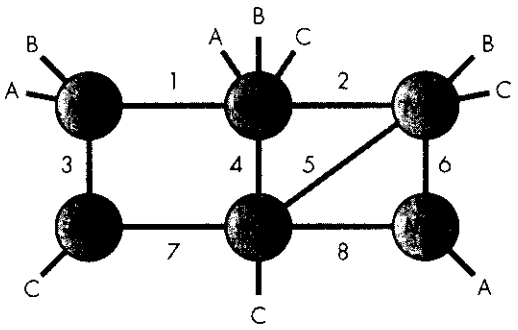
MOSPF

When link-state routing is being used, since each MR knows the current active topology of the internetwork, as we explained in Section 9.6.8, each computes a spanning tree for the internetwork using the algorithm we described in Section 8.5.1. For example, for the internetwork shown in Figure 9.25(a), the spanning tree computed by each MR is shown in Figure 9.25(b). The numbers shown by each line are used as global line identifiers and as port numbers in the derivation of the spanning tree.

At any point in time, each MR knows from its MAT the multicast addresses that each has an interest in. For each address – A, B, and C in the example – each MR proceeds to produce a pruned spanning tree by removing selected links. For the internetwork shown in Figure 9.25(a), the pruned spanning trees produced by each MR for each of the three multicast addresses are as shown in Figure 9.25(b).

For each address, these are produced by starting at the tip of each branch of the basic spanning tree and pruning each line that does not form a path back to the root for this address. Packets are then only forwarded on those lines that make up the resulting pruned spanning tree. The protocol is known as the **multicast open shortest path first** (**MOSPF**) routing protocol

(a)



Multicast address table (MAT)

MR	Multicast addresses
1	A, B
2	A, B, C
3	B, C
4	C
5	C
6	A

Note: 1–8 are global line identifiers. These are also used as port numbers in the derivation of the spanning tree

(b)

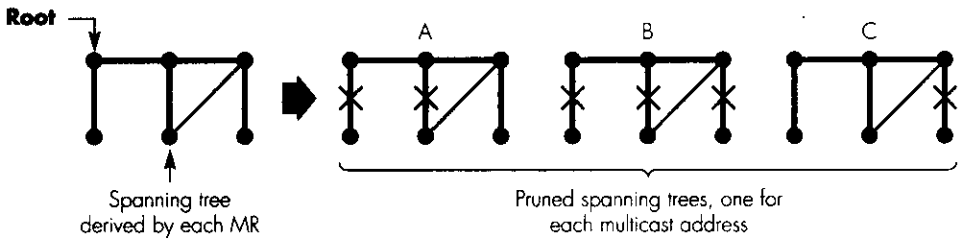


Figure 9.25 Spanning-tree multicast routing: (a) example topology and multicast address table; (b) set of spanning trees for each multicast address.

and, to explain how it works, assume that a packet with a (multicast) address of A is received from a network attached to MR1. The steps followed by each MR are as follows:

- On detecting the (multicast) address in the packet header is A, MR1 uses the pruned spanning tree for A to determine that a copy of the packet should be sent to MR2.
- On receipt of the packet, MR2 first determines from its MAT that a copy of the packet should be sent out onto its own network. It then uses its own copy of the pruned spanning tree for A to determine that a copy of the packet should also be sent to MR3.
- On receipt of the packet, MR3 first determines from its MAT that it has no interest in the packet. It then uses its own copy of the pruned spanning tree for A to determine that a copy of the packet should be sent to MR6.
- On receipt of the packet, MR6 first determines from its MAT that a copy of the packet should be sent out onto its own network. It then uses its own copy of the pruned spanning tree for A to determine that it is at the end of a branch of the tree and hence discards the packet.

9.6.10 IGMP

The two routing algorithms we described in the last section – the DVMRP and MOSPF – both assumed that each multicast router has stored in its multicast address table the complete list of multicast addresses that are currently in use and also the multicast addresses that each multicast router has an interest in. As we explained, the contents of the MAT are obtained by, first, each multicast router learning the set of multicast addresses of which all the hosts attached to the networks/subnets which the mrouter serves are currently members and second, this information is passed on to all the other mrouters. Clearly, the second step can be carried out using a broadcast procedure similar to that used in the distance vector algorithm to distribute network-wide routing information. Hence the unanswered question is how an mrouter learns the multicast addresses associated with its own attached networks/subnets. In practice, this is the role of the Internet group management protocol (IGMP) which we identified earlier in Figure 9.2. The IGMP is an integral part of the IP layer in all hosts and routers that support multicasting. It is defined in **RFC 1112.**

As we explained at the start of the last section, an application process (AP) within a host can join and leave a currently active multicast group at any time. To do this, the AP must know the 24-bit group address that has been allocated to this group by the IANA. As we showed in Figure 9.23, the AP uses this to derive both the IP multicast address of the group and the corresponding Ethernet group address. It then writes these into the MAT and the corresponding GAT respectively. In the case of a multicast session involving a group of APs in hosts that are all attached to the same LAN, this is all that is needed. In the case of a multicast session over the Internet, however, it is necessary for the host to inform its local mrouter the multicast address of the session that it wishes to join. The sequence of steps followed to do this are shown in Figure 9.26.

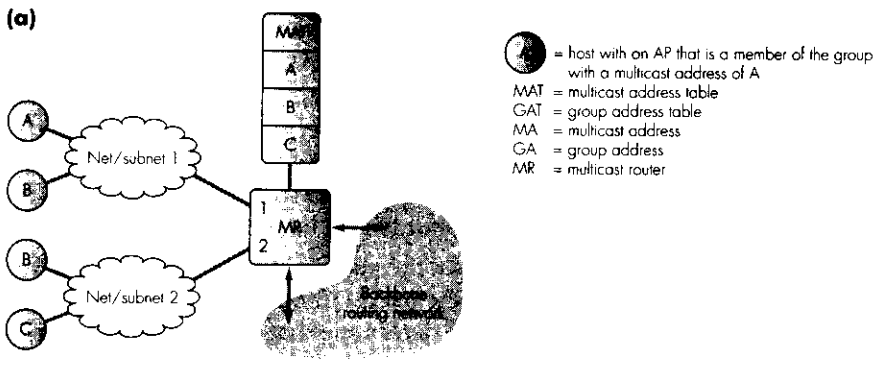


Figure 9.26 IGMP summary: (a) example network topology; (b) IGMP message transfer sequence to join and leave a multicast session; (c) IGMP message format.

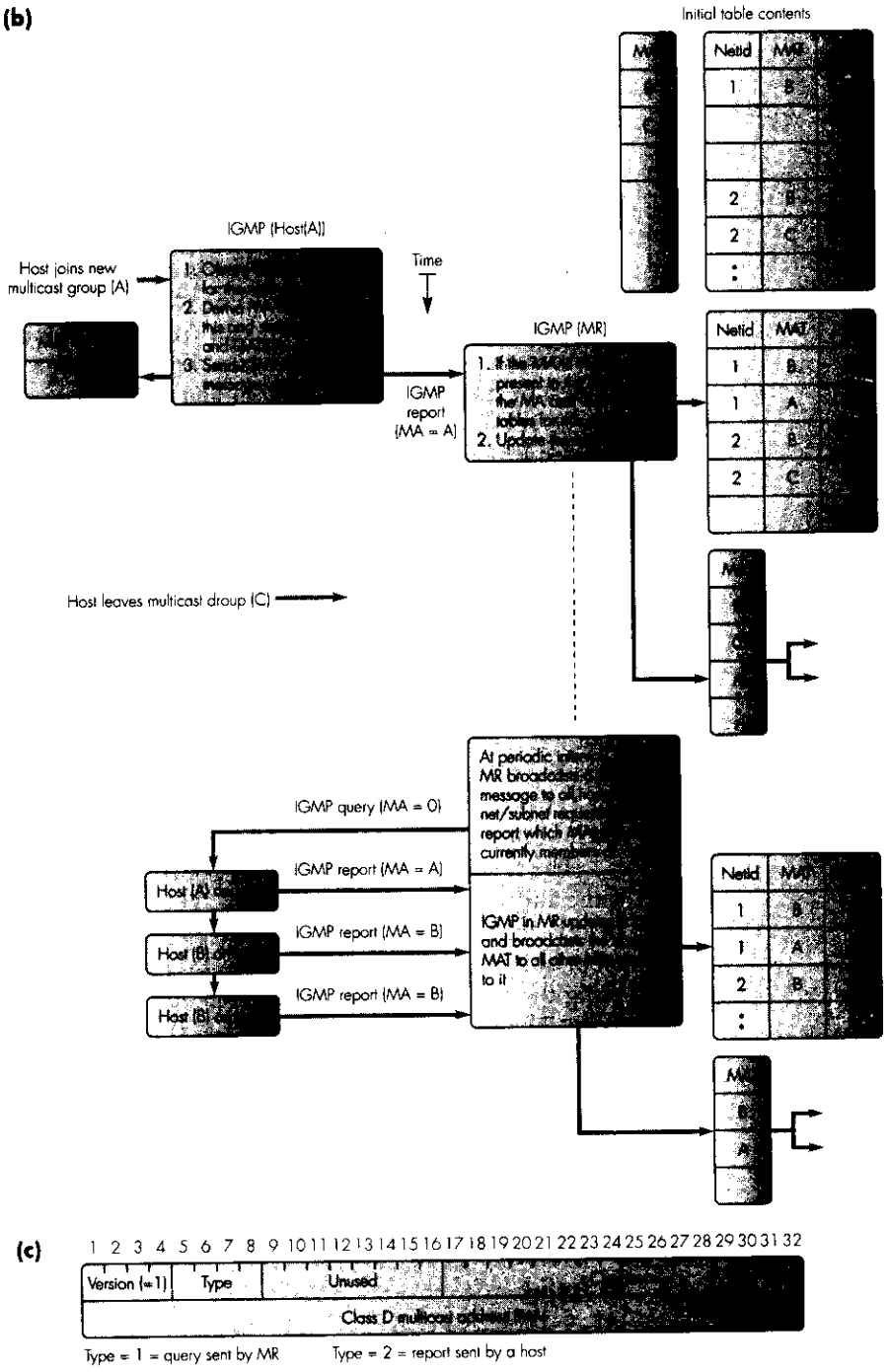


Figure 9.26 Continued

In this example, an AP running in a host that is attached to net/subnet 1 wishes to take part in a multicast session that has a derived multicast address of A. We assume that three other APs/hosts attached to the same multicast router are already members of two existing multicast groups, two with a multicast address of B and one with an address of C. In the case of B, one is attached to net/subnet 1 and the other to net/subnet 2 and, in the case of C, this is attached to net/subnet 2. Hence the contents of the initial tables are as shown in Figure 9.26(b).

First the IGMP in host (A) sends out a message – known as a *report* – to the IGMP in the attached mrouter (MR) containing the multicast address (MA) of the group it wishes to join (A). In addition to the MAT that is used for routing over the backbone network, a separate MAT and GAT are maintained by the MR for both net/subnet 1 and net/subnet 2. Hence on receipt of the report message, the IGMP first writes the MA into the MAT and the related Ethernet group address into the GAT of net/subnet 1. Also, since A is not already present in the backbone routing MAT, this is added to it and the new contents forwarded to each of the attached MRs.

Hosts do not need to inform an MR when it leaves a multicast group. At regular intervals, the IGMP in each MR broadcasts a *query* message to all hosts on each net/subnet requesting them to report of which MAs they are currently members. On receipt of a query, the IGMP in each host responds by returning a separate report message for each MA of which the host is currently a member. In the example shown in Figure 9.26, we assume the host that was a member of MA C has now left the group and hence a report message for C is not returned. Hence the IGMP in the MR removes the entries for C from its tables and proceeds to forward the updated table contents to each of the attached MRs.

The format of the two types of IGMP message – query and report – is the same and is shown in Figure 9.26(c). The *version* field is equal to 1 and the *type* field is either 0 for a report message (sent by a host) or 1 for a query message (sent by a MR). The *checksum* applies to the complete message and is computed using the same 1s complement procedure used with the header of an IP datagram. The *group address* is a 32-bit class D multicast address. In a query message it is set to all 0s and in a report it is set to the multicast group address being reported. Both are transmitted over the attached net/subnet in an IP packet with a *protocol* value of 2 and a *time-to-live* value of 1. In the case of a query, the *destination IP address* is the all-hosts broadcast address of 224.0.0.1 and the *source IP address* is the IP address of the MR. In the case of a report, the two addresses are the group MA being reported and the host IP address respectively.

9.6.11 M-bone

As we explained in Section 9.6.9 when we discussed the two multicast routing algorithms, currently, only a subset of the routers that make up the Internet global internetwork are capable of routing IP packets which have a multicast

destination address. These are known as multicast routers (mrouters) and the network formed by the interconnected set of mrouters, the **multicast backbone (M-bone) network**. The two routing algorithms we described in Section 9.6.9 are then used to route multicast packets between the mrouters that make up the M-bone.

In practice, therefore, because only a subset of the routers in the global internetwork can route multicast packets, there may be other routers that do not support multicasting present in the physical path that links two mrouters together. Hence many of the transmission paths that link the mrouters that form the M-bone are logical links that are implemented using IP tunneling. As we explained in Section 9.6.7, with tunneling, in order to send an IP datagram containing a multicast group address over the logical link that links two mrouters together, the datagram is carried in the user data field of a second IP datagram. This has the (known) IP unicast address of the intended destination mrouter in the destination IP address field of the second datagram. In this way, the packet is routed over the global internetwork in the same way as a unicast packet using either the distance vector or the link-state shortest-path-first algorithm. Then, on arrival at the destination mrouter, the latter extracts the multicast datagram contained within the packet and proceeds to route it using one of the two related multicast routing algorithms we described in Section 9.6.9.

As we can deduce from this, in order to implement the M-Bone, each mrouter has the IP (unicast) address of the adjacent mrouters that are (logically) connected to it stored in their connectivity/adjacency table. Normally, this information is entered by the management authority responsible for the network in which the mrouter is located. In this way, the possible presence of other (unicast) routers between two mrouters is transparent to the two mrouters which simply route (multicast) packets as if there is a physical transmission line linking them together.

9.7 ICMP

The Internet control message protocol (ICMP) forms an integral part of all IP implementations. It is used by hosts, routers and gateways for a variety of functions, and especially by network management. The main functions associated with the ICMP are as follows:

- error reporting,
- reachability testing,
- congestion control,
- route-change notification,
- performance measuring,
- subnet addressing.

The message types associated with each of these functions are shown in Table 9.2. Each is transmitted in a standard IP datagram.

Since the IP is a best-effort (unacknowledged) protocol, packets may be discarded while they are in transit across the Internet. Of course, transmission errors are one cause, but packets can be discarded by a host, router, or gateway for a variety of reasons. In the absence of any error reporting functions, a host does not know whether the repeated failure to send a packet to a given destination is the result of a poor transmission line (or other fault within a network) or simply the destination host being switched off. The various messages associated with the **error reporting** function are used for this purpose.

Table 9.2 ICMP message types and their use

Function	ICMP message(s)	Use
Error reporting	Destination unreachable	A datagram has been discarded due to the reason specified in the message
	Time exceeded	Time-to-live parameter in datagram expired and hence discarded
	Parameter error	A parameter in the header of a datagram is unrecognizable
Reachability testing	Echo request/reply	Checks the reachability of a specified host or gateway
Congestion control	Source quench	Requests a host to reduce the rate at which datagrams are sent
Route exchange	Redirect	Used by a gateway to inform a host attached to one of its networks to use an alternative gateway on the same network for forwarding datagrams to a specific destination
Performance measuring	Time-stamp request/reply	Determines the transit delay between two hosts
Subnet addressing	Address mask request/reply	Used by a host to determine the address mask associated with a subnet

ICMP = Internet control message protocol

If a packet is corrupted by transmission errors, it is simply discarded. If a packet is discarded for any other reason, the ICMP in the host, router, or gateway that discards the packet generates a *destination unreachable* error report message and returns it to the ICMP in the source host with a reason code. Reasons include the following:

- destination network unreachable,
- destination host unreachable,
- parameter problem,
- specified protocol not present at destination,
- fragmentation needed but don't fragment (DF) flag set in datagram header,
- communication with the destination network not allowed for administrative reasons,
- communication with the destination host not allowed for administrative reasons.

Although in most cases error reports are received as a result of some type of failure condition arising, in some instances an error report is used to gain some knowledge of the operational characteristics of the path/route followed by a packet. In general, the transfer of a message over the global internetwork is much quicker if no fragmentation is involved. Most networks (and subnetworks) support a maximum transmission unit (MTU) equal to or greater than 576 bytes. Hence one way of ensuring no fragmentation takes place is for the source IP to adopt this as the maximum size of all datagrams (including the IP header). In most cases, however, the actual MTU of the path will be greater than this and hence this would result in more packets being used to send each message than is necessary.

An alternative is for the source IP to use a procedure known as **path MTU discovery** to determine the MTU of a path/route prior to sending any datagrams relating to a session/call. Essentially, the first message received from the transport layer protocol relating to a new call/session is sent in a single datagram with the *don't fragment* bit set. Normally, if a router along the path followed by the packet cannot forward the packet over an attached link without fragmenting it, the router will return an ICMP error report with *fragmentation needed* as a reason code and an indication of the size of MTU that is possible. The source IP then adopts the latter as its own MTU to send all the remaining messages relating to the call/session.

Other error report messages include *time exceeded*, which indicates that the time-to-live parameter in a discarded packet has expired, and *parameter error*, which indicates that a parameter in the header of the discarded packet was not recognized.

If a network manager receives reports from a user that a specified destination is not responding, the reason must be determined using the

reachability testing function which is implemented in a program called **ping**. Typically, on receipt of such a report, the network manager initiates the sending of an *echo request* message to the suspect host to determine whether it is switched on and responding to requests. On receipt of an echo request message, the ICMP in the destination simply changes this to an *echo reply* message which it returns to the source. A similar test can be performed on selected routers and gateways if necessary.

If a packet is discarded because no free memory buffers are available as a result of a temporary overload condition, a *source quench* message is returned to the ICMP in the source host. Such messages can be generated either by a host or by a router or gateway. They request the source host to reduce the rate at which it sends packets. When a host receives such a message, it reduces the sending rate by an agreed amount. A new source quench message is generated each time a packet is discarded so that the source host incrementally reduces the sending rate. Such messages help to alleviate congestion within the global internetwork. Congestion is discussed further in Section 9.8.

When a network has multiple gateways attached to it, a gateway may receive datagrams from a host even though it determines from its routing table that they would be better sent via a different gateway attached to the same network. To inform the source host of this, the ICMP in the gateway returns a *redirect* message to the ICMP in the source indicating which is the better gateway to the specified destination. The ICMP in the source then makes an entry in its routing table for this destination.

An important operational parameter for an internet is the mean transit delay of packets/datagrams. This is a measure of the time a datagram takes to traverse the internet from a specified source to a specified destination. To ascertain this time, a host or a network manager can send a *time-stamp request* message to a specific destination. Each message contains the following three time-related parameters (known as *time-stamps*):

- the time the datagram was sent by the source,
- the time the datagram was received by the destination,
- the time the datagram was returned by the destination.

On receipt of a time-stamp request message, the ICMP in the destination simply fills in the appropriate time-stamp fields and returns the message to the source. On receipt of the reply, the source can quantify the current round-trip delay to that destination and from this determine the packet transit delay.

Finally, when subnet addressing is being used, the *address mask request* and corresponding reply messages are used by a host to ascertain the address mask associated with a local subnet. This is needed by a host to determine, for example, whether a specified destination is attached to the same subnet. The address mask is held by the local router associated with the subnet. The ICMP in a host can obtain the address mask by sending a request message and reading the mask from the reply.

9.7.1 ICMP message formats and transmission

The format of an ICMP message is shown in Figure 9.27. The first three fields are the same for all messages. The *type* field indicates the ICMP message type and these are related to the functions we listed earlier. For example, a *type* field of 0 relates to the error reporting function, 3 reachability testing, 4 congestion control, and so on. The *code* field then gives additional information such as the reason why a destination is unreachable. For example, 6 indicates the destination network is unknown and 7 the destination host is unknown. The *checksum* covers the entire ICMP message and uses the same algorithm as that used for the checksum present in the IP header. The number and meaning of the following 32-bit words then depend on the *type* and *code* fields in the header.

As we showed in Figure 9.3 and explained in the accompanying text, the *protocol* field in each datagram header is used to route the payload in a datagram to the appropriate protocol – ICMP, IGMP, OSPF, TCP, or UDP. In the case of reply messages, normally, these are for use by the local IP. When unsolicited error reports are received, in most cases, they are passed to an application-level process and result in an appropriate error message being output on the host screen.

9.8 QoS support

Congestion arises within a network when the demand for a network resource exceeds the level that is provided. For example, if a burst of packets arrive at a router (within the global internetwork) on a number of different input lines that all require the same output line, then the output line will become

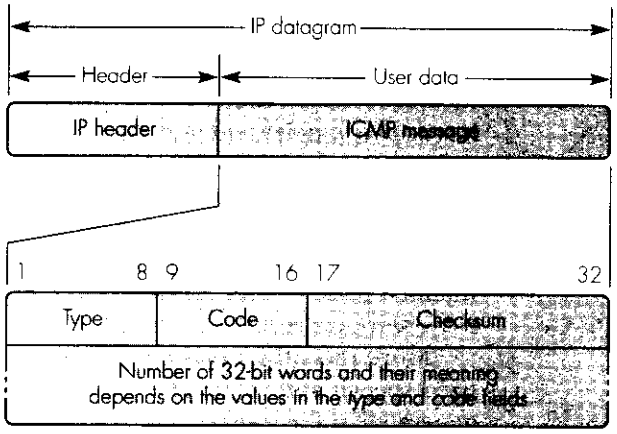


Figure 9.27 ICMP message format and transmission.

congested if the rate of arrival of packets is greater than the rate they can be output. To allow for this possibility, each output line has a first-in first-out (FIFO) queue associated with it which is used to hold a defined number of packets that are awaiting output on that line. Hence, providing the burst is of a relatively short duration and the number of packets to be queued is less than the number of packet buffers available, the congestion will be transient and the only effect should be a small increase in the end-to-end transfer delay experienced by each packet using that line. In the event of a longer burst, however, then all the packet buffers may become full and, as a result, some packets will have to be discarded.

Similarly, at a network-wide level, since the Internet is a best-effort connectionless network, the global internetwork will become congested if, over a sustained period, the aggregate rate at which packets are entering the internetwork exceeds its total capacity in terms of transmission bandwidth and packet buffers. As we saw in Section 1.5.5, associated with each call is a defined set of parameters which form what is called the minimum quality of service (QoS) requirements for the call. For example, with a packet-switched network like the Internet, these include a defined minimum mean packet throughput rate and a maximum end-to-end packet transfer delay. Hence, if as a result of congestion these requirements are not met, then the quality of the call may no longer be acceptable to the user. This is the case with applications involving real-time media streams, for example, such as Internet telephony.

As we can conclude from the above, two levels of congestion control are required, one that operates at the global internetwork level and the other that operates at the router level. The aim of the first is to limit the aggregate rate at which packets are entering the global internetwork to below its maximum rate, and the aim of the second is to maximize the flow of packets through each router. In the following subsections we discuss aspects of two of the schemes that are used to perform these functions.

9.8.1 Integrated services

Most early applications of the Internet were text based and hence relatively insensitive to delay and jitter. Examples include FTP and email, both of which can tolerate the added delays incurred by the use of a host-to-host retransmission control mechanism to overcome the effect of lost packets resulting from the best-effort service provided by IP. Other text-based applications, however, cannot tolerate the delay caused by retransmissions but nevertheless require minimal packet losses. Examples of this type of application are those relating to network control.

More recently, a number of interpersonal applications involving packetized speech and video were introduced. These require the packets that are generated at the source to be transferred over the Internet and played out at the destination in real time. This means that the retransmission of lost pack-

ets is not possible and that the packet flow is particularly sensitive to lost packets and jitter. Such applications also require a guaranteed minimum bandwidth. To meet this more varied set of QoS requirements, two schemes have been researched and standardized, one called integrated services (IntServ) and the other differentiated services (DiffServ). In this section we describe aspects of the IntServ scheme while aspects of DiffServ are described in Section 9.8.2.

In both schemes, the packets relating to the different types of call/session are each allocated a different value in the precedence bits of the *type of service* field of the IP packet header. This is used by the routers within the Internet to differentiate between the packet flows relating to the different types of call. The IntServ solution defines three different classes of service:

- guaranteed: in this class, a specified maximum delay and jitter and an assured level of bandwidth are guaranteed. It is intended for applications involving the playout of real-time streams;
- controlled load (also known as predictive): in this class, no firm guarantees are provided but the flow obtains a constant level of service equivalent to that obtained with the best-effort service at light loads. Examples of applications in this class are those involving real-time streams that have the capability of adjusting the amount of real-time data that is generated to the level that is offered;
- best-effort: this is intended for text-based applications.

To cater for the three different types of packet flows, within each router, three separate output queues are used for each line, one for each class. In addition, appropriate control mechanisms are used to ensure the QoS requirements of each class are met. We shall first discuss a number of these as these are used in both the IntServ and the DiffServ schemes.

Token bucket filter

This is used with each of the packet flows in both the guaranteed and predictive service classes. A portion of the bandwidth of the outgoing line and an amount of buffer/queue space is reserved for the packet flow relating to each call. A control mechanism called the token bucket filter is used to enforce these allocations so that the guaranteed QoS requirements in terms of bandwidth, delay, and jitter are met.

Associated with each flow is a container called a *bucket* into which tokens are entered at a rate determined by the bandwidth requirement of the flow. The size of the bucket is the same as the maximum amount of buffer/queue space the flow may consume. A packet relating to a flow can only be transferred to the output queue if there are sufficient tokens currently in the bucket. The number of tokens required is determined by the packet length and, if sufficient tokens are currently in the bucket, the packet is queued and the corresponding number of tokens taken from the bucket. As we can deduce

from this, therefore, providing the arrival rate of packets is less than or equal to the rate of entry of tokens into the bucket, then both the agreed bandwidth and delay/jitter will be met. If the arrival rate of packets exceeds the allocated bandwidth, normally these are relegated to the best-effort queue.

Weighted fair queuing

Since the packet rate – and hence bandwidth – associated with each flow may be different, when the packets relating to each flow are queued for transmission, in order to ensure the guaranteed delay bounds are met, it is necessary to ensure that the packets relating to each flow are not delayed by the packets of other flows. Hence a queue management scheme is also required to schedule the order that queued packets are transmitted. The **weighted fair queuing (WFQ)** scheme performs this function.

In order to ensure the delay bounds for each flow are met, the order of transmission of packets from the queue is changed each time a new packet arrives for queuing. When a packet arrives at an incoming line of the router it is given a time-stamp. This is determined from the arrival time of the packet and its scheduled departure time, the latter computed from the bandwidth associated with the flow and the packet length. The time-stamp of the packet is then compared with the time-stamps of the packets that are currently queued and the packet with the smallest time-stamp is transmitted first. In this way the delay bounds of each flow are met.

Random early detection

The requirements of the queue management scheme used with the best-effort queue are different from those of the other two queues. As we indicated earlier, normally, a router simply discards/drops a packet if the required output queue is already full. However, as we shall describe later in Section 12.3.2, with TCP, each time a packet relating to a call/session is lost, the TCP in the source host detects this and halves its current rate of entry of new packets for the call. Since this is done by all the hosts that lose a packet, the utilization of the link bandwidth falls dramatically. This is also detected by the affected TCPs which then quickly ramp-up the rate of entry of new packets. This, in turn, often results in full queues and dropped packets occurring again with the effect that the utilization of the available transmission bandwidth is poor. To stop this occurring, the **random early detection (RED)** queue management scheme is often used.

With RED, when a packet arrives for an output queue and the queue is full, instead of discarding the packet, a packet that is already in the queue is randomly selected for discarding. This has the affect that a reduced number of different applications are affected and hence the bandwidth utilization of the link is much improved.

To implement the scheme, two thresholds relating to the queue are defined: a minimum threshold (MinTH) and a maximum threshold (MaxTH). Also, the average length (AvrLEN) of the queue is continuously

monitored and used as a measure of the current level of traffic using the line. The action taken by the scheduler is determined by the current AvrLEN relative to the two thresholds as follows:

AvrLEN < MinTH:	the new packet is entered into the queue;
AvrLEN < MaxTH:	the new packet is dropped;
MaxTH < AvrLEN < MinTH:	a randomly-selected packet from the queue is dropped and the new packet is queued.

As we can deduce from the last condition, the probability of packets that are already in the queue being dropped increases as the AvrLEN increases. This has been found to give high levels of bandwidth utilization during periods of congestion.

Resource reservation protocol

With the IntServ scheme, in order to ensure the aggregate bandwidth of real-time traffic flows does not exceed that which is allocated for both the guaranteed and controlled-load traffic, the resources required for each flow (in terms of transmission bandwidth and buffer capacity) are reserved in advance of each packet flow starting. The protocol used to do this is called the **resource reservation protocol (RSVP)**.

Because many of the new real-time applications involve multiple participants, RSVP is used to reserve resources in each router along either a unicast or a multicast path. The actual routing of the packets associated with both types of call, however, are not part of RSVP and these are carried out in the normal way using one of the routing algorithms we described earlier in Section 9.6. A selection of the messages associated with RSVP are shown in Figure 9.28(a).

Each traffic flow is identified uniquely by the combined source and destination addresses in the IP header and the port number in the UDP header. To perform a reservation, the AP in the host that wishes to set up the call sends a *path* message in a UDP datagram with either the unicast or multicast address of the other host(s) in the destination address field of the IP header and the port number in the UDP datagram header. The purpose of the *path* message is firstly, to enable each router along the path/route followed by the packet to create an entry in a table known as the **path-state table** and secondly, to gather information about the resources that are currently available in each router along the path. The entry in the path-state table includes the flow identifier, a specification of the required parameters associated with the call – known as the traffic specification or Tspec – and the (IP) address of the router from which it received the *path* message.

On receipt of the *path* message, the AP (identified by the port number) in the destination host(s) uses the resource levels reported by each router to determine which type of call – guaranteed, controlled-load, or neither – the

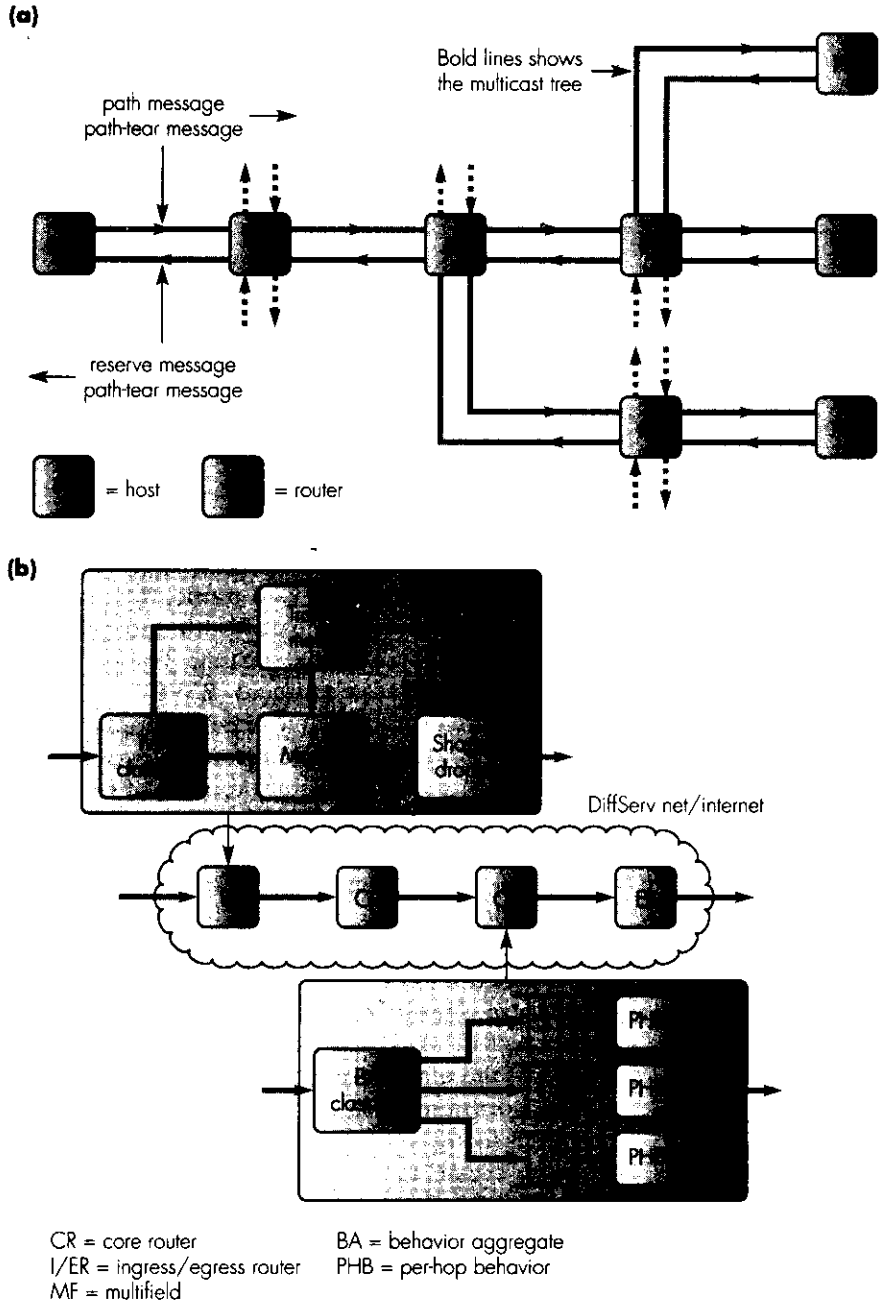


Figure 9.28 QoS support mechanisms: (a) RSVP principles; (b) DiffServ architecture.

path can support. Assuming the call can be accepted, the AP in each destination host then returns a *reserve* message (containing the Tspec for the call) to the router from which it received the *path* message. If the router still has the necessary resources, it reserves these for the flow (the resources may be different in each direction), makes an entry in the path-state table of the address of the router which sent it the message, and then sends the (*reserve*) message to the next router along the path. If the resources are not available, then a *path-tear* message is returned along the forward and return paths in order to release any resources that have been reserved and delete the entry in the path-state table. This procedure is repeated by each router along the path back to the source which, on receipt of the *reserve* message, proceeds with the call.

The path associated with each call/session may change during the lifetime of a call; for example, due to a router going down. To allow for this occurring, associated with each entry in the path-state table kept by each router is a timer called the *cleanup timer* and, should this expire, the entry is deleted. The timer is restarted each time a *path* message relating to the call is received. At periodic intervals – less than the cleanup timeout period – the source host sends a new *path* message which is acted upon by each router in the same way as the first *path* message. Hence should the new path not include a particular router, its cleanup timer will expire and the related path-state information be deleted. This mode of operation is known therefore as *soft-state* since it may change during a call.

Finally, on completion of the call/session, the AP that set up the call sends out a *path-tear* message which results in the entry in the path-table held by each router along the path(s) being deleted. The RFCs associated with RSVP and the other IntServ procedures are in **RFC 2205–2216**.

The major disadvantage of RSVP is that state information is retained by each router for each call/flow. Although this may be acceptable in a company intranet, for example, in a backbone router of the Internet the tables used for this purpose can be very large. Hence with the very high bit rates that are used, the heavy overheads per call can be unacceptably high. It is for this reason that the alternative DiffServ scheme was developed.

9.8.2 Differentiated services

Using the DiffServ approach, individual flows are not identified and instead the individual flows in each service class are aggregated together. Flows are then treated on a per-class basis rather than a per-flow basis. The general architecture used with a DiffServ network is shown in Figure 9.28(b).

The incoming packet flows relating to individual calls – referred to as microflows – are classified by the router/gateway at the edge of the DiffServ-compliant net/internet into one of the defined service/traffic classes by examining selected fields in the various headers in the packet. Within the DiffServ network the *type of service* (*TOS*) field in the IP packet header is

replaced by a new field called the *differentiated services (DS) field.* As we saw earlier in Section 9.2, this is an 8-bit field although currently only six bits are used for the DS field. The six bits form what is called the **(DS) packet codepoint (DSCP)** which is used to enable each router to determine the traffic class to which the packet data relates and the output queue into which the packet should be put. The queue management/scheduling procedure relating to each queue is called a **per-hop behavior (PHB)** and, since this applies to an aggregated set of packet flows, a PHB is said to be applied to a **behavior aggregate (BA)**.

Within the DiffServ network, a defined level of resources in terms of the buffer space within each router and the bandwidth of each output line is allocated to each traffic class using, for example, a token bucket filter. As each packet arrives at the ingress router it is first classified as belonging to a particular traffic class. In some instances, however, as we shall see later, the actual classification is also a function of how well the microflow (to which the packet relates) is conforming to the agreed traffic profile for the flow. This is determined by the **traffic meter module** and, based on the level of conformance, the traffic meter informs the **marker module** whether the packet should be marked with a low, medium, or high drop precedence. In addition, the traffic meter also informs the **shaper/dropper** module of this and the latter decides whether the packet should be dropped or allowed into the network.

Normally, real-time microflows with hard QoS guarantees are placed in the highest-priority traffic class. Typically, the traffic meter for this type of stream is a token bucket filter with a defined rate and bucket depth. Hence if an arriving packet relating to a flow in this class is deemed to be out-of-profile, the traffic meter informs the shaper/dropper module of this. The latter then either drops the packet or relegates it to the best-effort class by setting all six DSCP bits to zero.

Once within the network, as each packet arrives at a core router, the **BA classifier** first determines to which traffic class the packet belongs and hence to which output queue the packet should be transferred. Each queue is then serviced using an appropriate PHB. Currently, two PHBs have been defined: **expedited forwarding (EF)** and **assured forwarding (AF)**. The EF PHB is similar to the guaranteed service class associated with IntServ and hence has the highest priority. The PHB used with this is based on a queue scheduling procedure such as weighted fair queuing.

The AF PHB has four ordered traffic classes associated with it each of which has three drop procedure levels: low, medium, and high. Should congestion arise, this is used together with the traffic class to determine which packet(s) should be dropped. The PHB used in this case, therefore, is based on a queue scheduling procedure such as random early discard. The RFCs associated with DiffServ are in **RFC 2474–5**.

9.9 The PPP link layer protocol

As we showed in Figure 9.1 and explained in the accompanying text, access networks are connected to the Internet global internetwork by gateways. Also, as we showed in Figure 9.9, access gateways are connected to interior gateways which, in turn, are interconnected together using high bit rate synchronous transmission lines leased from a network provider. Typically, the leased lines are either from the PDH – DS1/3 or E1/3 for example – or from the SDH/SONET hierarchy – STS-3/STM-1 for example. Hence, since the various types of gateway (router) operate at the IP (network) layer, as we showed in Figure 9.1, a link layer protocol is required to transfer the IP packets/datagrams over the different types of leased line.

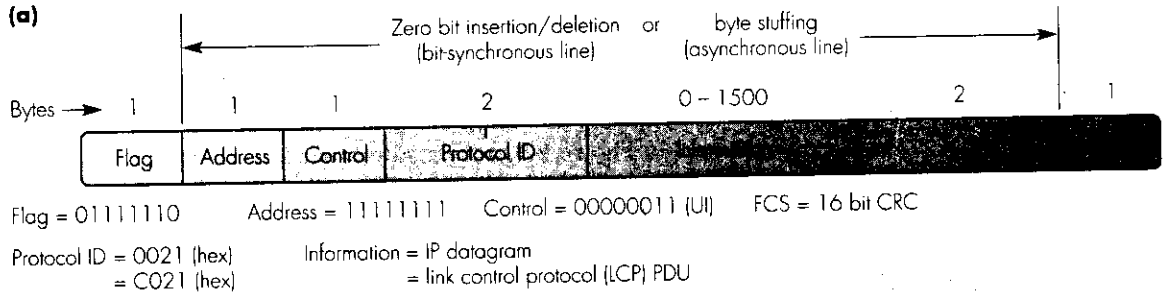
In addition, many of the access networks provided by ISPs require a link layer protocol to transfer the information entered by a person at home (using a PC for example) to the ISP network gateway. In many instances, the information is transferred over a switched connection through a PSTN using modems. Normally, the modem at home is connected to the serial port of the PC and, as we explained in Section 6.4, character-oriented asynchronous transmission is usually used. In order to avoid the proliferation of many different protocols, the IETF has defined a standard link layer protocol to meet these requirements. This is known as the **point-to-point protocol** (**PPP**) and is defined in **RFC 1661/2** and **3**.

To give the PPP the necessary flexibility, it has a number of features that enable it to be used in these and other applications. For example, it can operate in either the connection-oriented (reliable) or connectionless (best-effort) mode and with a variety of different types of network layer protocol. The latter feature is necessary, for example, if the access network uses the IPX network protocol rather than the IP. The PPP is based on the HDLC protocol we described in Section 6.8 and the format of all frames is shown in Figure 9.29.

Although the HDLC protocol is bit-oriented, in the PPP all frames are made up of an integral number of bytes/octets encapsulated by the opening and closing flag byte of 01111110. To achieve data transparency with the different types of synchronous transmission lines, zero bit insertion and deletion is used, the operation of which we described in Section 6.5.3.

In the case of asynchronous transmission lines, all characters are made equal to 8 bits and a form of character/byte stuffing is used, the principle of which we described in Section 6.4.3. In this application, however, in addition to flag bytes, a number of different characters/bytes are transmitted transparently. These include all the 32 control characters in the ASCII character set we identified in Figure 2.6(a). The list of bytes/characters and their codes is given in Figure 9.29(b).

As we can see, the escape byte/character used is 01111101 – 7D (hex) – and, whenever this is inserted, the sixth bit of the following byte/character is complemented. This rule is also used to transfer a byte/character that is equal to the escape byte.



(b)

Byte/character	Normal encoding	Byte-stuffed encoding	
Flag	01111110	01111101	01011110
NUL	00000000	01111101	00100000
SOH	00000001	01111101	00100001
⋮			
SI	00001111	01111101	00101111
US	00011111	01111101	00111111
Escape	01111101	01111101	01011101

Figure 9.29 The point-to-point protocol (PPP): (a) frame format; (b) byte stuffing principle.

Since the PPP is intended for use over point-to-point lines, the *address* field has no role to play. Hence it is always set to all binary 1s. Also, since in most applications all information is transferred over the line in the connectionless mode, as we explained in Section 6.8, the default value in the *control* field is 03 (hex) which is the code used to indicate an unnumbered information (UI) frame.

The default length of the *protocol ID* field is two bytes. It is used to indicate the type of packet – and hence network layer protocol – that is present in the *information* field. For example, a value of 0021 (hex) is used to indicate an IP packet/datagram is present. In addition, since the PPP can be used in a variety of applications, a single request-response protocol known as the **link control protocol** (**LCP**) has been defined to allow the operational mode and associated parameters for the line/link to be negotiated. Associated with the LCP are a number of messages – protocol data units (PDUs) – and, when one is present in the information field, the value in the *protocol ID* field is C021 (hex). For example, when transferring only IP datagrams over a line, an LCP *request* message can be sent to propose the use of a reduced frame header with no address or control fields and just a single byte for the protocol field. The other side then responds by returning either an *accept* message or a *reject* message in which case the default values must be used.

The default maximum size of the information field is 1500 bytes although a different maximum size can be negotiated using LCP request-

response messages. The *FCS* field is used to detect the presence of transmission errors in the frame and, as we explained in Section 6.8, it is based on a CRC. The FCS has a default length of 16 bits but a length of 32 bits can be negotiated using the LCP.

9.10 IPv6

Until the mid-1990s the Internet was used primarily by universities, government agencies, research establishments, and some sectors of industry. Since that time, however, it has gone through unprecedented growth owing to a large extent to the rapid rise in interest in the use of the World Wide Web. As a result, most schools, colleges, and many homes now have PCs with connections to the Internet. These are now being used, in addition to Web access, to exploit the range of other applications that are supported by the Internet. Moreover, this growth is predicted to increase even faster as new applications emerge: for example, the widespread use of hybrid mobile phones/computers with Internet interfaces, television set-top boxes with integral Web browsers, plus a potentially vast number of consumer products – such as meters, household appliances, office equipment, and so on – many of which may have an Internet interface.

Although the introduction of CIDR has extended considerably the usable address space of IPv4, the IETF has already defined and is using a new version of IP which has a number of features that have been introduced to meet the predicted growth levels. This is known as **IP version 6** (**IPv6**) or sometimes **IP next generation** (**IPng**). It is defined in **RFCs 1883–7** and a number of supporting RFCs. In addition to providing a large increase in the number of IP addresses, the IETF has taken the opportunity to correct some of the deficiencies associated with IPv4 and to provide a number of other features. The main new features of IPv6 are:

- a much increased address space from 32 bits to 128 bits;
- hierarchical addresses to reduce the size of the routing tables associated with the routers in the core backbone network;
- a simplified header to enable routers and gateways to process and route packets faster;
- the introduction of improved security and data integrity features including authentication and encryption;
- an autoconfiguration facility that enables a host to obtain an IP address via the network without human intervention;
- harder quality-of-service guarantees by means of the preferential treatment by routers of the packets associated with interactive and multimedia applications relative to those relating to traditional applications such as email and file transfers;

- support for mobile computing by the use of autoconfiguration to obtain an IP address dynamically via the network for the duration of a call/session.

Although many of the above features require some radical changes to IPv4 – for example a different address structure and datagram/packet format – in terms of the protocols that are used within the expanded global internetwork, these operate in much the same way as the current IPv4 protocols. For example, the LS-SPF (OSPF) routing algorithm we described in Section 9.6.4 is used as the standard interior gateway protocol (IGP) for IPv6 except, of course, that 128-bit addresses are used in the link-state and routing tables. There is also an updated version of the RIP – based on the distance vector routing algorithm we described in Section 9.6.3 – called **RIPng**. Similarly, at the backbone level, the border gateway protocol (BGP) we described in Section 9.6.5 (including the CIDR we described in Section 9.6.6) is used but with extensions to allow reachability information based on IPv6 hierarchical addresses to be exchanged. Hence in the remainder of this section we limit our discussion to a selection of the new features that are used.

9.10.1 Datagram format

In relation to the IPv4 datagram/packet header, a number of fields have been dropped and others have been made optional. The result is a basic/main header of 40 bytes, the contents and format of which are shown in Figure 9.30(a). The use of each field is as follows:

Version

This is set to 6 to enable routers to discriminate IPv6 packets from IPv4 packets. It is envisaged that both will need to coexist for many years.

Traffic class

This field plays a similar role to the *ToS* byte in the IPv4 header. It allows the source IP to allocate a different priority to packets relating to, say, multimedia applications involving real-time streams from those relating to traditional applications. It contains a 4-bit priority field and hence 16 priorities are possible, the higher the priority value the higher the packet priority. Values in the range 0–7 are for packets relating to applications for which best-effort delivery is acceptable; for example network news (1) and FTP(4). Both these applications are less sensitive to delay and delay variation (jitter) than applications involving real-time media but FTP is more sensitive to packet loss than network news. Values in the range 8–15 are for packets containing real-time streams such as audio and video. Typically, such streams/packets are more sensitive to delay and jitter than the packets in the first category and hence should be transmitted before them during periods of congestion. Also, within the second category, packets containing compressed video are more sensitive to packet loss than packets containing just audio and hence are given a higher priority.

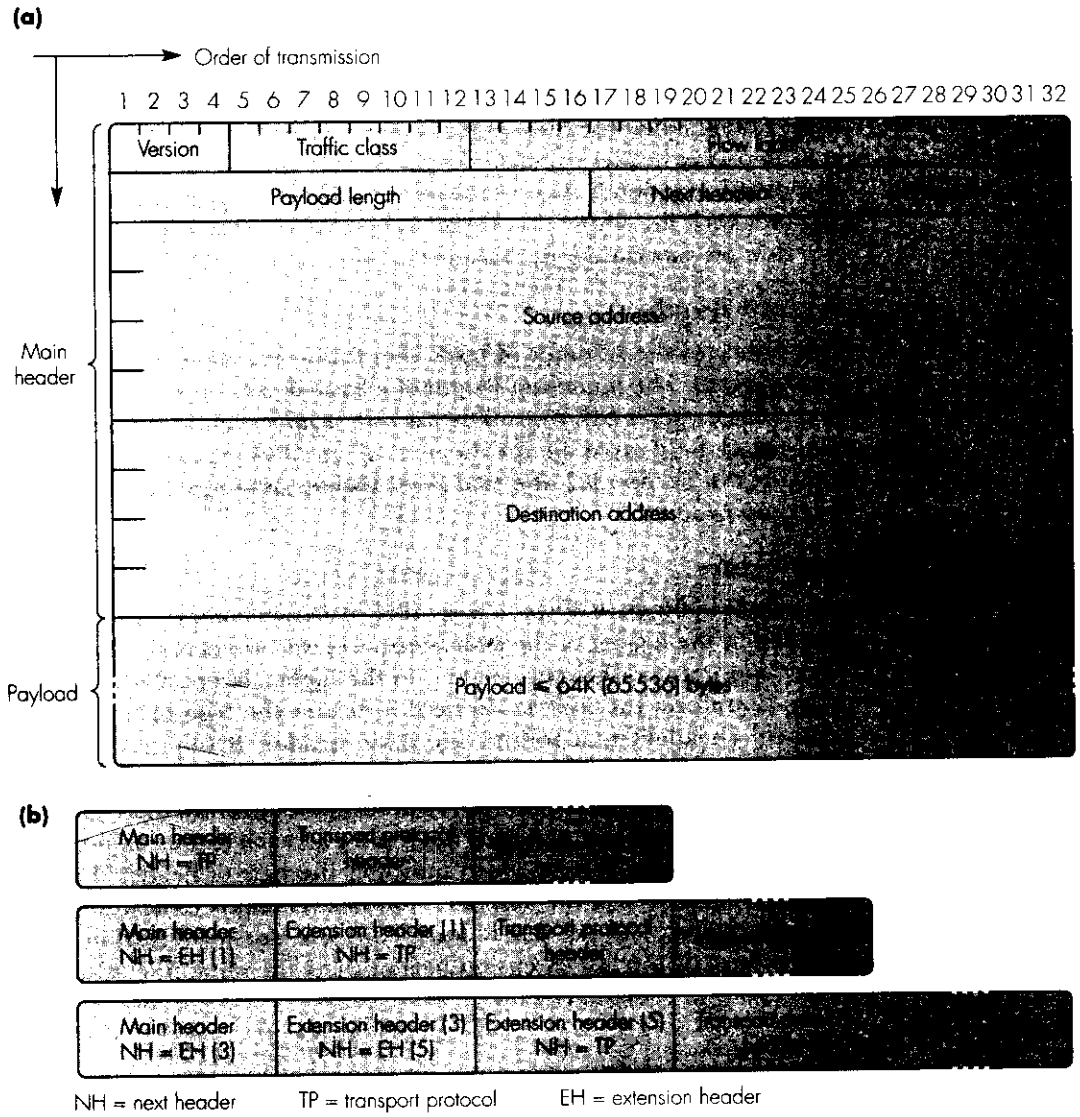


Figure 9.30 IPv6: (a) main header fields and format; (b) position and order of extension headers.

Flow label

This is a new field and is closely linked to the *traffic class* field. It is set to zero in best-effort packets and, in packets in the second category, it is used to enable a router to identify the individual packets relating to the same call/session. As we saw earlier in Section 9.8.1, one approach to handling packets containing real-time streams is to reserve resources – for example

transmission bandwidth – for such calls in advance of sending the packets relating to the call. During the reservation procedure, the call is allocated a *flow label* by the source. Also, each router along the reserved path keeps a record of this, together with the source and destination IP addresses, in a table. Routers then use the combined *flow label* and source IP address present in each packet header to relate the packet to a specific call/flow. The related routing information is then retrieved from the routing table and this is used, together with the *traffic class* value, to ensure the QoS requirements of the call/flow are met during the forwarding procedure.

Payload length

This indicates the number of bytes that follow the basic 40-byte header in the datagram. The minimum length of a basic datagram is 536 bytes and the maximum length is 64K bytes. The *payload length* is slightly different from the *total length* field used in the header of an IPv4 datagram since, as we explained in Section 9.2, the *total length* includes the number of bytes in the datagram header.

Next header

As we show in Figure 9.30(b), a basic IPv6 datagram comprises a main header followed by the header of the peer transport layer protocol (TCP/UDP) and, where appropriate, the data relating to the higher layers. With a basic datagram, therefore, the *next header* field indicates the type of transport layer protocol (header) that follows the basic header. If required, however, a number of what are called ***extension headers*** can be inserted between the main header and the transport protocol header. Currently, there are six types of extension header defined and, when present, each extension header starts with a new *next header* field which indicates the type of header that follows. The *next header* field in the last extension header always indicates the type of transport protocol header that follows. Thus, the *next header* field in either the main header or the last extension header plays the same role as the *protocol* field in an IPv4 datagram header.

Hop limit

This is similar to the *time-to-live* parameter in an IPv4 header except the value is a hop count instead of a time. In practice, as we explained in Section 9.2, most IPv4 routers also use this field as a hop count so the change in the field's name simply reflects this. The initial value in the *hop limit* field is set by the source and is decremented by 1 each time the packet/datagram is forwarded. The packet is discarded if the value is decremented to zero.

Source address and destination address

As we indicated earlier, these are 128-bit addresses that are used to identify the source of the datagram and the intended recipient. In most cases this will be the destination host but, as we shall explain later, it might be the next

router along a path if source routing is being used. Unlike IPv4 addresses, an IPv6 address is assigned to the (physical) interface, not to the host or router. Hence in the case of routers (which have multiple interfaces) these are identified using any of the assigned interface addresses.

9.10.2 Address structure

As we showed in Figure 9.9, the various networks and internetworks that make up the global Internet are interconnected in a hierarchical way with the access networks at the lowest level in the hierarchy and the global backbone network at the highest level. However, the lack of structure in the netid part of IPv4 addresses means that the number of entries in the routing tables held by each gateway/router increases with increasing height in the hierarchy. At the lowest level, most access gateways associated with a single site LAN have a single netid while at the highest level, most backbone routers/gateways have a routing table containing many thousands of netids.

In contrast, the addresses associated with telephone networks are hierarchical with, for example, a country, region, and exchange code preceding the local number. This has a significant impact on the size of the routing tables held by the switches since, at a particular level, all calls with the same preceding code are routed in the same way. This is known as **address aggregation**.

As we explained earlier in Section 9.6.6, classless inter-domain routing is a way of introducing a similar structure with IPv4 addresses and reduces considerably the size of the routing tables held by the routers/gateways in the global backbone. From the outset, therefore, IPv6 addresses are hierarchical. Unlike telephone numbers, however, the hierarchy is not constrained just to a geographical breakdown. The large address space available means that a number of alternative formats can be used. For example, to help interworking with existing IPv4 hosts and routers, there is a format that allows IPv4 addresses to be embedded into an IPv6 address. Also, since the majority of access networks are now Internet service provider (ISP) networks, there is a format that allows large blocks of addresses to be allocated to individual providers. The particular format being used is determined by the first set of bits in the address. This is known as the **prefix format** (**PF**) and a list of the prefixes that have been assigned – together with their usage – is given in Figure 9.31(a).

Unicast addresses

As we can see, addresses starting with a prefix of 0000 0000 are used to carry existing IPv4 addresses. There are two types, the formats of which are shown in Figure 9.31(b). As we shall expand upon in Section 9.11, a common requirement during the transition from IPv4 to IPv6 is to tunnel the IPv6 packets being generated by the two communicating IPv6 hosts – often written **V6 hosts** – over an existing IPv4 network/internetwork. Hence to simplify the routing of the IPv4 packet – containing the IPv6 packet within it – the IPv6

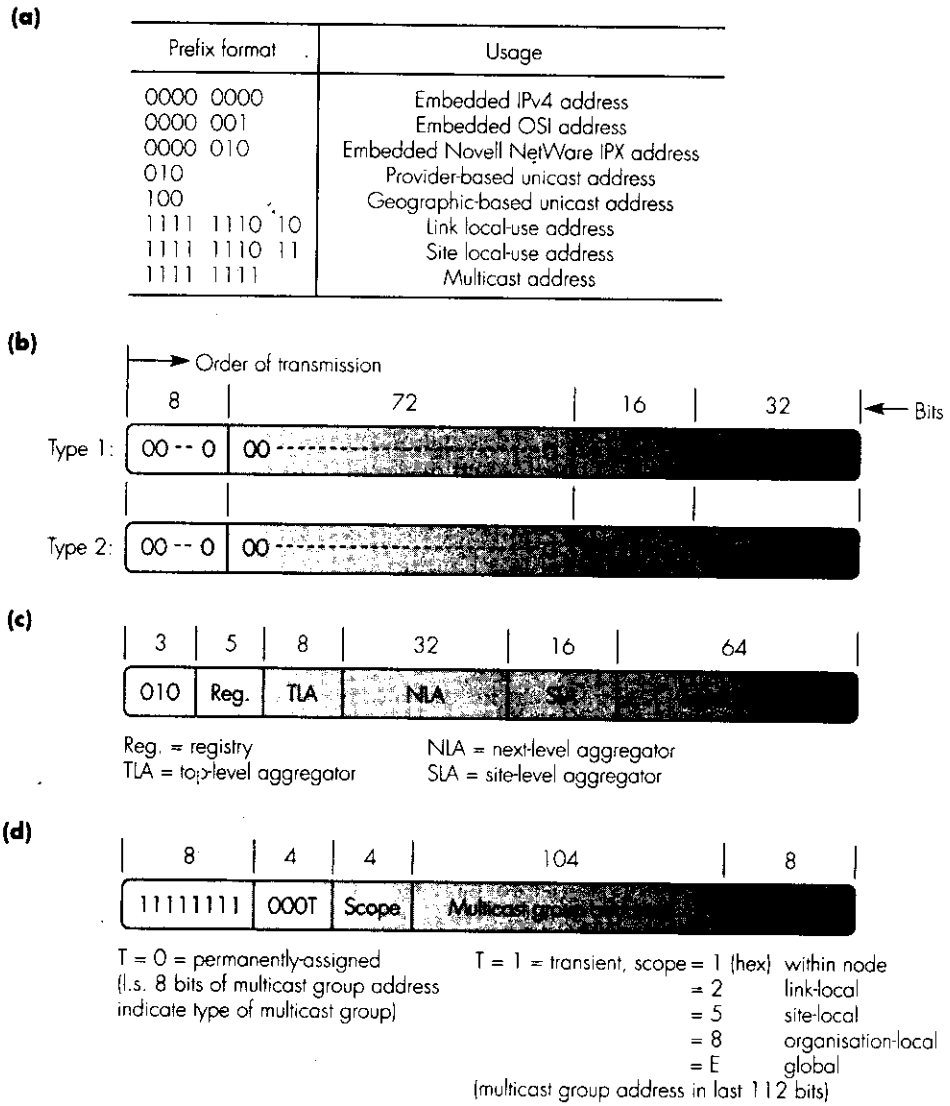


Figure 9.31 IPv6 addresses: (a) prefix formats and their use; (b) IPv4 address types; (c) provider-based unicast address format; (d) multicast address format.

address contains the IPv4 address of the destination gateway embedded within it. The second type is to enable a V4 host to communicate with a V6 host. The IPv4 address of the V4 host is then preceded by 96 zeros. In addition, two addresses in this category are reserved for other uses. An address comprising all zeros indicates there is no address present. As we shall expand

upon in Section 9.10.4, an example of its use is for the source address in a packet relating to the autoconfiguration procedure. An address with a single binary 1 in the least significant bit position is reserved for the loopback address used during the test procedure of a host protocol stack.

The OSI and NetWare address prefixes have been defined to enable a host connected to one of these networks to communicate directly with a V6 host. The most widely used is the provider-based prefix as this format reflects the current structure of the Internet. A typical format of this type of address is shown in Figure 9.31(c). As we saw in Figure 9.9, the core backbone of the Internet consists of a number of very high bandwidth lines that interconnect the various continental backbones together. The routers that perform this function are owned by companies known as **top-level aggregators (TLAs)**. Each TLA is allocated a large block of addresses by what is called a **registry**, the identity of which is in the field immediately following the 010 prefix. Examples include the North American registry, the European registry, the Asia and Pacific registry, and so on.

From their allocation, the TLAs allocate blocks of addresses to large Internet service providers and global enterprises. These are known as **next-level aggregators (NLAs)** and, in the context of Figure 9.9, operate at the continental backbone and national and regional levels. The various NLAs allocate both single addresses to individual subscribers and blocks of addresses to large business customers. The latter are known as **site-level aggregators (SLAs)** and include ISPs that operate at the regional and national levels. The 64-bit **interface ID** is divided locally into a fixed subnetid part and a hostid part. Typically, the latter is the 48-bit MAC address of the host and hence 16 bits are available for subnetting.

As we can deduce from Figure 9.31(c), the use of hierarchical addresses means that each router in the hierarchy can quickly determine whether a packet should be routed to a higher-level router or to another router at the same level simply by examining the appropriate prefix. Also, the routers at each level can route packets directly using the related prefix. The same overall description applies to the processing of geographic-based addresses.

As their names imply, the two types of **local-use addresses** are for local use only and have no meaning in the context of the global Internet. As we shall expand upon in Section 9.10.4, *link local-use addresses* are used in the autoconfiguration procedure followed by hosts to obtain an IPv6 address from a local router. The router only replies to the host on the same link the request was received and hence this type of packet is not forwarded beyond the router.

The *site local-use addresses* are used, for example, by organizations that are not currently connected to the Internet but wish to utilize the technology associated with it. Normally, the 64-bit interface ID part is subdivided and used for routing purposes within the organization. In this way, should the organization wish to be connected to the Internet at a later date, it is only necessary to change the site local-use prefix with the allocated subscriber prefix.

Multicast addresses

The format of an IPv6 multicast address is shown in Figure 9.31(d). As we can see, following the multicast prefix are two additional fields which have been introduced to limit the geographic scope of the related multicast operation. The first is known as the *flags* field and is used to indicate whether the multicast is a permanently-assigned (reserved) address (0000) or a temporary (transient) address (0001). In the case of a permanently-assigned address, the least significant 8 bits of the *multicast group address* field identify the type of the multicast operation, while for a transient address, the full 112 bits identify the multicast group address.

In both cases, the 4-bit *scope* field defines the geographic scope of the multicast packet. The various alternatives are identified in the figure and mrouters use this field to determine whether the (multicast) packet should be forwarded further or discarded.

Anycast addresses

In addition to unicast and multicast addresses, a new address type known as an ***anycast group address*** has been defined. These are allocated from the unicast address space and are indistinguishable from a unicast address. With an anycast address, however, a group of hosts or routers can all have the same (anycast) address. A common requirement in a number of applications is for a host or router to send a packet to any one of a group of hosts or routers, all of which provide the same service. For example, a group of servers distributed around a network may all contain the same database. Hence in order to avoid all clients needing to know the unique address of its nearest server, all the servers can be members of the same anycast group and hence have the same address. In this way, when a client makes a request, it uses the assigned anycast address of the group and the request will automatically be received by its nearest server. Similarly, if a single network/internetwork has a number of gateway routers associated with it, they can all be allocated the same anycast address. As a result, the shortest-path routes from all other networks/internetworks will automatically use the gateway nearest to them.

In order to perform the routing function, although an anycast address has the same format as a unicast address, when an anycast address is assigned to a group of hosts or routers, it is necessary for each host/router to be explicitly informed – by network management for example – that it is a member of an anycast group. In addition, each is informed of the common part of the address prefixes which collectively identify the topological region in which all the hosts/routers reside. Within this region, all the routers then maintain a separate entry for each member of the group in its routing table.

Address representation

A different form of representation of IPv6 addresses has been defined. Instead of each 8-bit group being represented as a decimal number (with a dot/period between them), groups of 16 bits are used. Each 16-bit group is

then represented in its hexadecimal form with a colon between each group. An example of an IPv6 address is:

FEDC:BA98:7654:3210:0000:0000:0000:0089

In addition, a number of abbreviations have been defined:

- One or more consecutive groups of all zeros can be replaced by a pair of colons.
- Leading zeros in a group can be omitted.

Hence the preceding address can also be written as:

FEDC:BA98:7654:3210:89

Also, for the two IPv4 embedded address types, the actual IPv4 address can remain in its dotted decimal form. Hence assuming a dotted decimal address of 15.10.0.6, the two embedded forms are:

:: 150.10.0.6 (IPv4 host address)
:: FFFF:150.10.0.6 (IPv4 tunnel address)

Example 9.6

Derive the hexadecimal form of representation of the following link-local multicast addresses:

(i) a permanently-assigned multicast group address of 67,

(ii) a transient multicast group address of 317.

Answer:

The formats of the two types of multicast addresses were shown in Figure 9.31(d). Hence:

The most significant 16 bits are FF02 = permanently-assigned, link-local and FF12 = transient, link-local

(i) A permanently-assigned multicast group address of 67 = 0043 (hex)
Hence IPv6 address = FF02 :: 67

(ii) A transient multicast group address of 317 = 013D (hex)
Hence IPv6 address = FF12 :: 13D

9.10.3 Extension headers

As we have just indicated, if required, a number of extension headers can be added to the main header to convey additional information, either to the routers visited along the path followed or to the destination host. The six types of extension header currently defined are:

- **hop-by-hop options**: information for the routers visited along a path;
- **routing**: list of routers relating to source routing information;
- **fragment**: information to enable the destination to reassemble a fragmented message:
- **authentication**: information to enable the destination to verify the identity of the source;
- **encapsulating security payload**: information to enable the destination to decrypt the payload contents;
- **destination options**: optional information for use by the destination.

The two options headers can contain a variable number of option fields, each possibly of a variable length. For these, each option field is encoded using a **type-length-value** (**TLV**) format. The *type* and *length* are both single bytes and indicate the option type and its length (in bytes) respectively. The *value* is then found in the following number of bytes indicated by the *length*. The option type identifiers have been chosen so that the most significant two bits specify the action that must be taken if the type is not recognized. These are:

00 ignore this option and continue processing the other option fields in the header;

01 discard the complete packet;

10 discard the packet and return an ICMP error report to the source indicating a parameter problem and the option type not recognized;

11 same as for 10 except the ICMP report is only returned if the destination address is not a multicast address.

Note also that since the type of extension header is indicated in the preceding *next header* field, the related decoder that has been written to decode the contents of the header is invoked automatically as each header is processed. Some examples of each header type now follow.

Hop-by-hop options

This type of header contains information that must be examined by all the gateways and routers the packet visits along its route. The *next header* value for this is 0 and, if this header is present, it must follow the main header. Hence the *next header* field in the main header is set to 0. An example of its use is for a host to send a datagram that contains a payload of more than 64K bytes.

This is particularly useful, for example, when a host is transferring many very large files over a path that supports a maximum transmission unit (MTU) significantly greater than 64K bytes. Datagrams that contain this header are known as **jumbograms** and the format of the header is shown in Figure 9.32(a).

As we can see, this type of header is of fixed length and comprises two 32-bit words (8 bytes). The *header extension length* field indicates the length of the header in multiples of 8 bytes, excluding the first 8 bytes. Hence in this case the field is 0. This option contains only one option field, the *jumbo payload length*. As we indicated earlier, this is encoded in the TLV format. The *type* for this option is 194 (11000010) and the *length* is 4 (bytes). The *value* in the *jumbo payload length* is the length of the packet in bytes, excluding the main header but including the 8 bytes in the extension header. This makes the *payload length* in the main header redundant and hence this is set to zero.

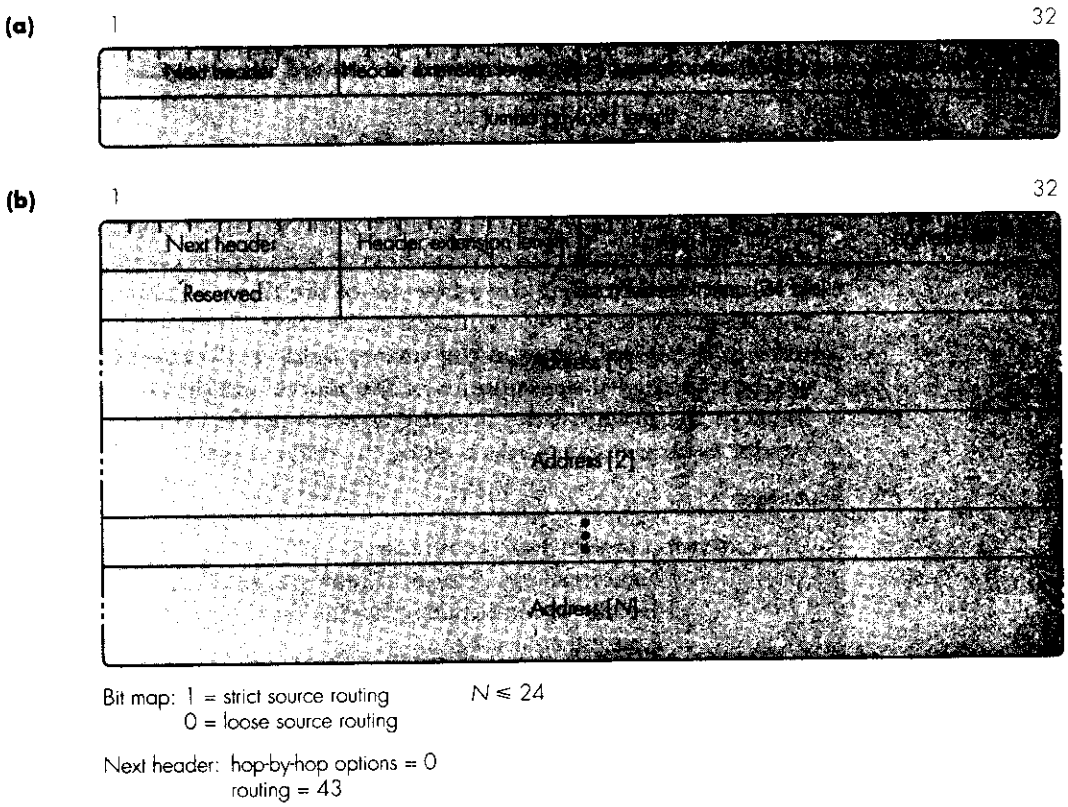


Figure 9.32 Extension header formats: (a) hop-by-hop options; (b) routing.

Routing

This plays a similar role to the (strict) *source routing* and *loose source routing* optional headers used in IPv4 datagrams. The *next header* value for this type of header is 43. Currently only one type of routing header has been defined, the format of which is shown in Figure 9.32(b).

The *header extension length* is the length of the header in multiples of 8 bytes, excluding the first 8 bytes. Hence, as we can deduce from the figure, this is equal to two times the number of (16-byte) addresses present in the header. This must be an even number less than or equal to 46. The *segments left* field is an index to the list of addresses that are present. It is initialized to 0 and is then incremented by the next router in the list as it is visited. The maximum therefore is 23.

The second 4-byte word contains a *reserved* byte followed by a 24-bit field called the *strict/loose bit map*. This contains one bit for each potential address present starting at the leftmost bit. If the related bit is a 1, then the address must be that of a directly-attached (neighbor) router – that is, strict source routing. If the bit is a 0, then the address is that of a router with possibly several other routers in between – loose source routing. The latter is used, for example, when tunneling is required to forward the datagram. In the case of

Example 9.7

A datagram is to be sent from a source host with an IPv6 address of A to a destination host with an IPv6 address of B via a path comprising three IPv6 routers. Assuming the addresses of the three routers are R1, R2, and R3 and strict source routing is to be used, (i) state what the contents of the initial values in the various fields in the extension header will be and (ii) list the contents of the source and destination address fields in the main header and the *segments left* field in the extension header as the datagram travels along the defined path.

Answer:

(i) Extension header initial contents:

Next header = Transport layer protocol
Header extension length = 2 × 3 = 6
Routing type = 0
Segments left = 0
Strict/loose bit map = 11100000 00000000 00000000
List of addresses = R1, R2, R3 (each of 16 bytes)

(ii) Contents of main header fields:

At source	SA = A	DA = R1	Segments left = 0
At R1	SA = A	DA = R2	Segments left = 1
At R2	SA = A	DA = R3	Segments left = 2

strict source routing, at each router the destination address in the main header is changed to that obtained from the list of addresses before the index is incremented. In the case of loose source routing, the destination address will be that of an attached neighbor which is on the shortest-path route to the specified address.

Fragment

This header is present if the original message submitted by the transport layer protocol exceeds the MTU of the path/route to be used. The *next header* value for a *fragment* extension header is 44. The fragmentation and reassembly procedures are similar to those used with IPv4 but, in the case of IPv6, the fragmentation procedure is carried out only in the source host and not by the routers/gateways along the path followed by the packet(s). As we saw in Figure 9.30(a), there is no don't fragment (D) bit in the IPv6 main header since, in order to speed up the processing/routing of packets, IPv6 routers do not support fragmentation. Hence, as we explained in Section 9.7, either the minimum MTU size of 576 bytes must be used or the *path MTU discovery* procedure is used to determine if the actual MTU size is greater than this. In either case, if the submitted message (including the transport protocol header) exceeds the chosen MTU (minus the 40 bytes for the IPv6 main header), then the message must be sent in multiple packets, each with a main header and a *fragment* extension header. The various fields and the format of each *fragment* extension header are shown in Figure 9.33(a). An example is shown in Figure 9.33(b).

Each packet contains a main header – plus, if required, a *hop-by-hop options* header and a *routing* header – followed by a *fragment* extension header and the fragment of the message being transmitted. Thus the maximum size of the payload (and hence message fragment) in each packet will be the MTU size being used minus the number of 8-byte fields required for the main header and any extension headers that are present. The *payload length* field in the main header of the first packet indicates the total number of bytes in the message being transmitted – including the IP header – plus the number of bytes that are required for the other extension headers that are being used. The *payload length* in the main header of the remaining packets indicates the number of bytes in the packet following the main header.

The various fields in each *fragment* header have similar functions to those used with IPv4. The *fragment offset* indicates the position of the first byte of the fragment contained within the packet relative to the start of the complete message being transmitted. Its value is in units of 8-bytes. The *M-bit* is the *more fragments bit*; it is a 1 if more fragments follow and a 0 if the packet contains the last fragment. Similarly, the value in the *identification* field is used by the destination host, together with the source address, to relate the data fragments contained within each packet to the same original message. Normally, the source uses a 32-bit counter (that is incremented by 1 for each new message transmitted) to keep track of the next value to use.

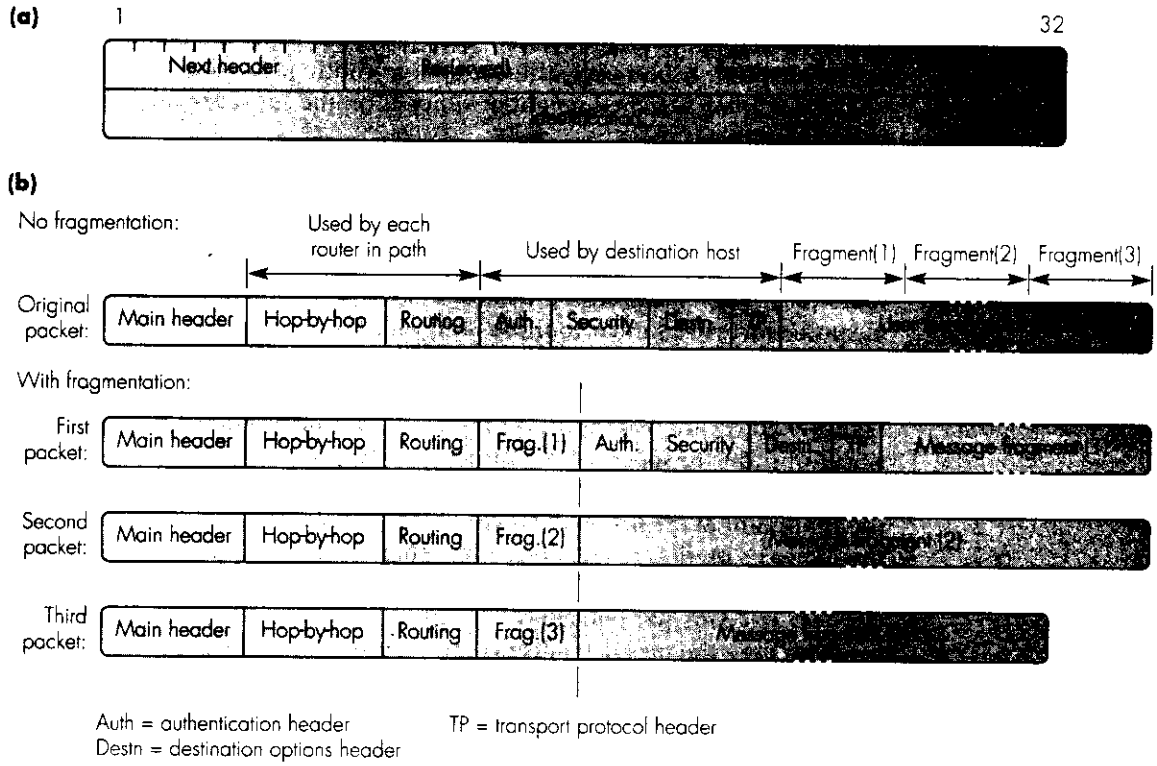


Figure 9.33 IPv6 fragmentation: (a) fragment header fields and format; (b) example.

Authentication and encapsulating security payload

As we shall expand upon in Section 13.4, authentication and the related subject of encryption are both mechanisms that are used to enhance the security of a message during its transfer across a network. In the case of authentication, this enables the recipient of a message to validate that the message was indeed sent by the source address present in the packet/datagram header and not by an impostor. Encryption is concerned with ensuring that the contents of a message can only be read by – and hence have meaning to – the intended recipient. The *authentication* and *encapsulating* security payload (ESP) extension headers are present when both these features are being used at the network layer.

When using IPv6 authentication, prior to any information (packets) being exchanged, the two communicating hosts first use a secure algorithm to exchange secret keys. An example is the MD5 algorithm we describe later in Section 13.5. Then, for each direction of flow, the appropriate key is used to compute a checksum on the contents of the entire datagram/packet. The computed checksum is then carried in the authentication header of the

packet. The same computation is repeated at the destination host and only if the computed checksum is the same as that carried in the authentication header is it acknowledged that the packet originated from the source host address indicated in the main header and also that the packet contents have not been modified during the packets transfer over the network. We discuss the subject of authentication further in Section 13.6.

The encryption algorithm used with the ESP is also based on the use of an agreed secret key. An example is the DES algorithm we describe later in Section 13.4.3. The agreed key is used to encrypt either the transport protocol header and payload parts of each packet or, in some instances, the entire packet including the main header and any extension headers present. In the case of the latter, the encrypted packet is then carried in a second packet containing a completely different main and, if necessary, extension headers. The first is known as **transport mode encryption** and the second **tunnel mode encryption**. Figure 9.34(a) illustrates the principle of operation of both modes.

As we can deduce from the figure, the overheads associated with the tunnel mode are significantly higher than those of the transport mode. The extra

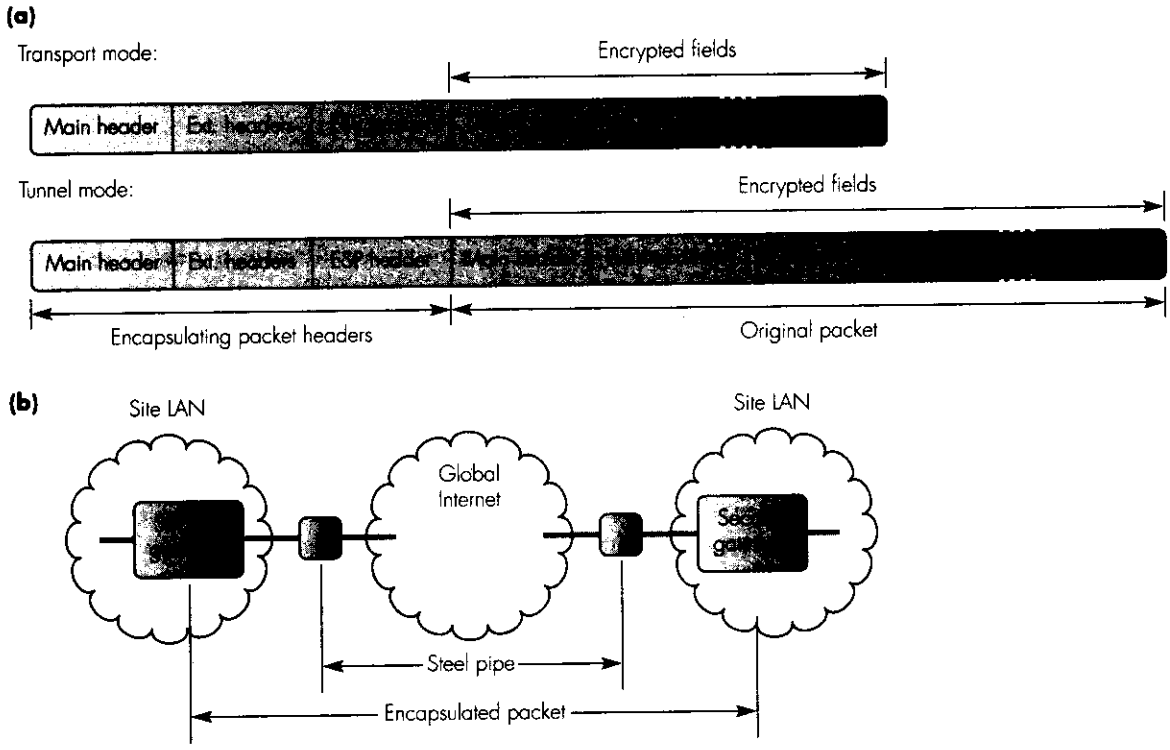


Figure 9.34 Encapsulating security payload: (a) transport and tunnel modes; (b) example application of tunnel mode.

security obtained with the tunnel mode is that the information in the main and extension headers of the original packet cannot be interpreted by a person passively monitoring the transmissions on a line. An example use of this mode is in multisite enterprise networks that use the (public) Internet to transfer packets from one site to another. The general scheme is shown in Figure 9.34(b).

As we showed earlier in Figure 5.16(a) and explained in the accompanying text, associated with each site is a security gateway through which all packet transfers to and from the site take place. Hence to ensure the header information (especially the routing header) of packets is not visible during the transfer of the packet across the Internet, the total packet is encrypted and inserted into a second packet by the IP in the security gateway with the IPv6 address of the two communicating gateways in the source and destination address fields of the main header. The path through the Internet connecting the two gateways is referred to as a **steel pipe**.

Destination options

These are used to convey information that is examined only by the destination host. As we indicated earlier, one of the ways of encoding options is to use the type-length-value format. Hence in order to ensure a header that uses this format comprises a multiple of 8 bytes, two *destination options* have been defined. These are known as Pad1 and Pad*N*, the first to insert one byte of padding and the second two or more bytes of padding. Currently these are the only two options defined.

9.10.4 Autoconfiguration

As we described earlier in Section 9.1, the allocation of the IP addresses for a new network involves a central authority to allocate a new netid – the Internet Network Information Center (InterNIC) – and a local network administrator to manage the allocation of hostids to each attached host/end system. Thus, the allocation, installation, and administration of IPv4 addresses can entail considerable effort and expenditure. To alleviate this, IPv6 supports an autoconfiguration facility that enables a host to obtain an IP address dynamically via the network and, in the case of mobile hosts, use it just for the duration of the call/session.

Two types of autoconfiguration are supported. The first involves the host communicating with a local (site) router using a simple (stateless) request-response protocol. The second involves the host communicating with a site (or enterprise) address server using an application protocol known as the **dynamic host configuration protocol (DHCP)**. The first is suitable for small networks that operate in the broadcast mode (such as an Ethernet LAN) and the second for larger networks in which the allocation of IP addresses needs to be managed.

With the first method, a simple protocol known as **neighbor discovery (ND)** is used. As we show in Figure 9.35, this involves the host broadcasting a *router solicitation* packet/message on the subnet/network and the router responding with a *router advertisement* message. Both messages are ICMPv6

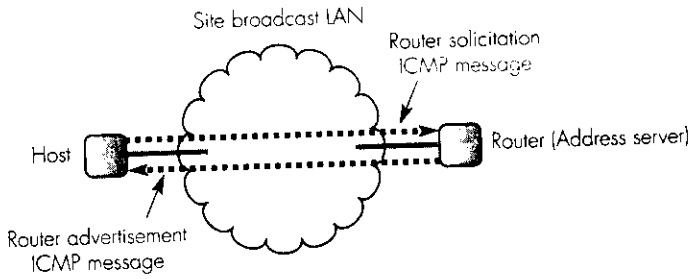


Figure 9.35 Neighbor discovery protocol messages.

messages and hence are carried in an IPv6 packet. The latter is then broadcast over the LAN in a standard frame.

The main header of the IPv6 packet containing the router solicitation message has an IPv6 source address created by the host. This is made up of the (standard) link-local address prefix and the 48-bit MAC address of the host's LAN interface. As we indicated earlier, with IPv6 a number of permanent multicast group addresses have been defined including an all-routers group address. Hence the destination address in the packet main header is set to this and, since the packet is broadcast over the LAN, it is received by the ICMP in all the routers that are attached to the LAN.

A single router is selected to process this type of ICMP message and this responds to a router solicitation message with a route advertisement message containing the Internet-wide address prefix for the subnet/network. On receipt of this, the ICMP in the host proceeds to create its own IPv6 address by adding its 48-bit MAC address to the prefix. As we can deduce from this, in relation to IPv4, this is equivalent to the router providing the netid of the site and the host using its own MAC address as the hostid. Also, the same procedure can be used by mobile hosts.

With the second method, the host requests an IPv6 address from the site address server using the DHCP. The **DHCP address server** first validates the request and then allocates an address from the managed list of addresses the server contains. Alternatively if a site does not have its own address server – for example if the site LAN is part of a larger enterprise network — then a designated router acts as a **DHCP relay agent** to forward the request to the DHCP address server.

9.11 IPv6/IPv4 interoperability

The widespread deployment of IPv4 equipment means that the introduction of IPv6 is being carried out in an incremental way. Hence a substantial amount of the ongoing standardization effort associated with IPv6 is concerned with the interoperability of the newer IPv6 equipment with existing

IPv4 equipment. Normally, when a new network/internetwork is created, it is based on the IPv6 protocol and, in the context of the existing (IPv4) Internet, it is referred to as an **IPv6 island**. It is necessary to provide a means of interworking between the two types of network at both the address level and the protocol level. In this section, we identify a number of situations where interoperability is required and describe a selection of the techniques that are used to achieve this.

9.11.1 Dual protocols

Dual stacks are already widely used in networks that use dissimilar protocol stacks; for example a site server that supports IPX on one port and IP on a second port. In a similar way, dual protocols can be used to support both IPv4 and IPv6 concurrently. An example is shown in Figure 9.36.

In this example, the site has a mix of hosts, some that use IPv4 and others IPv6. In order to be able to respond to requests originating from both types of host, the server machine has both an IPv4 and an IPv6 protocol at the network layer. The value in the version field of the datagram header is then used by the link layer protocol to pass a datagram to the appropriate IP. In this way the upper layer protocols are unaware of the type of IP being used at the network layer.

9.11.2 Dual stacks and tunneling

A common requirement is to interconnect two IPv6 islands (networks/internetworks) through an intermediate IPv4 network/internetwork. To achieve

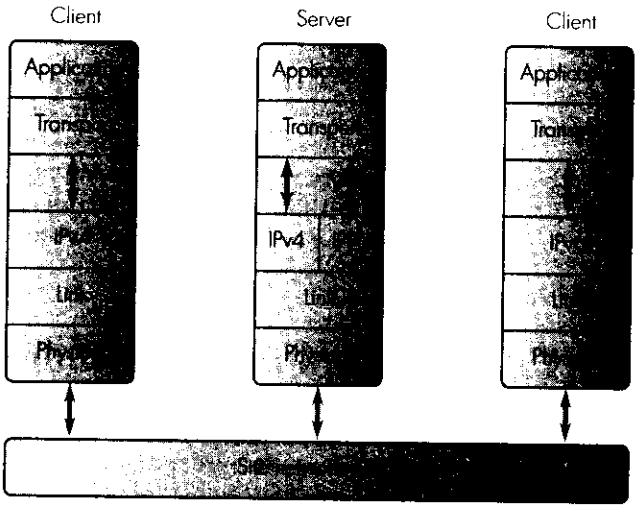


Figure 9.36 IPv6/IPv4 interoperability using dual (IPv6/IPv4) protocols.

this, the gateway/router that connects each IPv6 island to the IPv4 network must have dual stacks, one that supports IPv6 and the other IPv4. The IPv6 packets are then transferred over the IPv4 network using tunneling. The general approach is illustrated in Figure 9.37(a) and the protocols involved in Figure 9.37(b).

As we showed earlier in Figure 9.20 and explained in the accompanying text, tunneling is used to transfer a packet relating to one type of network layer protocol across a network that uses a different type of network layer protocol. Hence in the example shown in Figure 9.37, each IPv6 packet is transferred from one (IPv6/IPv4) edge gateway to the other edge gateway

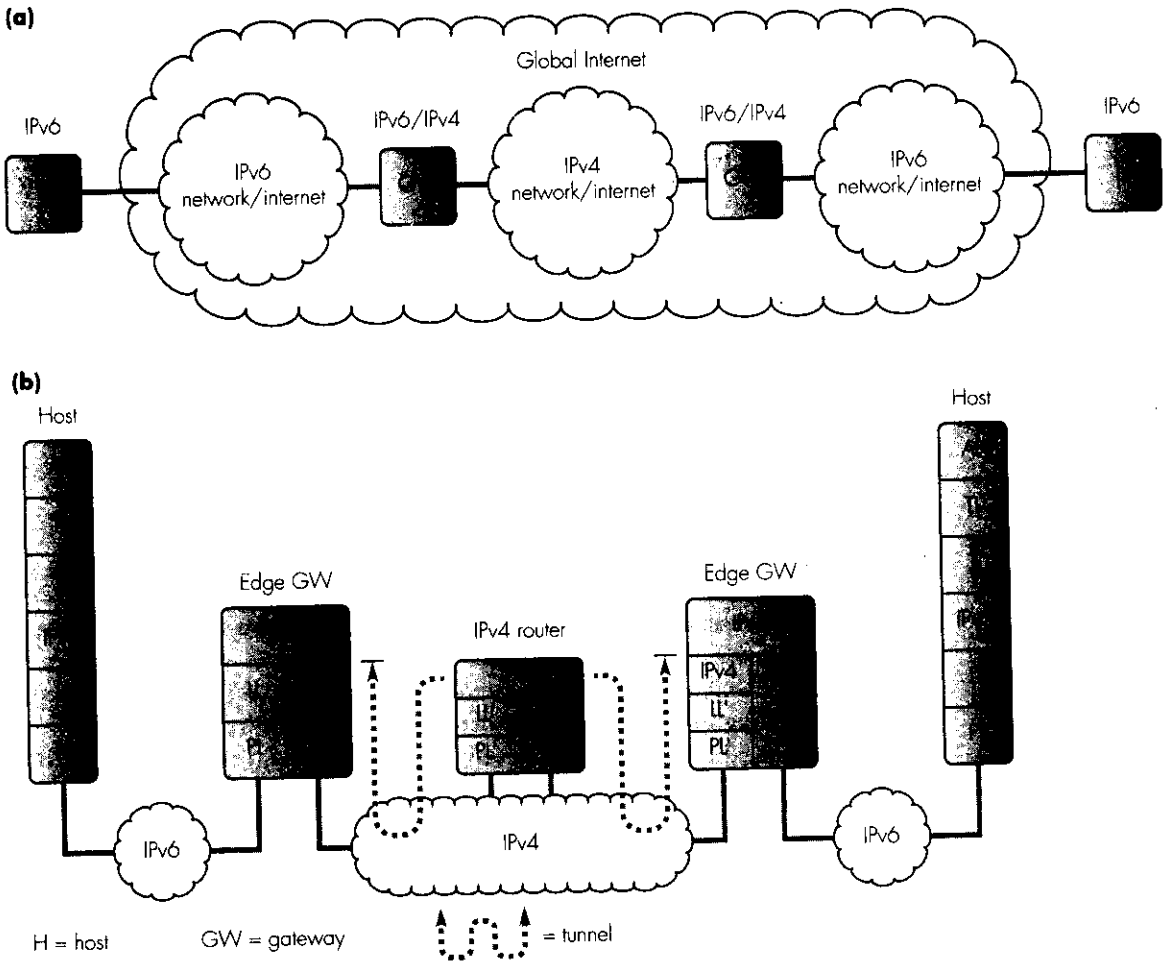


Figure 9.37 IPv6/IPv4 interoperability using dual stacks and tunneling: (a) schematic; (b) protocols.

within an IPv4 packet. As we show in the figure, in order to do this, the two edge gateways contain dual stacks each of which has a related IPv6/IPv4 address associated with it. Normally, entered by network management, the routing table entry for the remote destination IPv6 host is the IPv4 address of the remote edge gateway.

The IPv6 in each gateway, on determining from its routing table that an (IPv6) packet should be forwarded to a remote IPv6 network via an IPv4 tunnel, passes the packet to the IPv4 protocol together with the IPv4 address of the remote gateway. The IPv4 protocol first encapsulates the IPv6 packet in an IPv4 datagram/packet with the IPv4 address of the remote gateway in the destination address field. It then uses this address to obtain the (IPv4) address of the next-hop router from its own (IPv4) routing table and proceeds to forward the packet over the IPv4 network/internetwork. On receipt of the packet, the IPv4 in the remote gateway, on detecting from its own routing table that it is a tunneled packet, strips off the IPv4 header and passes the payload – containing the original IPv6 packet – to the IPv6 layer. The latter then forwards the packet to the destination host identified in the packet header in the normal way.

9.11.3 Translators

A third type of interoperability requirement is for a host attached to an IPv6 network – and hence having an IPv6 address and using the IPv6 protocol – to communicate with a host that is attached to an IPv4 network – hence having an IPv4 address and using the IPv4 protocol. In this case, both the addresses and the packet formats are different and so a translation operation must be carried out by any intermediate routers/gateways. As we show in Figure 9.38, the translations can be performed either at the network layer – part (a) – or the application layer – part (b).

Using the first approach, on receipt of an IPv6/IPv4 packet, this is converted into a semantically equivalent IPv4 /IPv6 packet. This involves a **network address translator** (**NAT**) and a **protocol translator** (**PT**). The intermediate gateway is then known as a **NAT-PT gateway**. As we explained earlier in Section 9.10.2, an IPv4 address can be embedded in an IPv6 address. Hence to send a datagram/packet from a host with an IPv6 address to a host with an IPv4 address, the NAT in the gateway can readily obtain the destination IPv4 address from the destination address in the main header of the IPv6 packet. The issue is what the source address in the IPv4 packet header should be.

A proposed solution is for the NAT to be allocated a block of hostids for the destination IPv4 network. These, together with the netid of the destination network, then form a block of unique IPv4 addresses. For each new call/session, the NAT allocates an unused IPv4 address from this block for the duration of the call/session. It then makes an entry in a table containing the IPv6 address of the V6 host and its equivalent (temporary) IPv4 address.

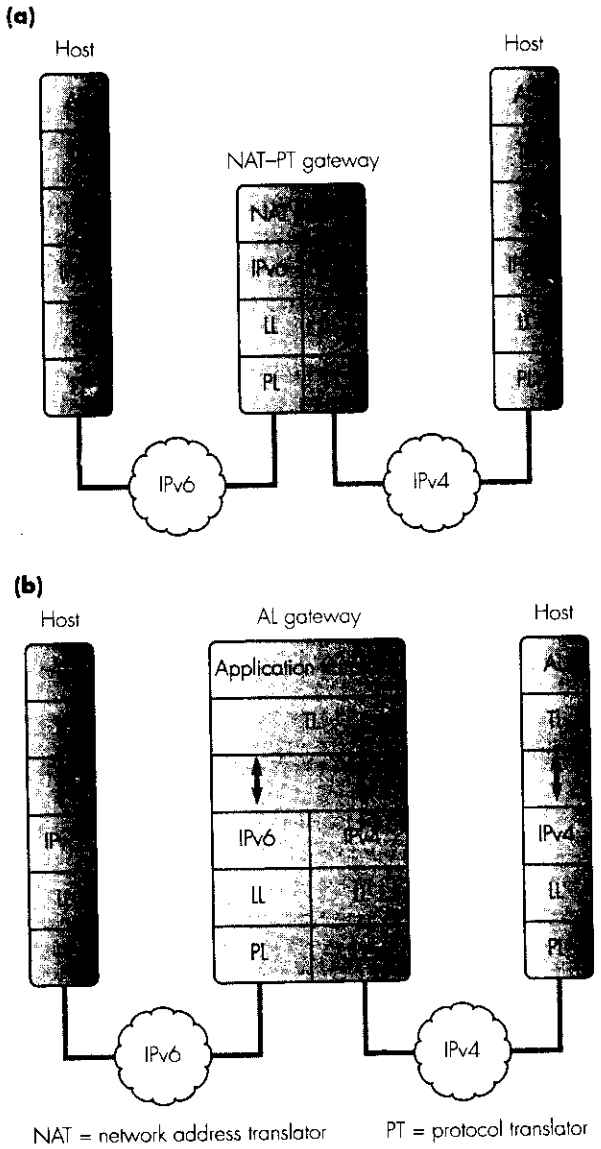


Figure 9.38 IPv6/IPv4 interoperability using translators: (a) network level; (b) application level.

The NAT translates between one address and the other as the packet is relayed. A timeout is applied to the use of such addresses and, if no packets are received within the timeout interval, the address is transferred back to the free address pool.

The protocol translation operation is concerned with translating the remaining fields in the packet header and, in the case of ICMP messages, converting ICMPv4 messages into and from ICMPv6 messages. As we indicated in Section 9.10.1, most of the fields in the IPv6 main header have the same meaning as those in the IPv4 header and hence their translation is relatively straightforward. In general, however, there is no attempt to translate the fields in the options part. Similarly, since ICMPv6 messages have different *type* fields, the main translation performed is limited to changing this field. For example, the two ICMPv4 query messages have *type* values of 8 and 0 and the corresponding ICMPv6 messages are 128 and 129.

The use of a NAT-PT gateway works providing the packet payload does not contain any network addresses. Although this is the case for most application protocols, a small number do. The FTP application protocol, for example, often has IP addresses embedded within its protocol messages. In such cases, therefore, the translation operation must be carried out at the application layer. The associated gateway is then known as an **application level gateway** (**ALG**). This requires a separate translation program for each application protocol. Normally, therefore, most translations are performed at the network layer – using a NAT and a PT – and only the translations relating to application protocols such as FTP are carried out in the application layer.

Summary

Figure 9.39 opposite summarizes the various topics discussed in this chapter.

Exercises

Section 9.1

9.1 With the aid of the network schematic shown in Figure 9.1, explain briefly the role of the following network components and protocols:
(i) access network gateway,
(ii) routing gateway,
(iii) internet protocol (IP),
(iv) datagram/packet.

9.2 With the aid of the protocol stack shown in Figure 9.2, explain briefly the meaning of the term "adjunct protocol" and how the IP in the destination host determines to which transport protocol – TCP/UDP – the contents/payload of a received IP datagram should be passed.

Section 9.2

9.3 In relation to the IP datagram/packet format shown in Figure 9.3, explain the role of the following header fields:
(i) IHL,
(ii) TOS,
(iii) Total length and Identification,
(iv) flag bits,
(v) Fragment offset,
(vi) Time-to-live,
(vii) Protocol,
(viii Header checksum,
(ix) Options.

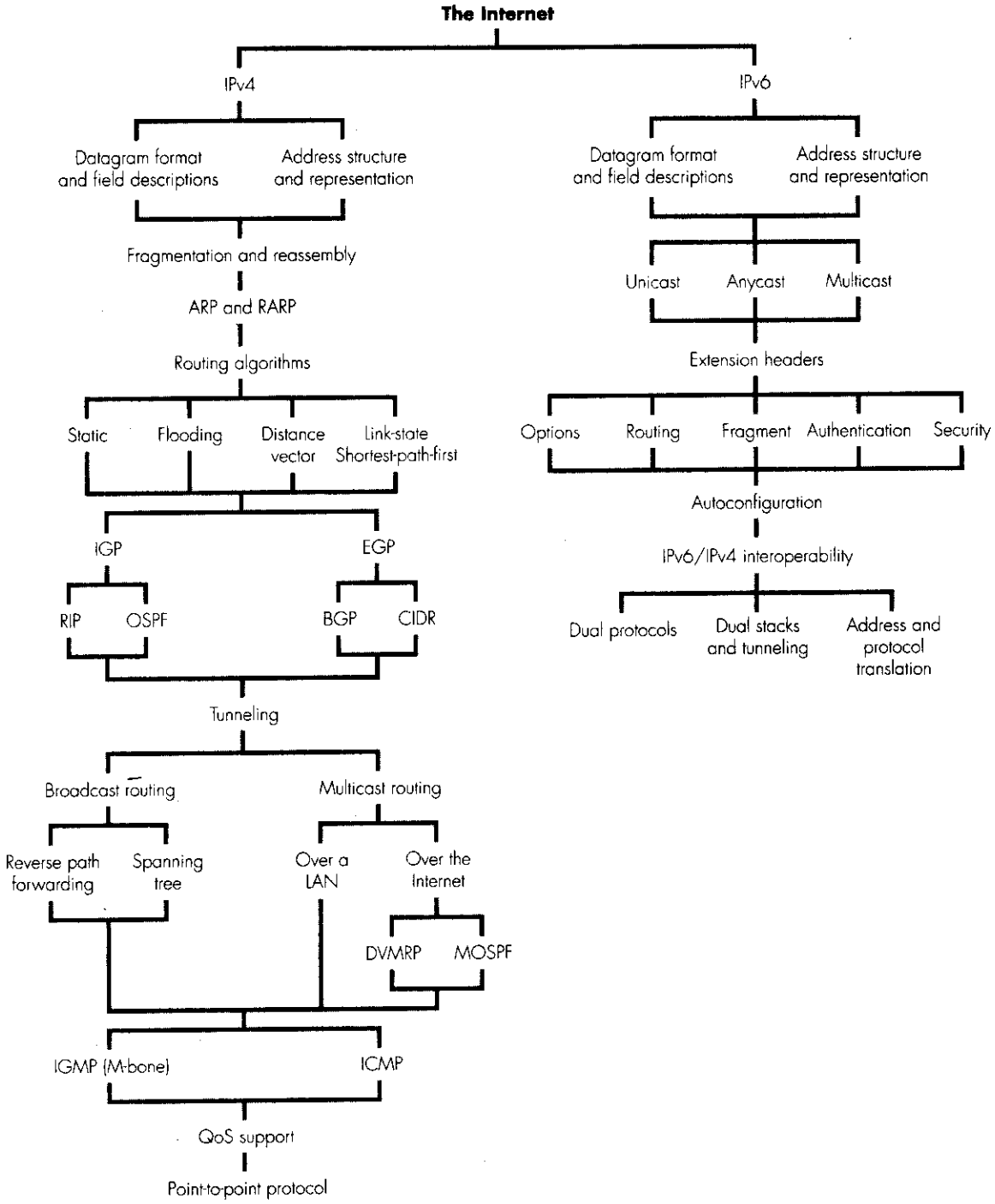


Figure 9.39 Summary of the topics discussed in relation to the Internet.

Section 9.3

9.4 Assume a message block of 7000 bytes is to be transferred from one host to another as shown in the example in Figure 9.4. In this instance, however, assume the token ring LAN has an MTU of 3000 bytes. Compute the header fields in each IP packet shown in the figure as it flows
(i) over the token ring LAN,
(ii) over the Ethernet LAN.

9.5 State why fragmentation is avoided whenever possible and the steps followed within each host IP to achieve this.

Section 9.4

9.6 Explain the meaning of the term "IP address class" and why these classes were created. Hence, with the aid of the three (unicast) address classes identified in Figure 9.5, identify a particular application for each class.

9.7 State the meaning of the following addresses:
(i) an address with a netid of all 0s
(ii) an address with a netid of all 1s
(iii) an address with a hostid of all 0s
(iv) an address of all 1s.

9.8 Explain the meaning of the term "dotted decimal". Hence derive the netid and hostid for the following IP addresses expressed in dotted decimal notation:
(i) 13.0.0.15,
(ii) 132.128.0.148,
(iii) 220.0.0.0,
(iv) 128.0.0.0,
(v) 224.0.0.1.

9.9 With the aid of an example, explain why subnetting was introduced. Hence state the meaning of a subnet router and an address mask.

9.10 A site with a netid of 127.0 uses 20 subnet routers. Suggest a suitable address mask for the site which allows for a degree of expansion in the future. Give an example of a host IP address at this site.

Section 9.5

9.11 Define the terms "IP address", "MAC address", and "hardware/physical address". Also explain the terms "address-pair" and "ARP cache".

9.12 In relation to the simple network topology shown in Figure 9.7, explain why:
(i) on receipt of an ARP request message, each host retains a copy of the IP/MAC address-pair of the source host in its ARP cache
(ii) on receipt of an ARP reply message, the ARP in the source host makes an entry of the IP/MAC address-pair in its own cache
(iii) the LAN port of the gateway keeps a copy of the IP/MAC address-pair from each ARP request and reply message that it receives.

9.13 Explain the role of a proxy ARP. Hence explain how an IP packet sent by a host at one site is routed to a host at a different site. Also explain how the reply packet is returned to the host that sent the first packet.

9.14 Explain how the reverse ARP is used to enable a diskless host to determine its own IP address from its local server.

9.15 With the aid of the two frame formats shown in Figure 9.8, explain:
(i) how the MAC sublayer in the receiver determines whether a received frame is in the Ethernet format or IEEE802.3
(ii) the number of pad bytes required with each frame type.

Section 9.6

9.16 With the aid of the global Internet architecture shown in Figure 9.9, explain the meaning/use of the following:
(i) continental backbone network,
(ii) core backbone network,
(iii) regional/national network,
(iv) access network gateway,
(v) multiprotocol gateway.