the Internet protocol

6.1 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 or a workstation that is attached to an access network. As we showed in Figure 1.1, the Internet comprises a large number of access networks that are interconnected together by means of a global internetwork. Associated with each access network – ISP network, wireless network, intranet, enterprise network, site/campus LAN, and so on – is an **access 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 lines and devices known as **switching gateways** or simply **routers**.

The Internet operates in a packet-switched mode and Figure 6.1 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. In practice, this is not always the case. Nevertheless, any end system (host) that communicates directly over the Internet 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

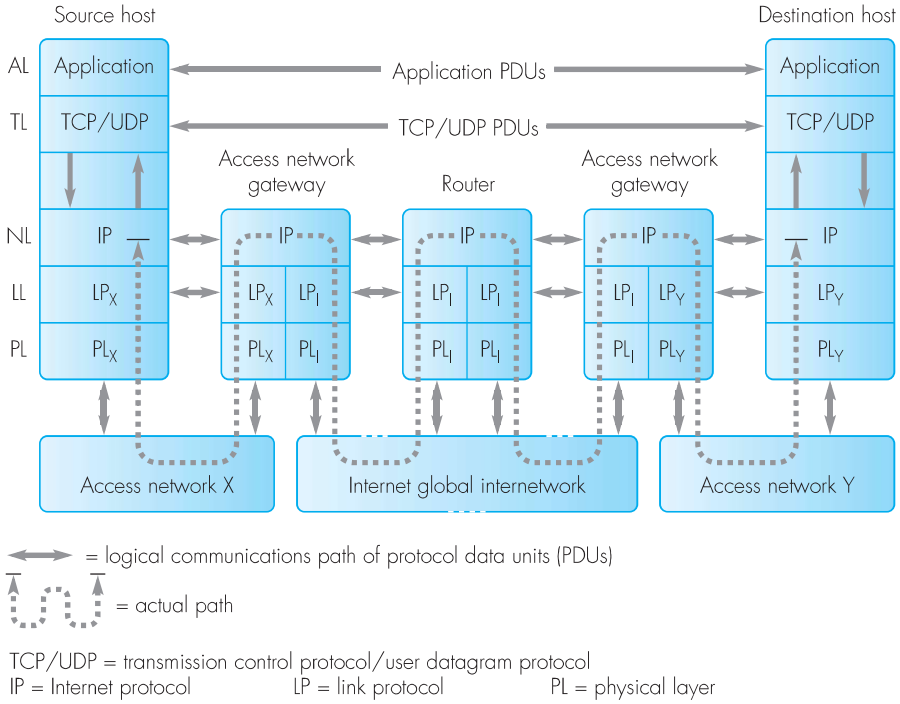


Figure 6.1 Internet networking components and protocols.

case of a site/campus LAN, as we saw in Chapters 3 and 4, a wireless 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 an access gateway and router 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 6.1, in order to transfer packets of data from one host to another, it is the IP in the two hosts, together with the IP in each access gateway and router involved, that perform the routing and other harmonization functions that are 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 number/identifier** (**netid**) and a **host number/identifier** (**hostid**). The allocation of netids is centrally managed by the **Internet Corporation for Assigned Names and Numbers** (**ICANN**) 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 TCP/UDP PDU to transfer, it simply passes the PDU 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 PDU, 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 data 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 data from each received packet to form the original submitted block of data 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 6.2 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

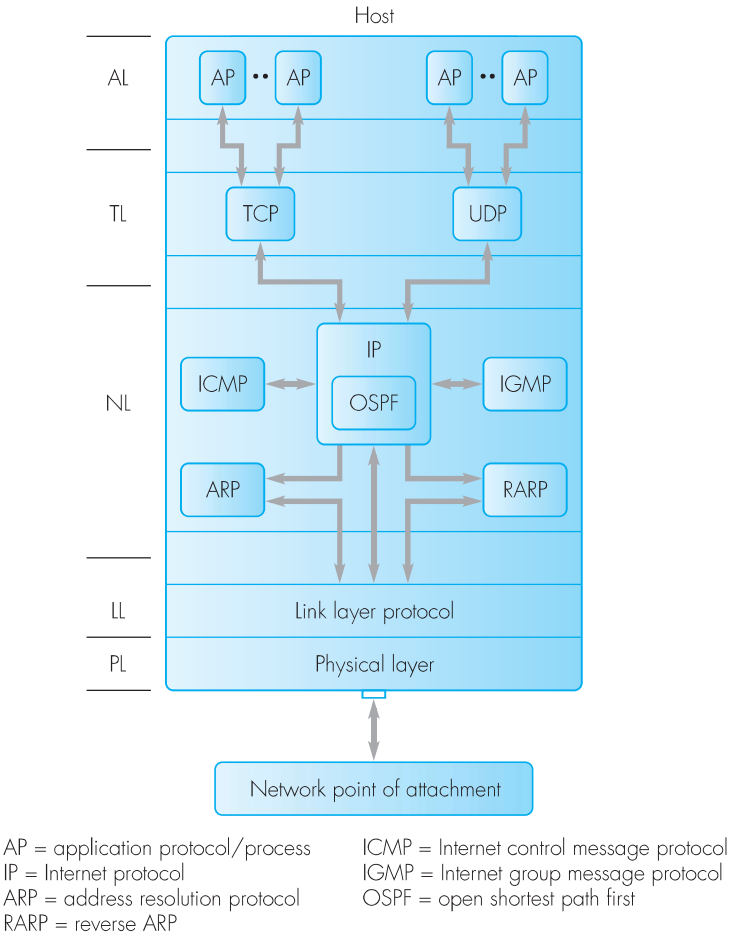


Figure 6.2 IP adjunct protocols.

Ethernet) 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.

- **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 and how it differs and interoperates with version 4. We shall defer discussion of the two transport layer protocols until the next chapter.

6.2 IP datagrams

As we indicated in the introduction, the IP is a connectionless protocol and all user data is transferred in the payload part of what is known as a datagram or packet. The header of each datagram contains a number of fields, the formats of which are shown in Figure 6.3.

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 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 6.7.

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

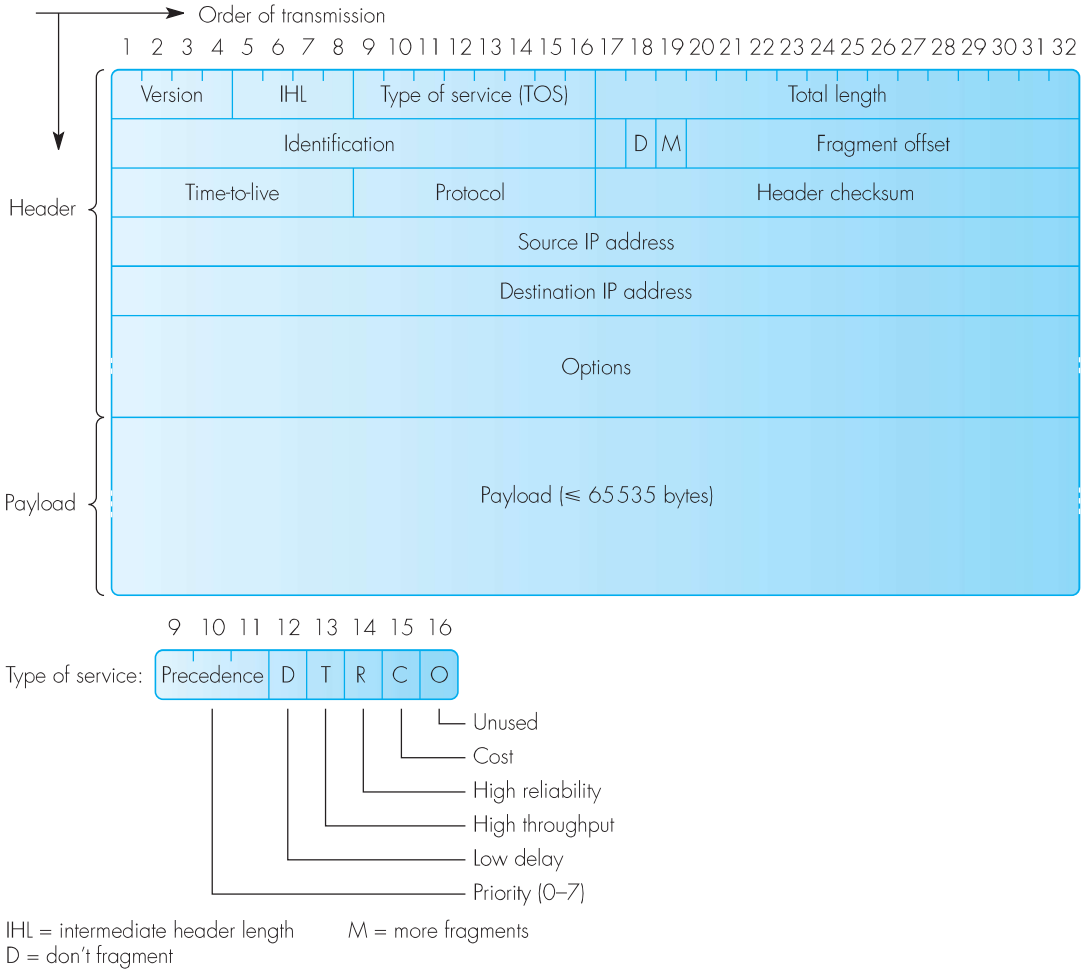


Figure 6.3 IP datagram/packet format and header fields.

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 6.6 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 6.2, 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 show in the example in Figure B.2(b) in Appendix B, the checksum is then the complement (inverse) of the 1s complement sum.

The *source address* and *destination address* are the Internet-wide 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.

6.3 Fragmentation and reassembly

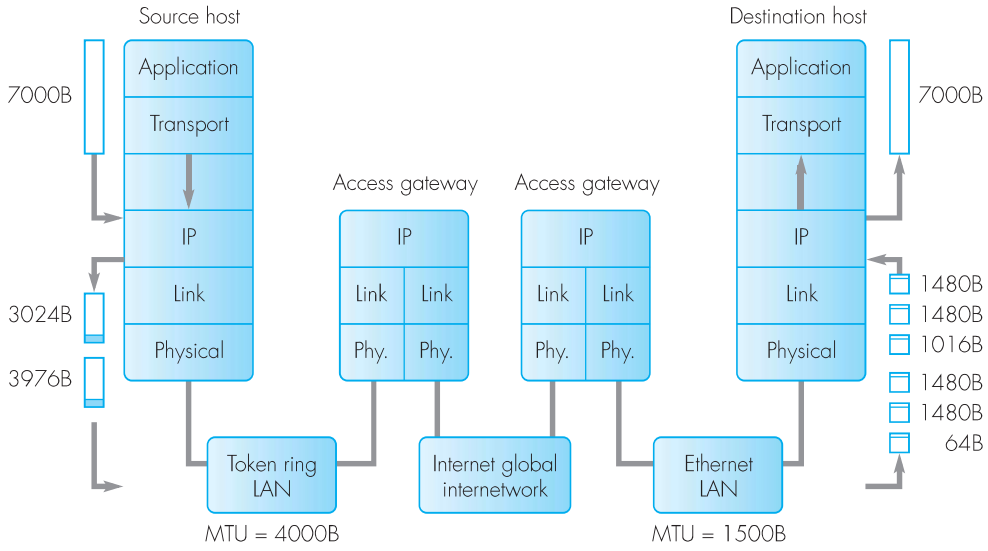
As we explained in Section 6.1, if the size of a packet is greater than the MTU of the destination access network – or an intermediate network in the global internetwork – the IP in the destination gateway – or intermediate router – divides the data received in the packet into a number of smaller blocks 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 data from each received packet to form the original submitted block of data. 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 legacy LAN – such as a token ring – 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, and 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 6.4(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 6.4(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 6.4 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 6.4(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 7.3.2, 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 6.4, 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 6.7, 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.

6.4 IP addresses

Each host, gateway and router has a unique Internet-wide IP address assigned to it that comprises a netid and hostid part. In the case of a host/computer, the netid identifies the network to which it is attached and the hostid then identifies the host on this network. In the case of an access gateway and router, however, each network interface of the gateway/router has a different netid assigned to it.

Over the lifetime of the Internet five different schemes have been used for assigning IP addresses. In general, as the number of Internet users has expanded, the aim of each scheme has been to utilize the 32-bit address space in a more efficient way. A brief summary of each scheme is helpful in order to follow their development.

- **Class-based addresses**: this was the first scheme that was used to assign addresses in the early Internet. It involved dividing the overall address space into five address classes: A, B, C, D and E. Each of the first three address classes – A, B and C – then has a defined boundary between the netid and hostid part of the address space. As we shall see, Class D is used for multicasting and class E is reserved.
- **Subnetting**: this was the first approach to utilizing the address space in a more efficient way. In the case of a large campus or company, for example, in the past it was not uncommon to have a number of different types of LAN at the same site, each of which had a different frame format and, more importantly, maximum frame length. Some examples of legacy LANs are token ring, token bus, FDDI and others. As a result, since MAC bridges do not support fragmentation, they can only be used to interconnect LANs of the same type. Hence IP routers were often used instead. As we saw earlier, with a router each LAN has to have its own netid so increasing the number of IP addresses required at each site. To overcome this, subnetting was introduced and, as we shall see, with subnetting only a single IP address is required at each site.
- **Classless addresses**: this is a more recent development and involves exploiting the full address space in a more efficient way than the use of class-based addresses. With this type of addressing, the network part of an IP address can be any number bits rather than being constrained to the fixed class boundaries and, as a result, this leads to a more efficient use of the total address space.
- **Network address translation**: this is the most recent development. The aim of the NAT scheme is for each access network to be allocated just a single IP address and this is then used by all the hosts when communicating outside of their local access network. For communications within the access network, however, every host is assigned its own (private) address.
- **IPv6**: this is a completely new version of IP – the current version is IPv4 – and was developed to overcome the limited address space – and other limitations – of IPv4. Hence it is the ideal solution. In practice, however, at this point in time, IPv4 is still the dominant protocol in the current Internet and the use of IPv6 is being introduced in an incremental way.

In the following subsections, we shall study the first four schemes in more detail. In the case of IPv6, however, we shall study this in a separate section since it is clearly the scheme of the future.

6.4.1 Class-based addresses

In order to have some flexibility when assigning netids, the original class-based addressing scheme – also known as **classful addressing** – was to divide the 32-bit address space into five different address formats. Each format is called an **address class** and, as we show in Figure 6.5, classes A, B and C are used for unicast; that is, for communications between a pair of host interfaces.

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.

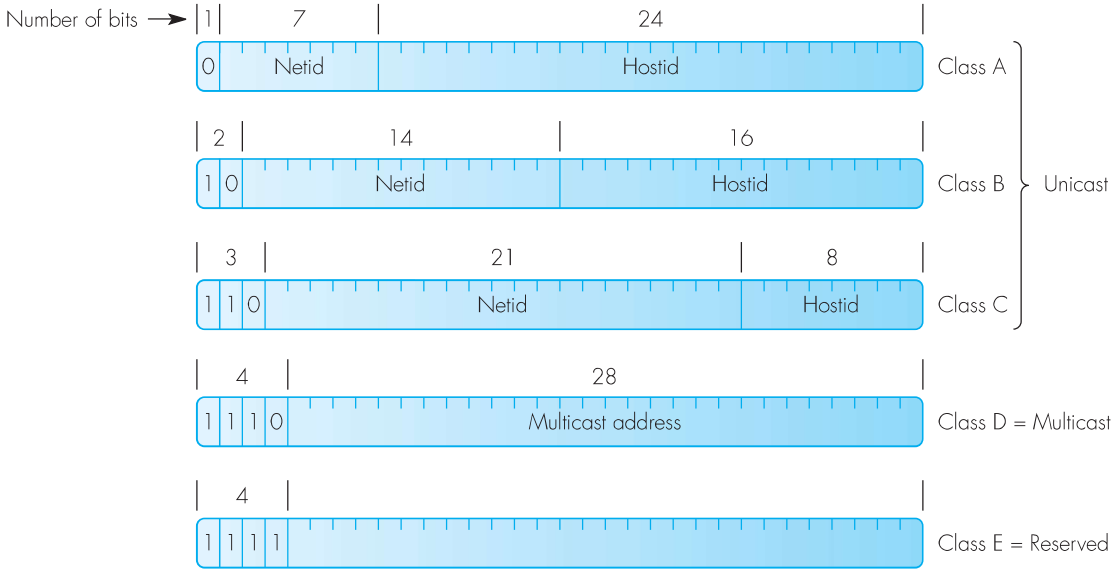


Figure 6.5 IP address formats.

- An address with a netid of all 0s implies the same network as the source network/netid.
- 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 6.1

Assuming the IP address formats shown in Figure 6.5, 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
= 126 networks
Hostid = 0.0.0 to 255.255.255
= 16 777 214 hosts

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

Class C:
Netid = 192.0.0 to 223.255.255
= 2 097 152 networks
Hostid = 0 to 255
= 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 3.2, 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 6.6.9, multicast group addresses are assigned by ICANN. 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.

6.4.2 Subnetting

As we described in Section 3.3.2, 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 earlier, 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 class-based 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 6.5, 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 6.6(a). It is defined in RFC 950.

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 Internet-wide routing, the netid is considered as the **Internet part.** For a single netid with a 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,

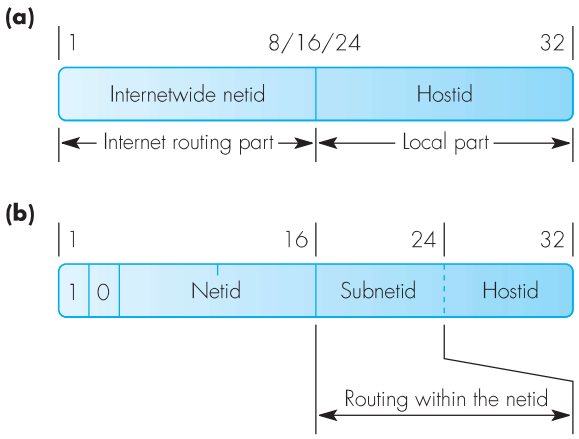


Figure 6.6 Subnet addressing: (a) address structure; (b) example.

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 Internet-wide netid, the next byte the subnetid, and the last byte the hostid on the subnet. Such an address is shown in Figure 6.6(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 6.2

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 a Fast Ethernet switch using a subnet router, define a suitable address mask for the site if the maximum number of hosts connected to each subnet is 70.

Answer:

A class B IP address means that both the netid part 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 subnetid and 8 bits for the hostid.

This will allow for up to 254 subnets and 254 hosts per subnet ignoring all 1s and all 0s.

The address mask, therefore, is 255.255.255

6.4.3 Classless addresses

This scheme was introduced in the mid-1990s and is defined in RFC 1519. With classless addresses, the network part of an IP address can be any number of bits rather than being constrained to the fixed class boundaries. A classless address is represented in dotted decimal form as w.x.y.z / n where n indicates the number of bits in the network part of the address.

For example, if an organization requests a network address with a block of, say, 1000 host addresses, then this could be allocated a block of 1024 addresses. The dotted decimal representation of this would then be w.x.y.z / 22, which indicates that the leading 22 bits of the address w.x.y.z represent the netid and the last 10 bits the hostid. Note that subnetting can still be used on the hostid part of the address. In addition, although this scheme leads to a more efficient use of the address space, the routing of packets is more complicated. The routing method is called **classless inter-domain routing** (**CIDR**) and is defined in RFC 1519.

The approach adopted with CIDR is similar to that we described in the previous section 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 an address mask. In a similar way, as we show in Example 6.3, an address mask is used to indicate the boundary between the netid and hostid parts of the complete IP address.

Example 6.3

A network within a large network has been allocated a block of 1024 addresses from 200.30.0.0 through to 200.30.3.255. Assuming the CIDR addressing scheme, represent these addresses in binary form and hence derive the address mask to be used in dotted decimal form and the netid of this network.

Answer:

Address 200.30.0.0 = 11001000 00011110 00000000 00000000

Address 200.30.3.255 = 11001000 00011110 00000011 11111111

Hence address mask = 11111111 11111111 11111100 00000000

= 255 . 255 . 252 . 0

and netid = 200.30.0.0

Each router within a large network then contains a copy of the address mask of each of the networks that make up the larger network together with the base address – netid – of the corresponding network. In this way, a router, on receiving a packet, reads the destination IP address from the packet header and then performs a logical AND operation on this and each of the address masks that it is holding. On detecting a match – that is, the resulting netid is the same as that stored with the corresponding mask – the router uses the netid and the related routing protocol to route the packet to the next router along the path to the destination network.

As we can deduce from this, each router must also contain a copy of the address masks of all the networks that make up the larger network. Also, each access gateway has a copy of its own address mask and, by using this, it first extracts the netid from the destination IP address and then uses it to route the packet to that host.

Finally, as we can see in Example 6.4, it is possible for a number of hosts associated with a network that 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 masks are tested for a match, this will be in addition to the match relating to the mask with the smaller block of addresses. When this happens, 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 6.4

Two networks within a larger network have been allocated the following block of addresses:

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 of 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

6.4.4 Network address translation

In the case of a local ISP with a large number of subscribers using low bit rate modems that are located in homes and in small businesses, normally the ISP is allocated a block of IP addresses that is significantly less than the number of subscribers it serves. Hence to overcome this, each IP address is allocated on-demand for the duration of a session and, on completion, it is then reused. In this way only a relatively small number of IP addresses are required. However, with the arrival of broadband modems, since these are permanently on, a dedicated IP address is required for each subscriber, or worse, for each member of a household/business. The same is true for each campus, company and wireless LAN.

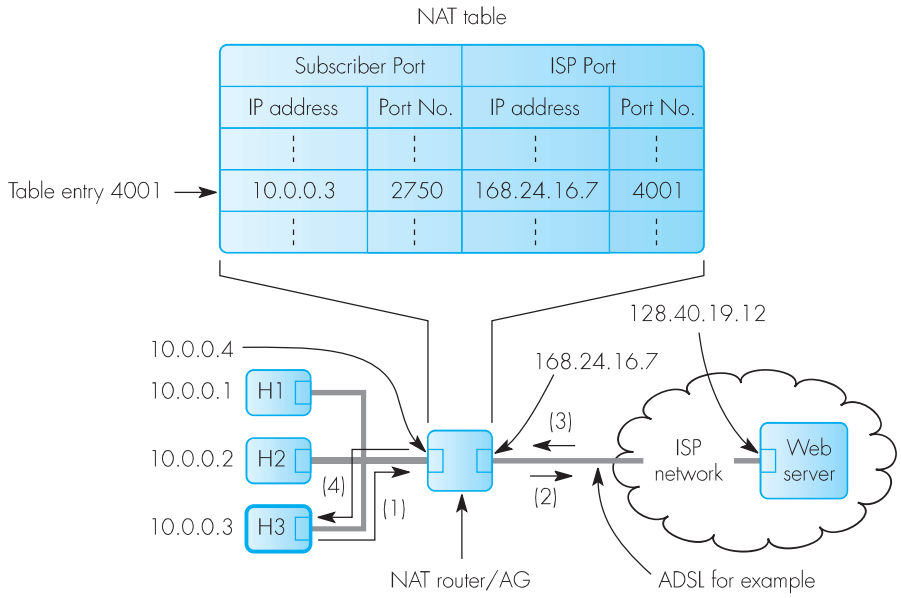
The aim of the network address translation (NAT) scheme is for each access network to be allocated just a single IP address or, for a large site, a small number of addresses. The allocated address is then used by all the hosts for communicating outside their local access network. For communications within the access network, however, every host (interface) is allocated its own (private) IP address. NAT is defined in RFC 3022.

To implement the NAT scheme, three blocks of IP addresses have been declared as private; that is, a household/company, etc., can allocate them to PCs/servers within their own network as they like but they must not be used outside the network and hence within the Internet. The three blocks of addresses are shown in Figure 6.7(a).

(a)

10.0.0.0 –	10.255.255.255/8 =	16,777,216	Host interfaces
172.16.0.0 –	172.31.255.255/12 =	1,048,576	
192.168.0.0 –	192.168.255.255/16 =	65,536	

(b)



(c)

	Source IP address	Destination IP address	Source port	Destination port
(1)	10.0.0.3	128.40.19.12	2750	80
(2)	168.24.16.7	128.40.19.12	4001	80
(3)	128.40.19.12	168.24.16.7	80	4001
(4)	128.40.19.12	10.0.0.3	80	2750

(1)–(4) = IP packets exchanged

Figure 6.7 Network address translation: (a) blocks of private addresses; (b) NAT operation schematic; (c) IP/TCP header fields.

To explain how the scheme operates, consider the simple home/small business network shown in Figure 6.7(b). Typically, the three host devices are PCs and each wants to use the services of the Internet such as Web access. As we can see in the figure, the three hosts have been allocated a private IP address of 10.0.0.1/2/3 respectively. As we saw earlier in Section 1.5.2, when an Internet application – a Web browser for example – wants to communicate

with, say, a remote Web server, the application uses the services of the TCP layer. As we will expand upon in the next chapter, within the header of each TCP protocol data unit (PDU) is a *source port number* that identifies the application making the request and a *destination port number* that identifies the correspondent application in the remote computer. For a Web server, for example, this is one of the reserved – also called well-known – port numbers and is 80. We assume also that the source port number selected by the PC is 2750 and that the IP address of the Web server on the Internet is 168.24.16.7. We shall explain how an IP address is obtained from a domain name later in Chapter 8 when we study Internet applications in more detail.

Associated with the site NAT router/access gateway is a **NAT table** and, as we can see in the figure, for each session this contains two entries. On the subscriber port side the entry comprises the private IP address of the host interface and the allocated source port number. On the ISP port side the entry is composed of the Internet IP address of the site and a new TCP source port number allocated by the NAT router. This is done to avoid the possibility of the same TCP source port number being selected by a different host. The sequence of datagrams/packets that are exchanged is shown in the figure together with the contents of the source and destination IP addresses and TCP port numbers in the respective header fields in part (c).

First host H3 creates an IP datagram and sends it to the NAT router (1). On receipt of this, the NAT router reads the source IP address and source port number from the datagram – 10.0.0.3 and 2750 respectively. The NAT router then proceeds to replace these two fields with the site IP address and the new source port number. It then makes two entries in the NAT table using the new source port number – 4001 in the example - as an index to the NAT table. Next, since the IP and TCP checksums in the respective headers are now invalid, these are recomputed and their respective fields updated. The IP datagram is then forwarded to the ISP access network (2).

As we show in part (c), when the response from the Web server is created, the two pairs of address fields in the header are simply exchanged. Hence, when the response datagram from the Web server is received (3), the NAT router reads the source port number from the datagram and uses this as an index to the NAT table. It then proceeds to read the original IP address and source port number from the table entry and writes these into the corresponding fields in the packet header. The IP and TCP checksums are then updated as before and the datagram is then relayed out to the interface of host 3.

In conclusion, before leaving this subsection it should be said that NAT is viewed as a temporary fix to avoid running out of IPv4 addresses and that the introduction of IPv6 will accelerate as the new set of IP addresses created by NAT are used up.

6.5 Routing algorithms

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 6.8.

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 local ISP or a site/campus LAN with a single netid.

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 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

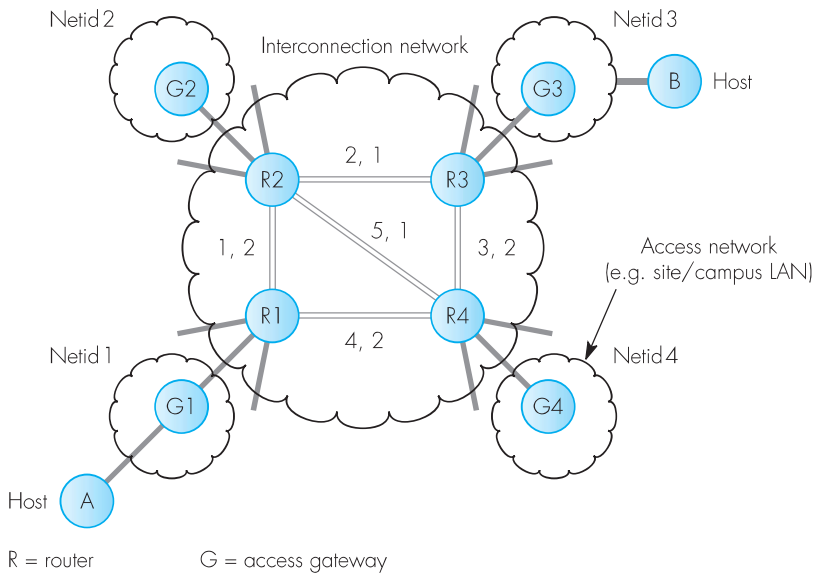


Figure 6.8 Example internetwork topology.

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. 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. What we do not know is how the datagram is routed across the interconnection network.

There are three unanswered questions involved:

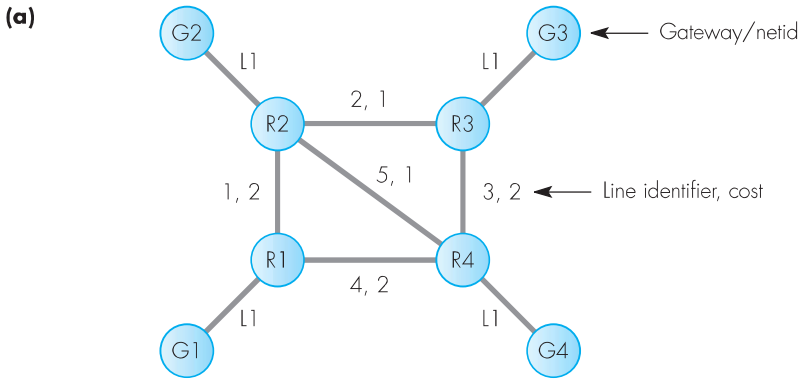
(1) How does R1 know from the netid contained within the destination IP address that the destination router is R3?
(2) How does R1 know the shortest path route to be followed through the interconnection network to R3?
(3) How does R3 know how to relay the datagram to G3 instead of one of the other gateways 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 of 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.

6.5.1 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 6.9. To avoid unnecessary repetition, we assume only one gateway/netid is attached to each router.

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.



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

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.

6.5.2 Flooding

To explain the operation of the flooding algorithm, we return to Figure 6.8 and again assume we are sending a datagram from host A to host B. The steps followed are summarized in Figure 6.10.

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 that it is addressed to one of its own netids, R3 forwards the packet to G3. Additional copies of the packet will then be received

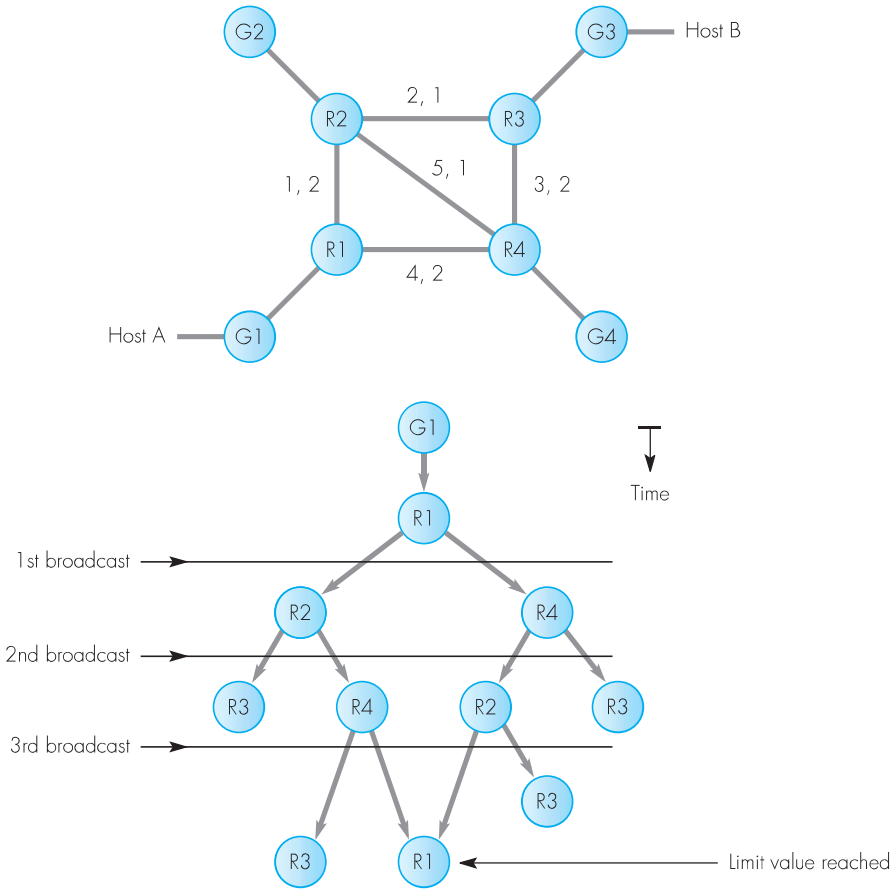


Figure 6.10 Flooding example.

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 mean that it is used primarily during the initialization phase that enables each router to determine the topology of a network.

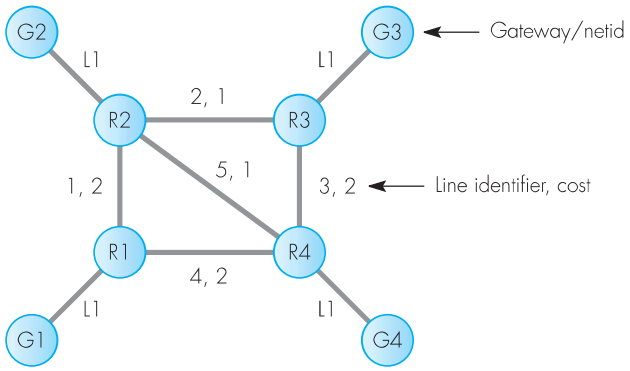
6.5.3 Distance vector routing

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 entered either 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 6.11(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 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.

(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)

R1: (R2, R4)

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

R2: (R1, R3, R4)

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

R3: (R2, R4)

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

R4: (R1, R2, R3)

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

R1: (R2, R4)

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

R2: (R1, R3, R4)

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

R3: (R2, R4)

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

R4: (R1, R2, R3)

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

Figure 6.11 Distance vector algorithm: (a) internet topology and initial tables; (b) derivation of final routing tables.

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 6.11(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, and 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.

6.5.4 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 6.12(a). The link-state algorithm is then run and the build-up of the internet topology by R1 is shown in Figure 6.12(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 6.13(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 6.13(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 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.

(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

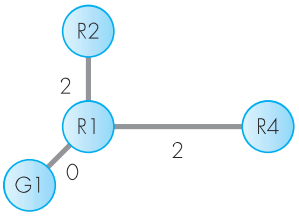
R4:

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

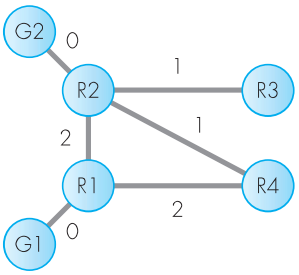
(b)

Topology build-up by R1:

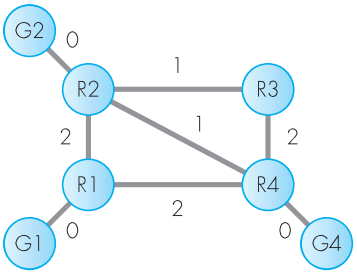
(i) Initial:



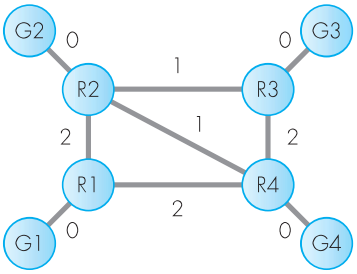
(ii) After connectivity information from R2:



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

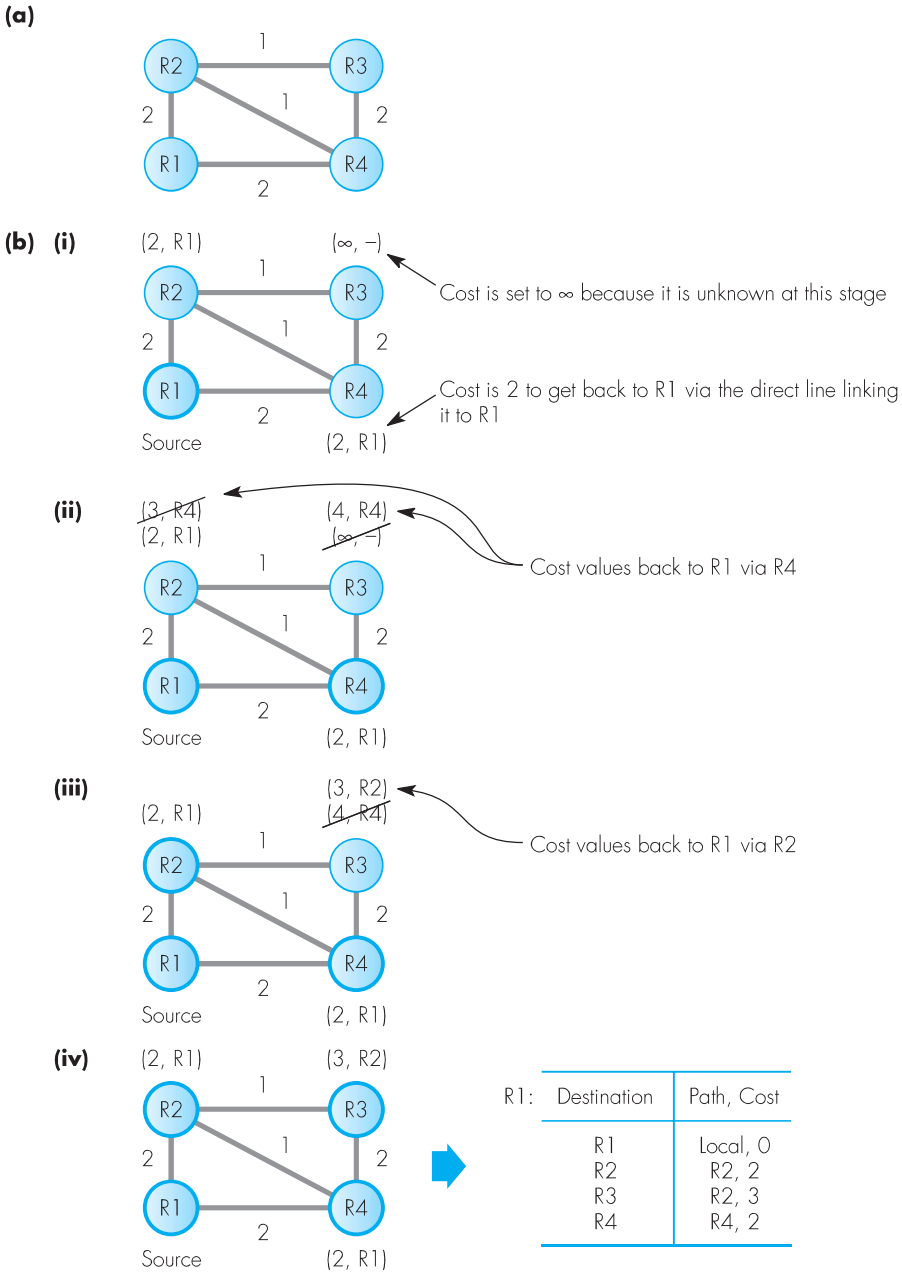


Figure 6.13 Dijkstra algorithm: (a) initial topology; (b) derivation of shortest paths from R1 to each other router.

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 6.14 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, 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 6.15(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

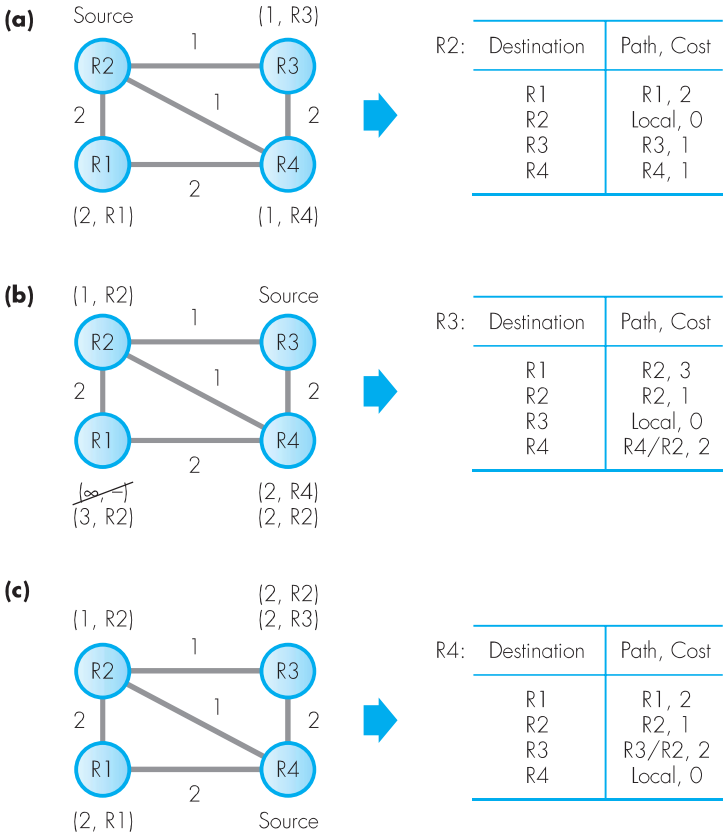


Figure 6.14 Shortest path derivations: (a) by R2; (b) by R3; (c) by R4.

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, 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 6.15(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

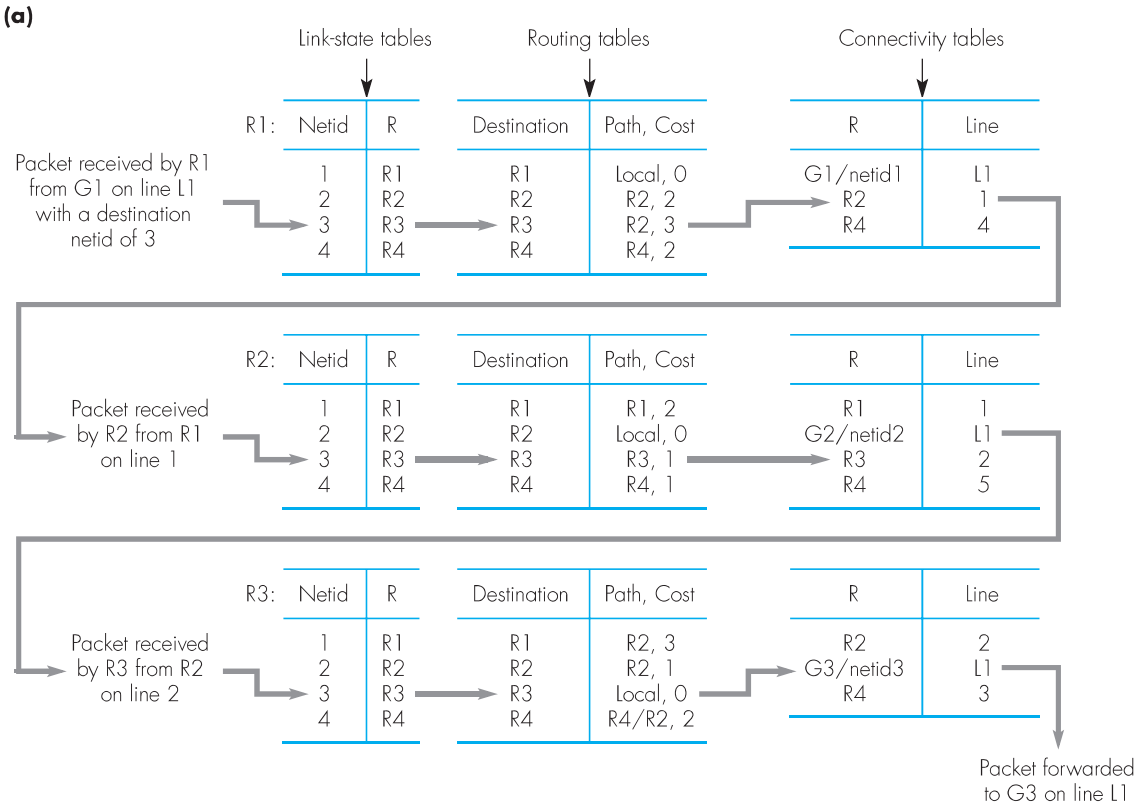


Figure 6.15 LS-SPF routing examples: (a) hop-by-hop routing; (b) source routing.

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 that 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 6.5.3, 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 6.2, this forms an integral part of the IP.

6.5.5 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 indicated earlier, the Internet is made up of many separately managed networks and internetworks that, 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 for security reasons 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 6.16 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 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.

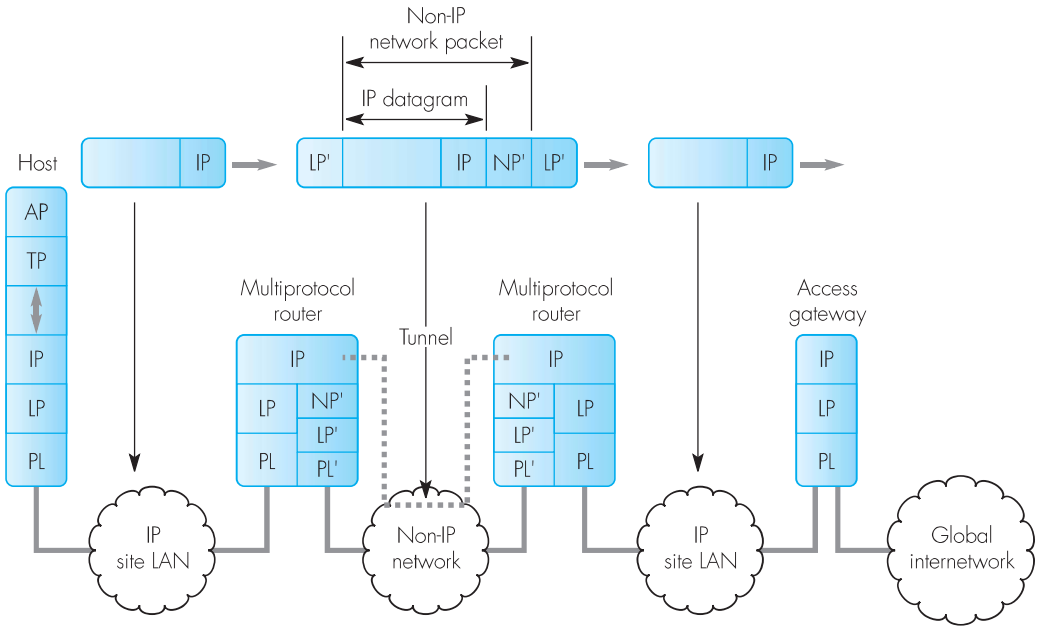


Figure 6.16 Tunneling example.

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 6.6.8, 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.

6.5.6 Broadcast routing

As we explained in Chapter 3, LANs such as Ethernet 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 6.4, 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 and any internet 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 and any internet 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 6.5.2, this has very high bandwidth overheads associated with it. Two alternative approaches are employed, the choice being 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 6.17(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 6.5.3, 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 6.17(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 6.17(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, and 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.

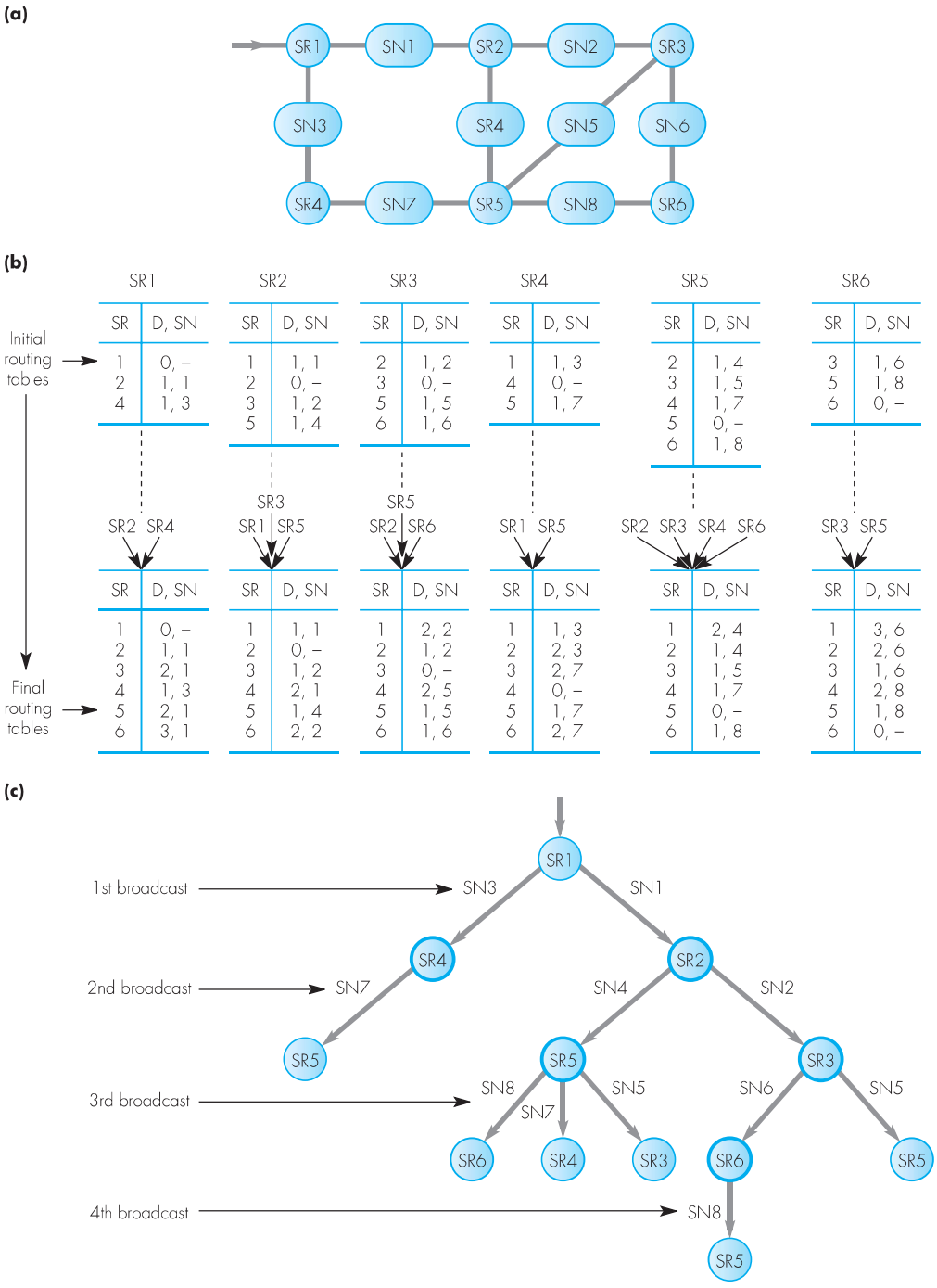


Figure 6.17 Reverse path forwarding: (a) network topology; (b) distance vector routing tables using a hop-count metric; (c) broadcast sequence.

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 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. Likewise, the packet received by SR5 from SR6 after the fourth broadcast is also discarded.

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, and 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 6.5.4, 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. A spanning tree topology is established in order to avoid frames looping within the total network. This is done by defining the ports associated with each subnet router 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 6.17(a) and, assuming each subnet router knows the root SR, each will derive the spanning tree shown in Figure 6.18(b). The resulting broadcast sequence is therefore as shown in Figure 6.18(c).

We can make a number of observations from this example:

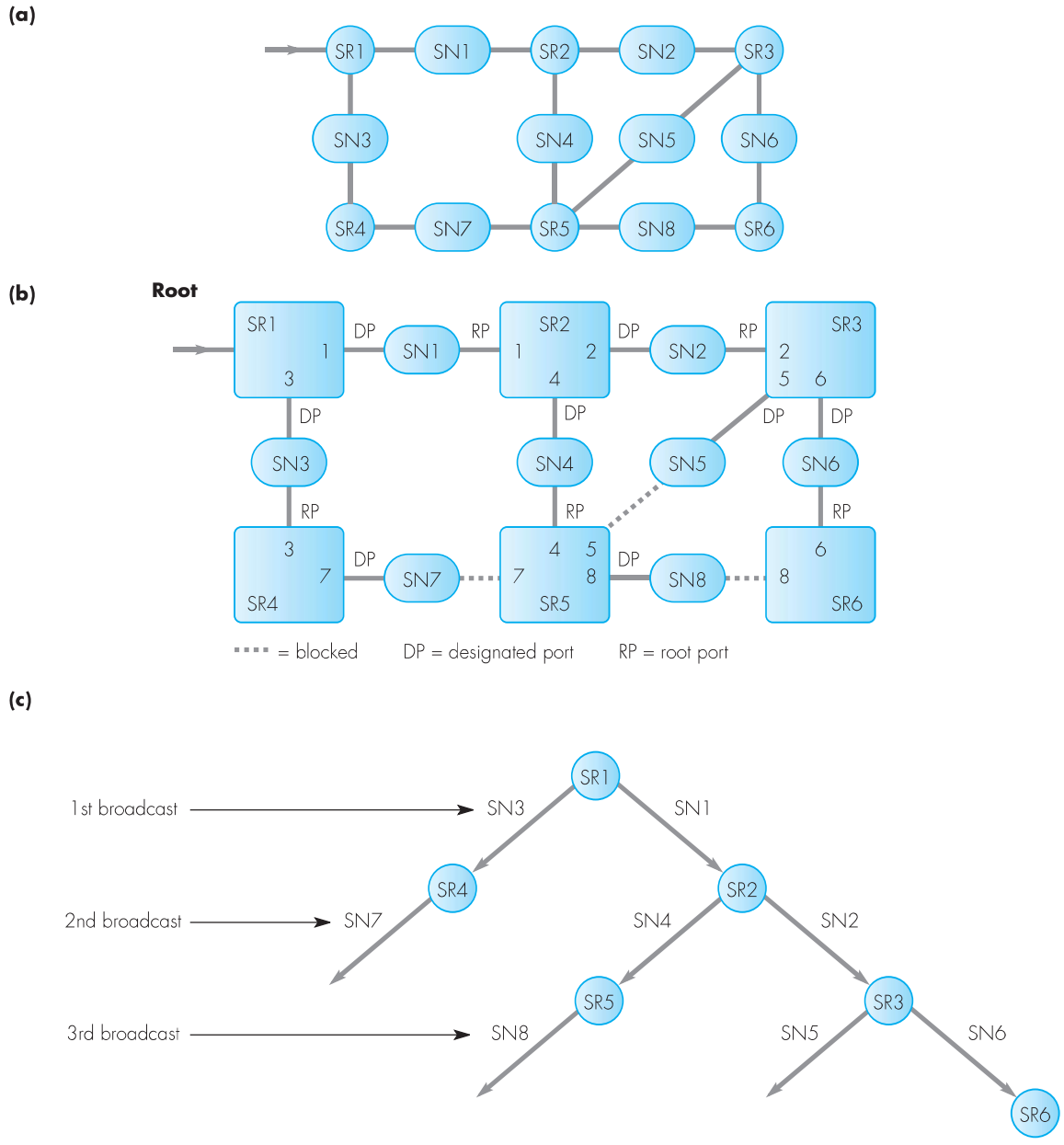


Figure 6.18 Spanning tree broadcast: (a) network topology; (b) spanning tree derived by each subnet router; (c) broadcast sequence.

- 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.
- 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.

6.6 Routing in the Internet

In the previous two sections we have built up an understanding of IP addresses and the different types of routing algorithms that are available. We can now build on this knowledge to explain how routing is carried out in the Internet. We shall start with a description of the current structure of the Internet and then proceed to use this to describe the specific routing algorithms that are used.

6.6.1 Internet structure and terminology

As we explained earlier in Section 2.6.5, the Internet is composed of many thousands of access networks that are geographically distributed around the world. As a result, the global internetwork that is used to interconnect the access gateways associated with these networks is not a single network but a collection of different types of regional, national and international networks that have evolved over the lifetime of the Internet. The general structure of the Internet is shown in Figure 6.19 and, as we can deduce from this, the Internet is often described as a **network-of-networks**.

At the lowest level in the hierarchy – known as **Tier 3** – are the various types of access networks – site/campus LANs, local ISP networks, cable networks, wireless LANs, etc. Associated with each access network is an access gateway and this is connected to the nearest gateway of a regional ISP network or, in some instances, a legacy regional network/internetwork, for a small country, a national network – **Tier 2**.

At the highest level in the hierarchy, each country has its own ISP/legacy backbone networks – **Tier 1**. These are composed of a geographically distributed

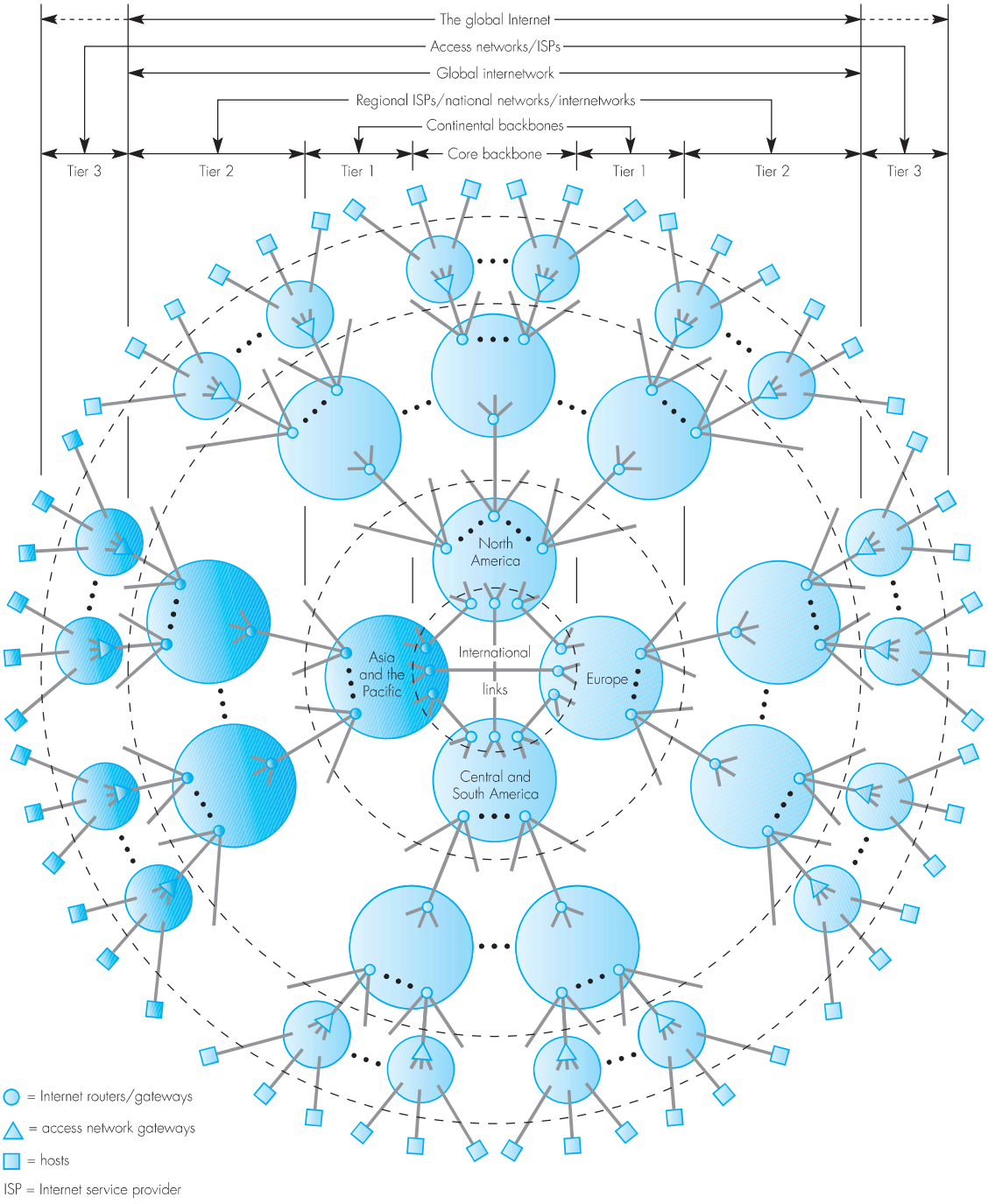


Figure 6.19 General architecture of the global Internet.

set of very high throughput routers that are interconnected using very high bit rate optical fiber lines leased from one or more telecommunication providers. The country backbones are then interconnected together by means of a smaller number of either very high throughput leased lines or devices known as **network access points** (**NAPs**).

In addition, as we indicated earlier in Section 2.6.5, because the networks at the different layers are operated by a number of different companies, private agreements are reached between them – normally on cost grounds – to provide direct links between selected routers in the same peer networks. This practice is called **private peering**.

Clearly, it is not possible for every router in the hierarchy to maintain routing information relating to every other router. Hence in order to make the routing of datagrams/packets around the global Internet manageable, the overall structure is broken down into a hierarchical structure involving a large number of what are called **autonomous systems** (**ASs**). The ASs are then interconnected by an intercontinental backbone network. As we can deduce from this, there are a large number of ASs in the Internet. Each Tier 3 and Tier 2 network within an AS is called an **area** and one of these networks is selected to be the backbone area of this AS. Every AS is then connected to a higher-level backbone area.

The routing algorithm used within an AS is called an **interior gateway protocol** – or sometimes an **intra-AS routing protocol** – and, in principle, each AS can use its own routing protocol. In practice, there are just two protocols used. The first is based on the distance vector algorithm and is called the **routing information protocol** (**RIP**). The second is based on the link-state shortest path first algorithm and is called the **open shortest path first** (**OSPF**) protocol. The original interior gateway protocol was RIP and there are still many ASs in the Internet that use this. However, the preferred protocol is now OSPF as this has been found to be more robust than RIP when erroneous routing updates occur. As a result, OSPF is now supported by all the major router manufacturers. The latest version is version 2 and this is defined in RFC 2328.

The routing protocol used for communications between ASs is called the **border gateway protocol** (**BGP**). The latest version of this is version 4 – BGP-4 – and is specified in RFCs 1771/2/3/4. It is called a **path vector protocol** in the standard since neighbor BGP routers that have a direct link between them – also known as **BGP peers** – exchange path information rather than link cost values. Typically, the path information is a list of the ASs that lie on a path to a particular destination AS. The total routing table for an AS is then built up by each AS exchanging routing updates with its directly connected neighbors.

In this section we shall describe the operation of both the OSPF and BGP routing protocols. Before we do this, however, we shall first describe the operation of two of the protocols that are used widely in access networks. These are ARP/RARP and DHCP.

6.6.2 ARP and RARP

As we outlined in Section 6.1 and illustrated in Figure 6.2, 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.

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 6.20.

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 access gateway (AG). We assume that all the hubs are simple repeater hubs and that all hosts have just been switched on

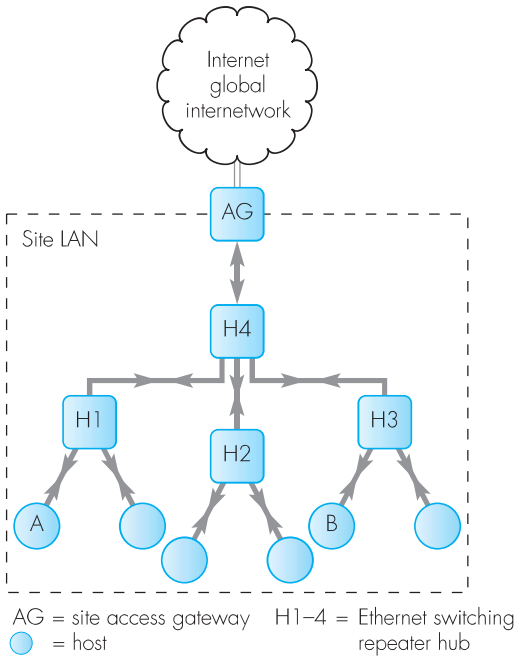


Figure 6.20 Example topology for describing the operation of the ARP.

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 that 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. 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.

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.

ARP/RARP message formats and transmission

As we show in Figure 6.21(a), the formats, 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" to 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 6.2, 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 6.21(b), this is specified in a two-byte *type* field that immediately precedes the message/datagram in the user data field of the

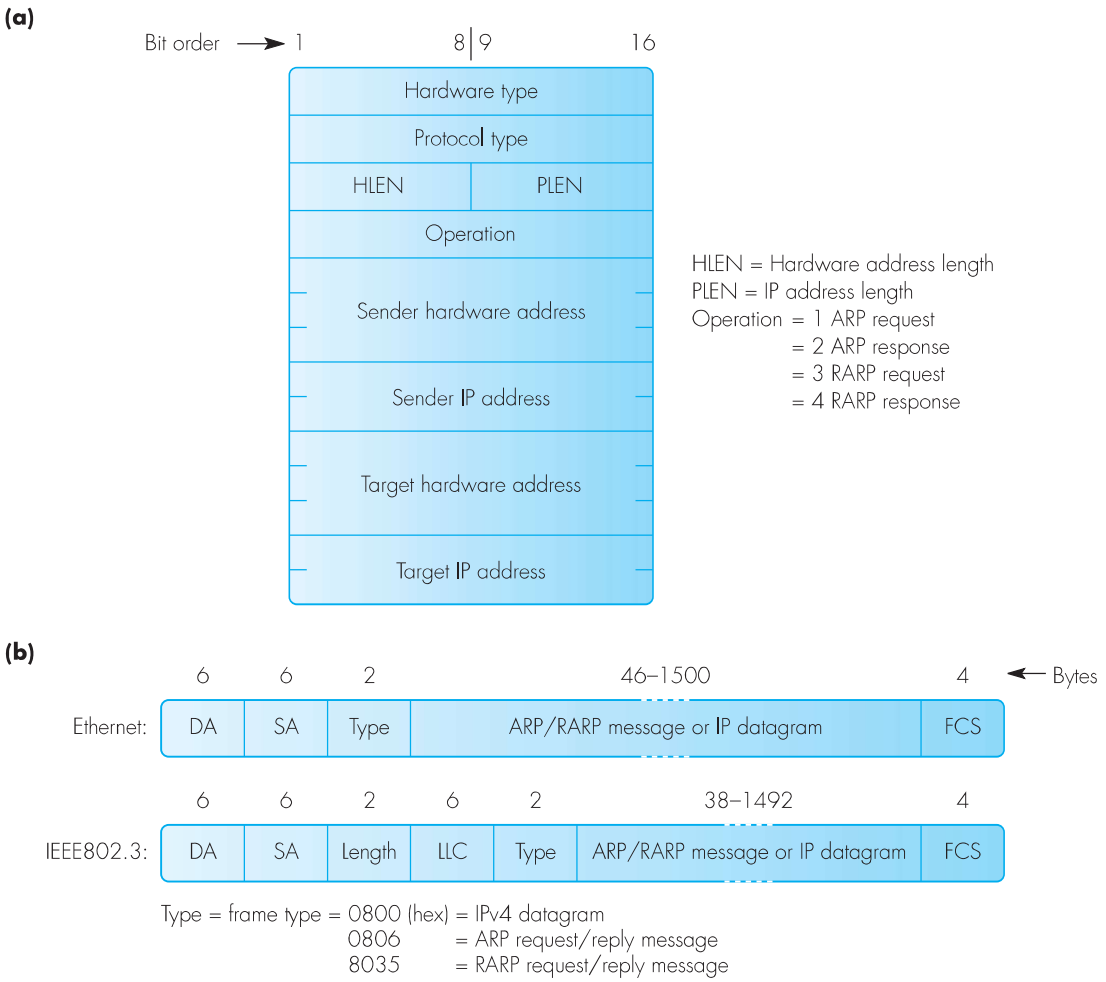


Figure 6.21 ARP and RARP message formats and transmission: (a) ARP and RARP message formats; (b) MAC frame format with Ethernet and IEEE802.3.

MAC frame. As we explained in Section 3.6.3, normally, the LLC sublayer is not used with the original Ethernet MAC standard and hence the *type* field immediately follows the MAC *source address* (*SA*). With the more recent IEEE802.3 MAC standard, however, the LLC sublayer is present and hence the *type* field immediately follows the six 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 6.5

Determine the amount of padding required in a MAC frame when transmitting an ARP/RARP message over (i) an Ethernet LAN and (ii) an IEEE802.3 LAN.

Answer:

As we showed in Figure 6.21(a), each ARP/RARP message comprises 28 bytes. Also, as we explained in Section 3.2.3, the minimum frame size associated with the CSMA/CD MAC method is 512 bits (64 bytes) including the FCS.

(i) With an Ethernet LAN, the header plus FCS requires 18 bytes and hence the minimum size of the user data field is (64 – 18) = 46 bytes. Thus the number of pad bytes required = (46 – 28) = 18 bytes.

(ii) With an IEEE802.3 LAN, the header plus FCS requires 26 bytes and hence the minimum size of the user data field is (64 – 26) = 38 bytes. Thus the number of pad bytes required = (38 – 28) = 10 bytes.

6.6.3 DHCP

As we described earlier in Section 6.1, before a host can communicate over the Internet it must have an IP address. The role of the **dynamic host configuration protocol** (**DHCP**) is to obtain an IP address on behalf of the host. For example, assuming the NAT scheme – Section 6.4.4 – this involves the network manager making a request to ICANN for a site netid with an indication of the number of hostids that are required. The network manager is then responsible for allocating IP addresses for hosts as they are attached to the network and reallocating them as hosts are removed. As we can deduce from this, the allocation, installation and administration of IP addresses can entail considerable effort. To alleviate this, an autoconfiguration facility is available

that enables a host to obtain an IP address over the network and, in the case of mobile hosts, use it for the duration of a single session.

The DHCP scheme is defined in RFCs 2131 and 2132. It involves the host communicating with a site – or enterprise – IP address server using an application protocol called the DHCP. To explain how it works, we shall use the simple example network shown in Figure 6.22(a). In this, we assume that a person arrives with a laptop and wants to communicate over the Internet to obtain e-mail from his/her home network. We assume the network supports autoconfiguration by way of a server that has DHCP as one of its applications. If this was not supported – for example the site LAN is part of a larger enterprise network – then the server runs a related application called a **DHCP relay agent**. This, as its name implies, relays requests and responses to/from the DHCP address server that is located in a different part of the enterprise network. The sequence of messages relating to the DHCP that are exchanged is shown in diagrammatic form in Figure 6.22(b).

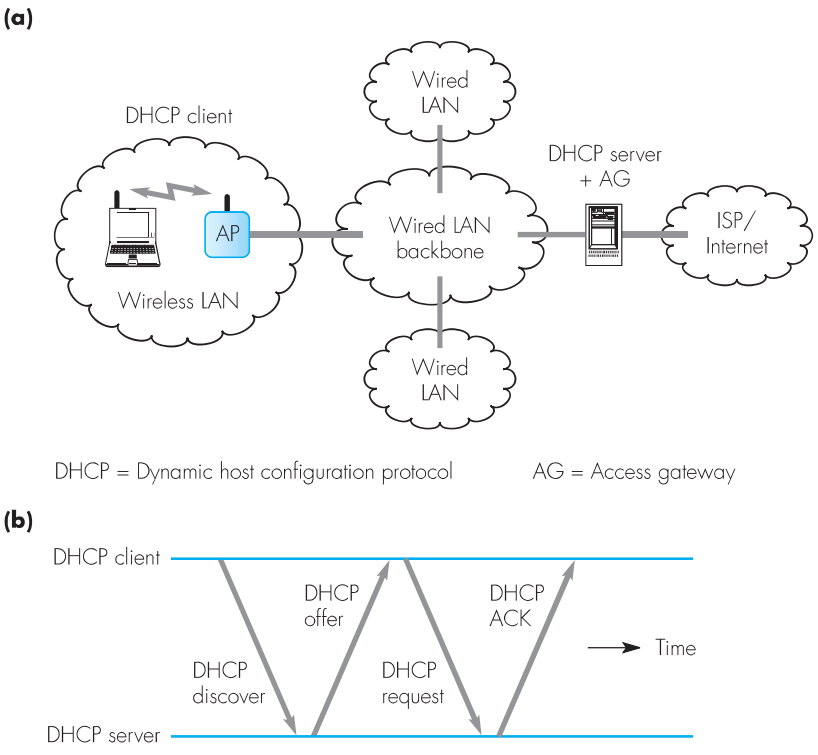


Figure 6.22 The DHCP protocol: (a) example network topology; (b) DHCP message exchange sequence.

As we can see, the DHCP is a simple client-server protocol and a brief description of each message follows.

- **DHCP discover**: this is sent by the DHCP client – that runs in the host – to all hosts/stations using the broadcast destination IP address of 255.255.255.255 and a source address of 0.0.0.0. The message then contains a **transaction ID** so that the client can relate the response from the server to this request.
- **DHCP offer**: this is returned by the DHCP server running in the site server and contains the same transaction ID that was in the discover message, the proposed IP address for the client, the IP address mask, and a duration value indicating the duration of time the IP address is valid.
- **DHCP request**: this is the response to the DHCP offer message and contains the same parameters as were present in the DHCP offer message.
- **DHCP ACK**: this is returned by the server and contains the same set of parameters so acknowledging their receipt by the client.

Normally, the value in the duration field is for a fixed period of time and the IP address is said to be **leased** for this length of time. Then, should the time expire, the client can no longer use the address. To avoid this occurring, prior to the time expiring, the client can make a request to have the time period renewed.

6.6.4 OSPF

The open shortest path first (OSPF) routing protocol is now the preferred interior gateway protocol; that is, for routing within an autonomous system (AS). The latest version is defined in RFC 2328. This is a public document, of course, and explains the use of the word *open* in the name. It can be used with many different types of network including simple point-to-point lines, enterprise LANs, and both regional and national networks. In order to explain how OSPF works and the terminology associated with it, we shall use the (much simplified) Internet structure shown in Figure 6.23.

As we can see in the figure, to route a datagram/packet over the Internet requires four types of router. These are shown as 1, 2, 3 and 4 in the figure and their roles are as follows:

- **Internal routers** (**1**): these are found only in non-backbone areas and carry out the routing of packets within an area.
- **Backbone routers** (**2**): these are found only in an AS backbone area and carry out the routing of packets within the AS backbone.
- **Area border routers** (**3**): these are used to connect an area within an AS to the AS backbone. They are considered to be part of both the area and the backbone area.

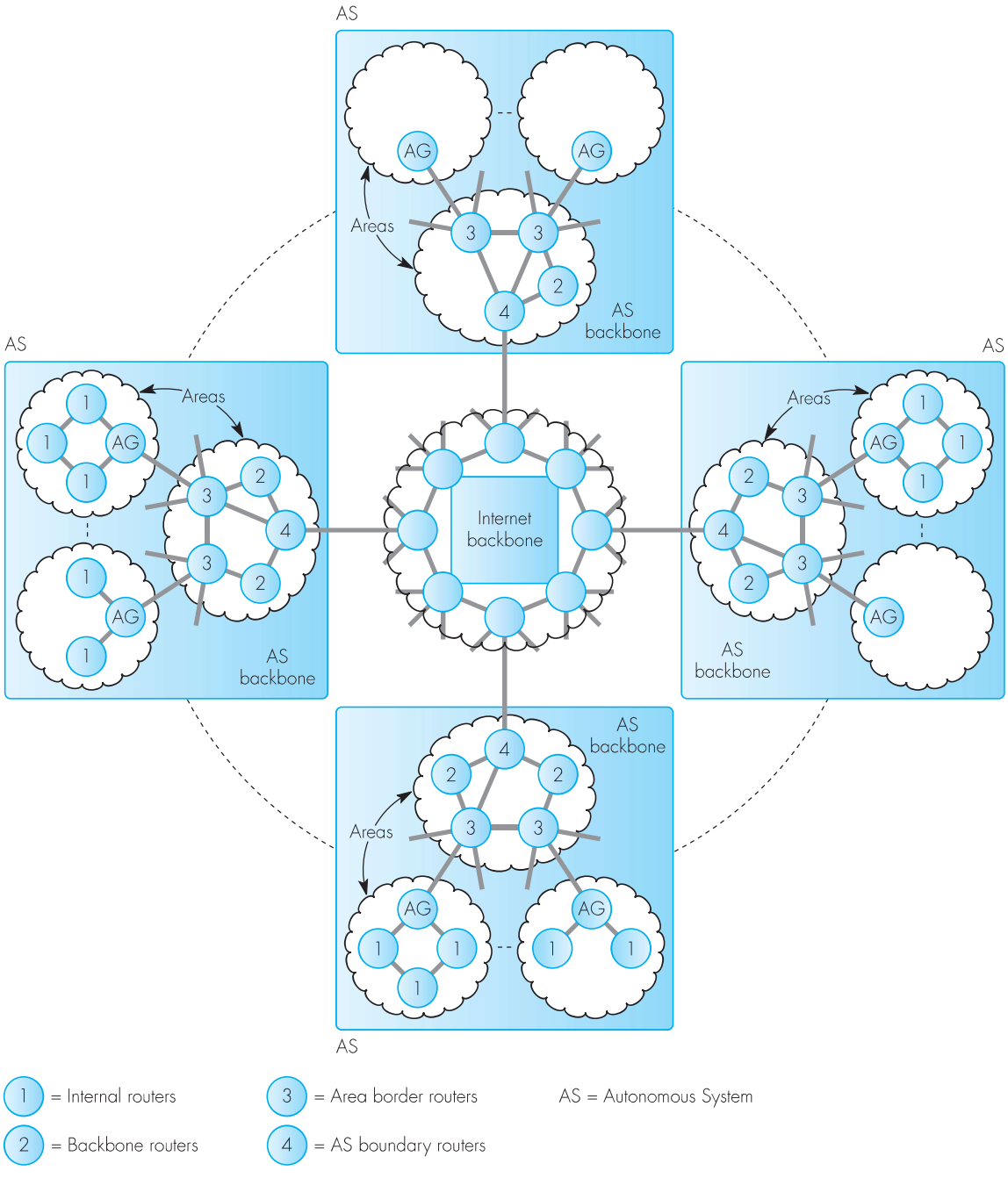


Figure 6.23 Simplified Internet structure for routing purposes.

- **AS boundary routers** (4): these have a connection to the Internet backbone and use BGP to carry out the routing of packets between ASs.

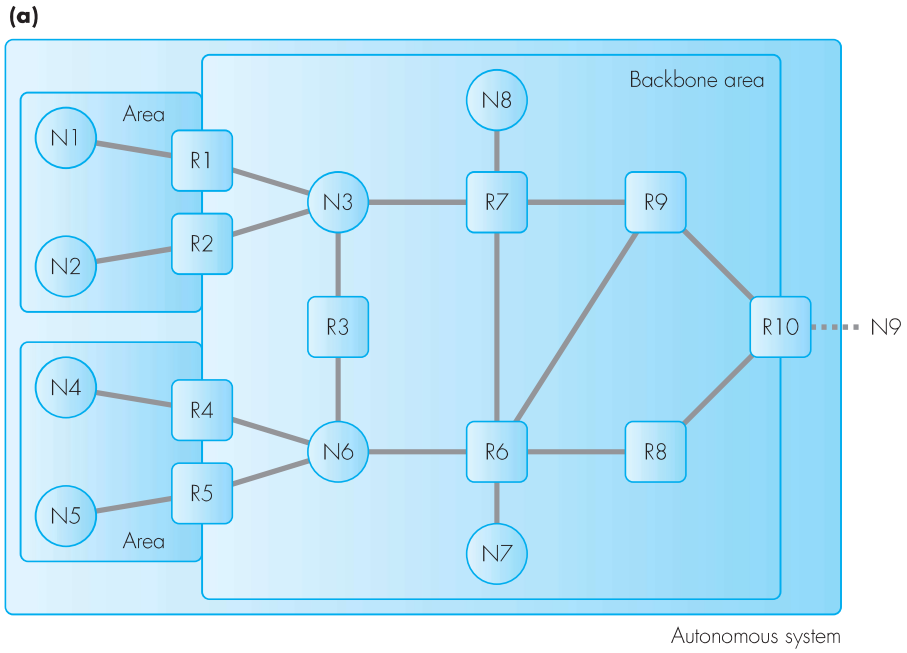
To route a packet within an AS, the packet is first routed to the area border using the routing protocol of that area. Then, since all area border routers are also part of the AS backbone area, the packet is routed over the AS backbone area – using OSPF – to the area border router that is connected to the required destination area. The packet is then forwarded to the destination host using the routing protocol of that area.

To route a packet from a host that is attached to an area network in one AS – AS(1) – to a host attached to an area network within a different AS – AS(2) – the packet is first routed to the boundary router of AS(1) using OSPF. The BGP is then used to relay the packet to the required boundary router of AS(2). The packet is then routed over the AS(2) backbone – again using OSPF – to the destination area border router and from there to the destination host using the area protocol.

The operation of OSPF is the same in principle to the LS-SPF algorithm we described earlier in Section 6.5.4. To explain how it works, we shall use the (much simplified) AS shown in Figure 6.24(a). As we can see, it is composed of two areas plus a single backbone area. First, using the link state algorithm, the topology of each area network within the AS is built up by each router using link state messages. The network is then represented in the form of a **directed graph** and, as an example, the graph of the backbone area in the simplified AS is shown in Figure 6.24(b). The following should be noted when interpreting the graph:

- Two routers that are joined by a point-to-point line – for example the link between R6 and R7 – are represented in the graph by a pair of arcs with the cost value alongside.
- When multiple routers are present in the same network, each router has a pair of arcs and their related cost values. In the example these are equal but may be different.
- When a router is attached to a single network, just a single arc is used called a **stub**.

Each router uses the shortest path first algorithm to compute the shortest path between itself and all the other routers using the cost values associated with each link. The cost values are user-configurable and can be based on delay/distance, bit rate and other parameters. A feature of OSPF is that it equalizes the load over paths that have equal path costs. This has been found to improve the network throughput. As an example, the SPF tree computed by R10 is shown in Figure 6.24(c) and, as we can see, the path cost from R9–R6–R7 is the same as that from R9–R7 and hence both paths can be used by R9 to share the load.



(continued overleaf)

Figure 6.24 OSPF: (a) example AS; (b) directed graph of AS; (c) SPF tree computed by R10.

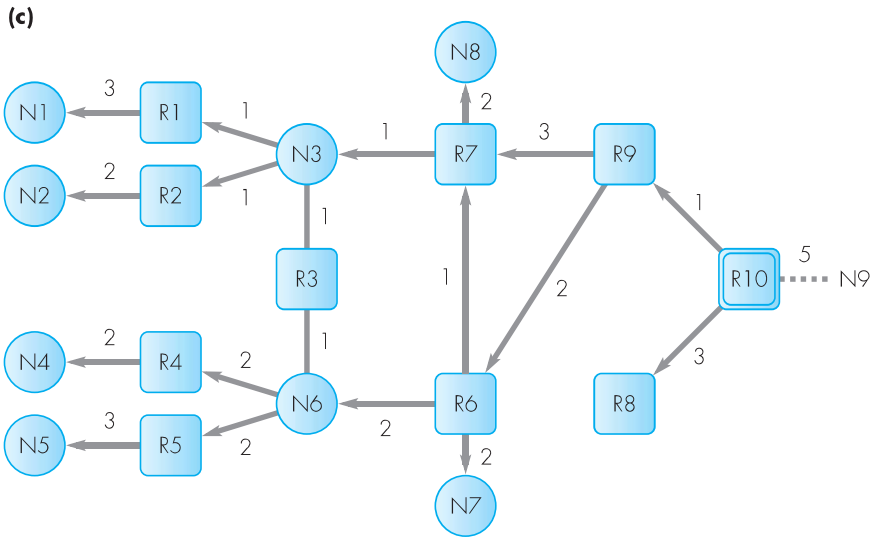


Figure 6.24 Continued.

Once the SPF tree has been computed, each router then creates its own routing table. This contains an entry for each destination netid, the next-hop router to use, and the total path cost to this destination. As an example, the routing table for router R10 is shown in Table 6.1.

Table 6.1 Routing table held by R10.

Destination netid	*Next-hop router*	*Distance*
N1	R9	9
N2	R9	8
N3	R9	5
N4	R9	9
N5	R9	10
N6	R9	5
N7	R9	5
N8	R9	6
N9	Backbone	5

Although the LS-SPF algorithm works for relatively small networks, in practice, many AS backbone networks in the Internet contain a very large number of routers/networks and, as a result, the overheads associated with every router within an AS backbone exchanging routing information with every other router in the backbone would be too large. To overcome this, one router is assigned to be the **designated router** for the AS backbone and all the other routers are then said to be **adjacent** to it. The exchange of routing information then only takes place between the designated router and each adjacent router so reducing considerably the amount of information that is exchanged. The downside of this, however, is that should the designated router crash, then the whole AS backbone is affected. Hence to overcome this possibility, a backup is always available and the routing information it contains mirrors that of the current active designated router.

The message types used in OSPF to enable a router to perform the various routing functions are:

- **Hello**: this is used by a router to discover the networks/routers to which it is directly connected together with the cost value associated with each link.
- **Link state update**: these are sent at periodic intervals by the designated router to all its adjacent routers using flooding, the principles of which we described in Section 6.5.2. These are used to inform each router of the cost and state of each link in the current topological database held by the designated router. Each new update message has a sequence number in it that is used by the receiver to determine whether the message is a new update or an old copy. These messages are also sent by a router if the state or cost of a line changes.
- **Link state ACK**: this acknowledges the correct receipt of the link state update and is returned by each adjacent router.
- **Database description**: this is used to inform the receiver of the message the list of updates that it possesses. Each receiver can then determine whether it has the most up-to-date information.
- **Link state request**: this is used to request link state update information from each adjacent router to it.

Each of the above messages is sent in a separate IP packet.

6.6.5 BGP

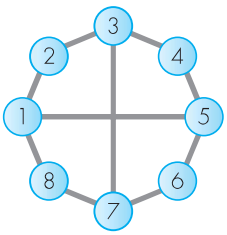
The border gateway protocol (BGP) runs in each AS boundary router. Its role is to communicate routing information to the border routers associated with each of the other ASs that make up the Internet backbone network. The protocol involves the exchange of messages which, in practice, are sent over TCP connections. As we can deduce from this, therefore, BGP is an application protocol rather than a network protocol.

Unlike OSPF, BGP does not communicate cost values to each of its neighbor boundary routers (BR) but instead details of the path followed between itself and the other BRs. Hence when communicating routing information a BR informs its neighbors of the path that it is currently using to reach each other BR. For example, consider the backbone network shown in Figure 6.25.

Assume that the path taken by BR2 to BR5 is 2–3–4–5 and that the paths taken by its two neighbors – BR3 and BR1 – are as shown in the figure. On receipt of the paths from BR1 and BR3 it is easy for BR2 to determine that, based on the number of hops, the path via BR1 is shorter. Each BGP router runs an algorithm that computes the *distance* for each path. Keeping in mind that the AS boundary routers may be located in different countries, the *distance* is determined not just on physical parameters like bit rate and hop count but also by political issues. For example, if two countries fall out with each other then the cost of using that BR is set to infinity and hence it is omitted from all paths.

There are just four message types associated with BGP:

- **Open**: this is used to open a relationship with another AS boundary router to which it is connected.
- **Update**: this is used both to transmit routing information relating to a single route and to list multiple routes to be withdrawn.
- **Keepalive**: this is used both to acknowledge receipt of an *open* message and to periodically confirm the relationship with the other AS boundary router.
- **Notification**: this is used to inform the receiver that an error condition has been detected.



Assume the path used by: BR1 to BR5 is B1–B5
BR2 to BR5 is B2–B3–B4–B5
BR3 to BR5 is B3–B4–B5
and BR2 receives an update message from BR1.
BR2 then determines that the better route to BR5 is BR2–BR1–BR5

Figure 6.25 Routing over the Internet backbone.

To reflect the different types of routing protocol that are involved in routing a packet over the Internet 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, to 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 and to the network access gateway.
- Access gateways maintain sufficient routing information to enable them to route a packet within its own (access) network either directly to a host or, if subnet routing is used, to the subnet router that is on the shortest path to the destination host.
- Area border routers maintain sufficient routing information to enable them to route a packet to another area border router within the same autonomous system and to the AS boundary router if the destination address of the packet is to a different AS.
- AS boundary routers maintain sufficient routing information to route a packet that is destined for a different AS over the Internet backbone network.

This is summarized in Figure 6.26.

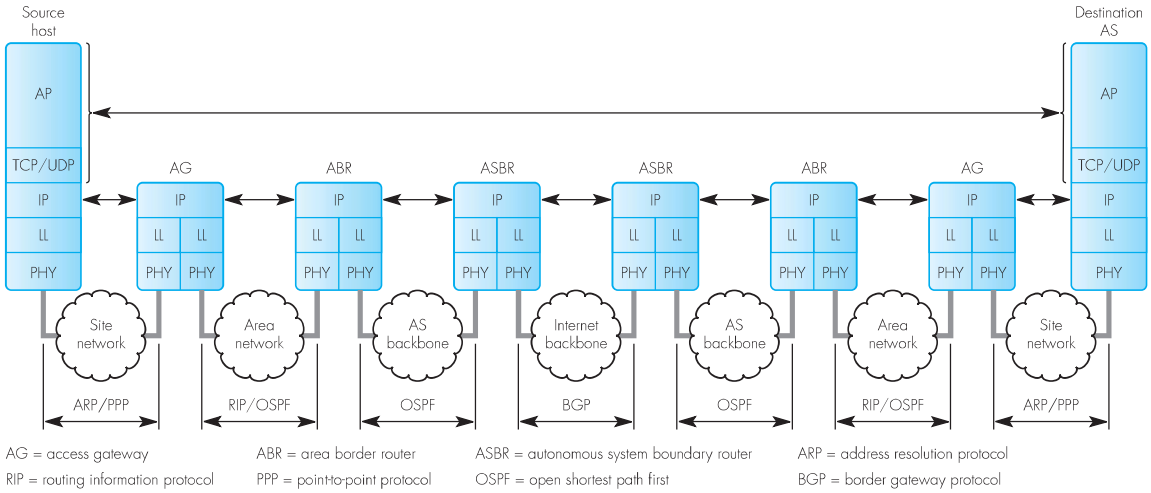


Figure 6.26 A summary of the routers and routing protocols used in the Internet.

6.6.6 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 6.5 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.

For applications such as a conference or meeting that 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 ICANN. 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 that 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 ICANN has been allocated a block of Ethernet MAC addresses for applications that involve multicasting. As we described in Section 3.2.3, 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 ICANN. 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 6.27(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 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 6.27(b).

Multicasting over an internet

When the hosts that are part of a multicast group are attached to different networks/subnetworks geographically distributed around an internet, then intermediate subnet routers and 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:

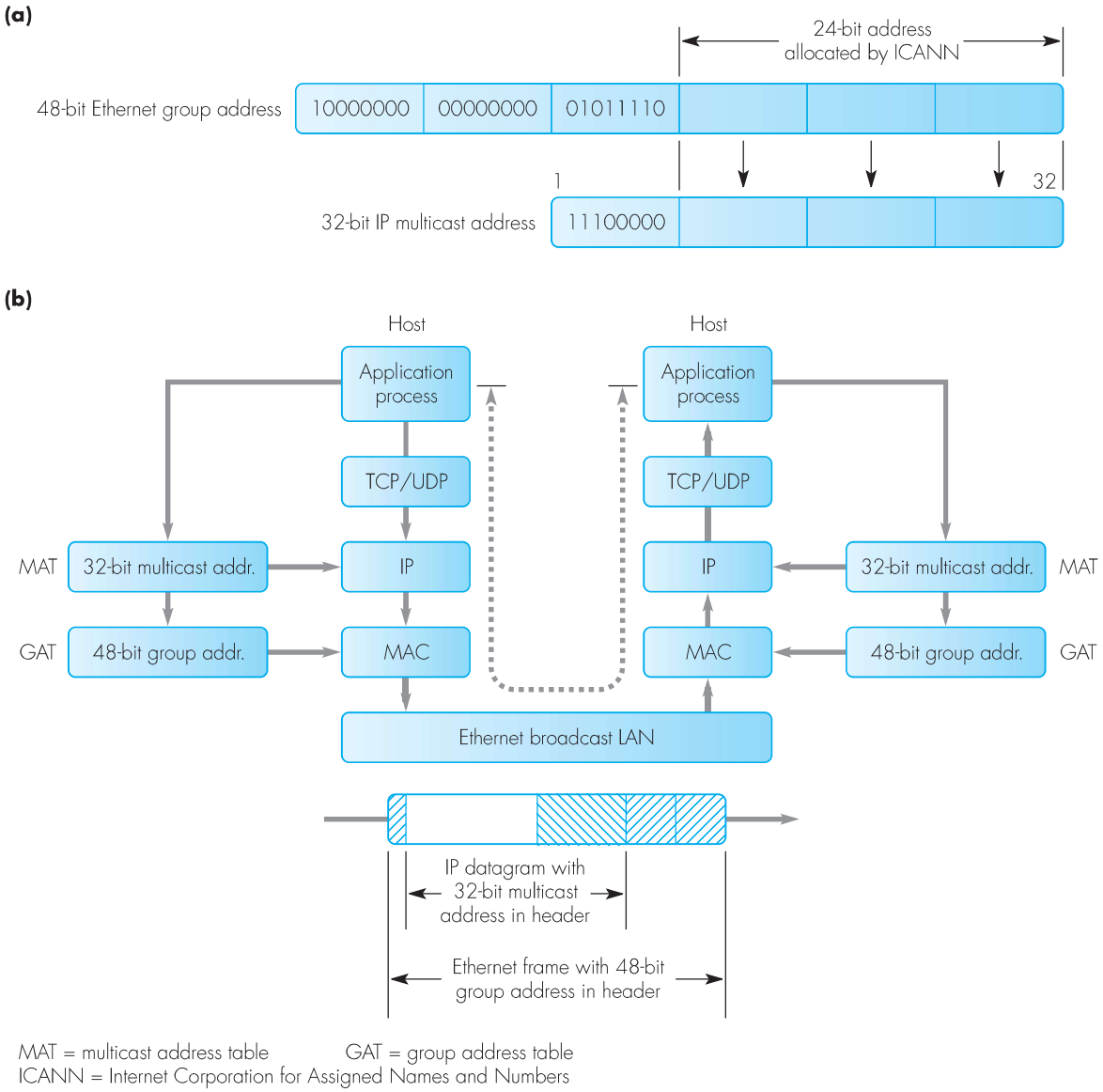


Figure 6.27 Multicasting over a LAN: (a) address allocation principle; (b) address usage.

- 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 that 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 being determined by the routing algorithm 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 6.28(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 6.28(a).

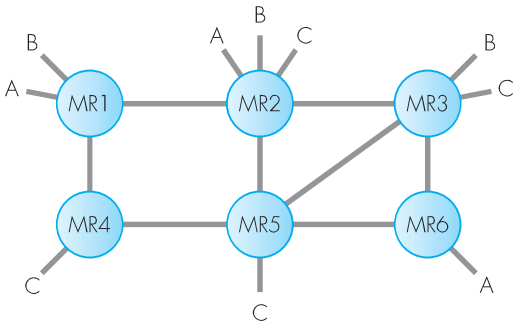
DVMRP

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 6.28(b). They are based on a routing metric of hop count and have been derived using the same procedure we described earlier in Section 6.5.3. Together with the contents of the MAT in each MR, they are used to route all packets that have a destination multicast address.

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.

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

- 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 6.5.6, each computes a spanning tree for the internetwork using the algorithm we

showed in Figure 6.18 and described in the accompanying text. For example, for the internetwork shown in Figure 6.29(a), the spanning tree computed by each MR is shown in Figure 6.29(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 6.29(a), the pruned spanning trees produced by each MR for each of the three multicast addresses are as shown in Figure 6.29(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 and, to explain how it works, assume that a packet with a (multicast) address

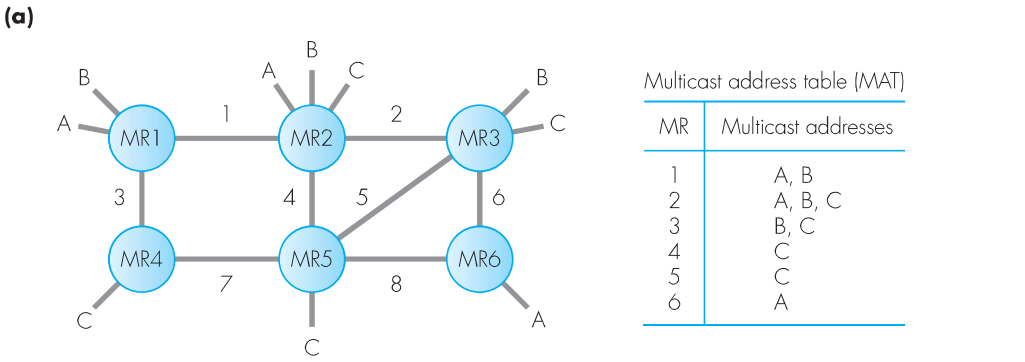


Figure 6.29 Spanning-tree multicast routing: (a) example topology and multicast address table; (b) set of spanning trees for each multicast address.

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.

6.6.7 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 that the mrouter serves are currently members, and second, this information being 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) that we identified earlier in Figure 6.2. The IGMP is an integral part of the IP layer in all hosts and routers that support multicasting. It is defined in RFC 1112.

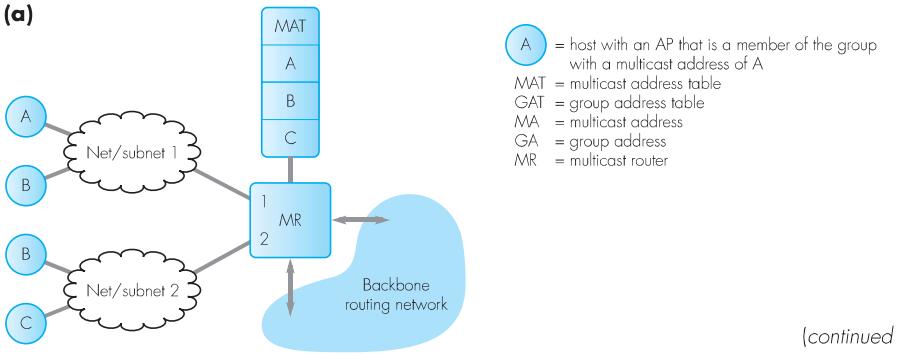
As we explained at the start of the last section, an application process (AP) within a host can join and leave a currently active multicast group at any time. To do this, the AP must know the 24-bit group address that has been allocated to this group by the ICANN. As we showed in Figure 6.27, the AP 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 of the multicast address of the session that it wishes to join. The sequence of steps followed to do this is shown in Figure 6.30.

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 6.30(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 a 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 6.30, 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.



(continued overleaf)

Figure 6.30 IGMP summary: (a) example network topology; (b) IGMP message transfer sequence to join and leave a multicast session; (c) IGMP message format.

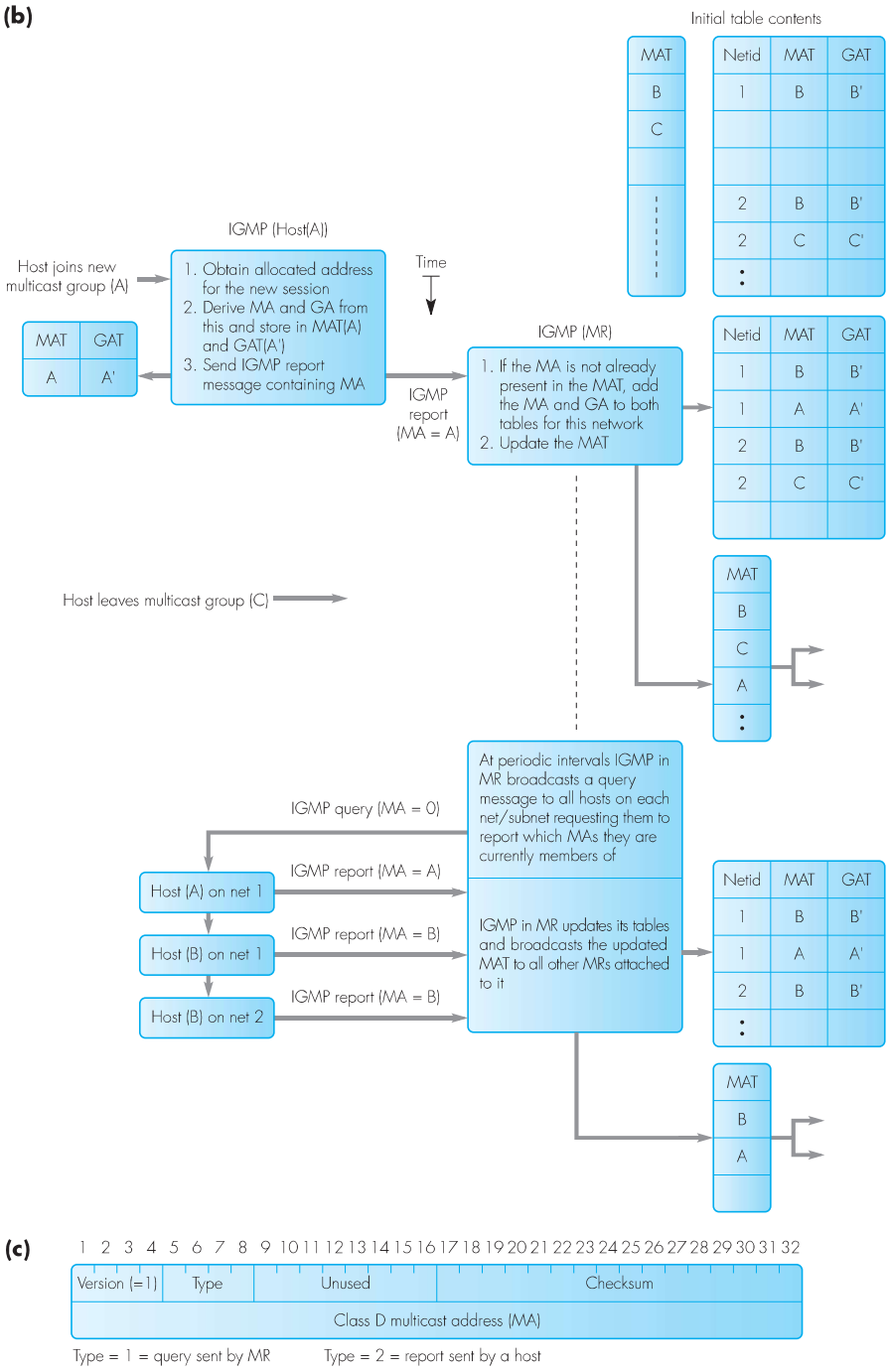


Figure 6.30 Continued.

The format of the two types of IGMP message – query and report – is the same and is shown in Figure 6.30(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.

6.6.8 M-bone

As we explained in Section 6.6.6 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 that have a multicast destination address. These are known as multicast routers (mrouters) and the network formed by the interconnected set of mrouters is known as the **multicast backbone** (**M-bone**) **network**. The two routing algorithms we described in Section 6.6.6 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 6.5.5, 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 6.6.6.

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.

6.6.9 ICMP

The Internet control message protocol (ICMP) forms an integral part of all IP implementations. It is defined in RFC 1256 and 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 6.2. 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.

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.

Table 6.2 ICMP message types and their use

Function	*ICMP message*	*Use*
Error reporting	Destination unreachable	A datagram has been discarded for the reason specified in the message
	Time exceeded	Time-to-live parameter in a datagram expired and hence datagram 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

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 that 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 6.7.

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.

ICMP message formats and transmission

The format of an ICMP message is shown in Figure 6.31. 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 to reachability testing, 4 to 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 that the destination network is unknown and 7 that 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

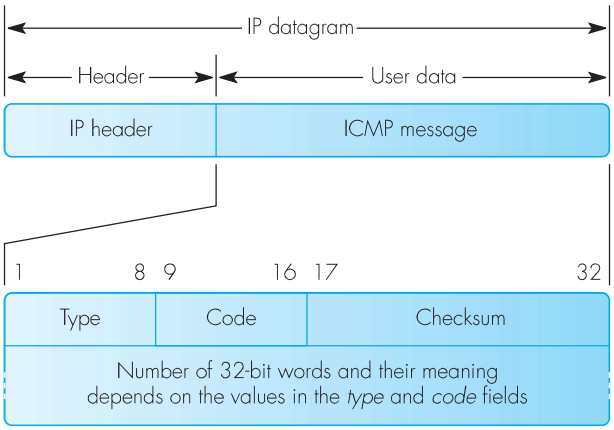


Figure 6.31 ICMP message format and transmission.

number and meaning of the following 32-bit words then depend on the *type* and *code* fields in the header.

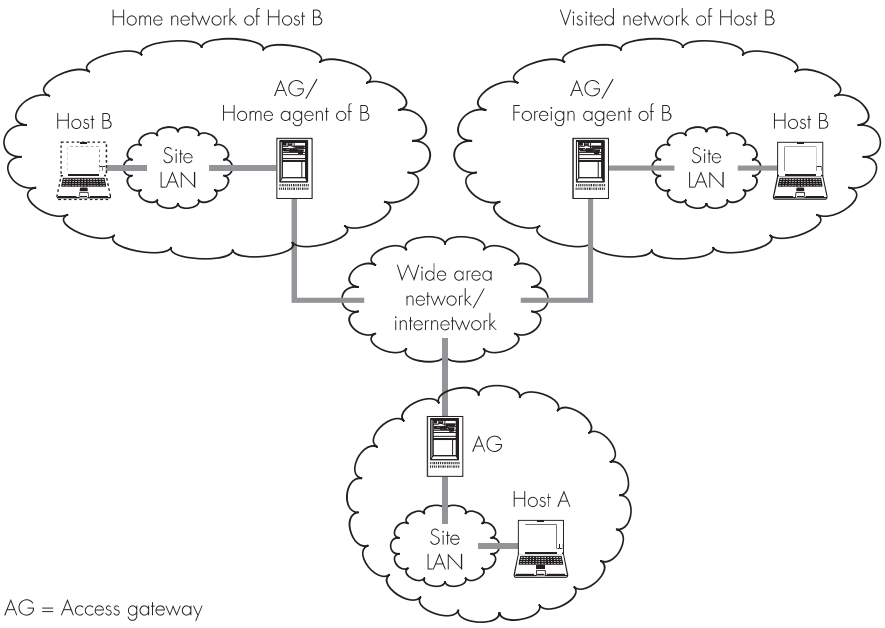
As we showed in Figure 6.3 and explained in the accompanying text, the *protocol* field in each datagram header is used to route the payload in a datagram to the appropriate protocol – ICMP, IGMP, OSPF, TCP or UDP. In the case of reply messages, normally, these are for use by the local IP. When unsolicited 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.

6.6.10 Mobile IP

We studied the various types of wireless access networks in Chapter 4 and, as we summarized in Figure 4.22, a user with a mobile device – laptop, mobile phone, notebook, etc. – can now gain access to the Internet from virtually anywhere in the world. As we showed in the figure, this can be through a public/private wireless LAN or, with a Bluetooth-enabled mobile phone, a mobile phone network. In this subsection we study how a user with, say, a laptop is able to temporarily relocate to a different network and still be able to be contacted by a correspondent host that is located in a different place/network.

A host that is permanently attached to a fixed network such as a home/campus/office LAN is said to be **stationary** and the IP address associated with it is the home location **permanent address**. As we learned from the earlier sections of the chapter, the netid part of the permanent IP address relates to a fixed network which, in this case, is the home/campus/office LAN. Also, all the routing tables within the Internet are built up to route the IP packets relating to a session to/from any other stationary device that is located anywhere in the world back to the home location of the host. Clearly, therefore, it is not possible for the owner of a laptop to simply take it away and attach it to a different network since the netid part of the IP address relates only to the home network. In this subsection, we shall study how this problem is overcome. The Internet architecture and the terminology and protocols used are collectively called **mobile IP** and the standard relating to this is defined in RFC 3220.

To explain how this is achieved, we shall use the simplified network architecture shown in Figure 6.32. In this, host A is located in a different network – called the correspondent network – from the home location network of host B. Whilst in its own network host B has a permanent IP address. The user of host A knows this address – the domain name in practice – and hence can communicate with host B by using it. However, we now assume that the owner of host B has left to visit a different site and has taken his/her laptop to the visited network. Whilst there, the owner of host A wants to communicate with the owner of host B but cannot do this because he/she knows only the permanent IP address of host B. The problem, therefore, is how does host A communicate with host B now that it has been relocated to the visited network.



The problem: Host A wants to communicate with Host B but B has left his/her home network and has taken his/her laptop to a different network – the visited network. Host A knows only the IP address of B at the home network. How does Host A communicate with Host B when it is in the vistied network?

Figure 6.32 Mobile IP – the simplified network architecture and terminology.

As we show in the figure, associated with the home network of host B is a **home agent** (**HA**). This is a process and in the figure it is running in the access gateway of the home network. The role of the home agent is to keep a record of any hosts that have a permanent IP address at the home network and are currently visiting a different network. Similarly, the network host B is visiting has a **foreign agent** (**FA**). This also is a process that, in the figure, is shown running in the access gateway of the visited network. The role of the foreign agent is to keep a record of all mobile hosts that are currently visiting this network. Each time a mobile host becomes attached to the visited network the host must register itself with both the foreign agent and also its home agent. The procedure followed is summarized in Figure 6.33.

To start the procedure, the host must first obtain the IP address of the foreign agent. This can be done in one of two ways. In the first method – called **agent advertisement** – at periodic intervals the foreign agent broadcasts an ICMP message that contains the IP address of the foreign agent. As we can see, the *Type* field has a value of 9, which indicates it is a **router advertisement**

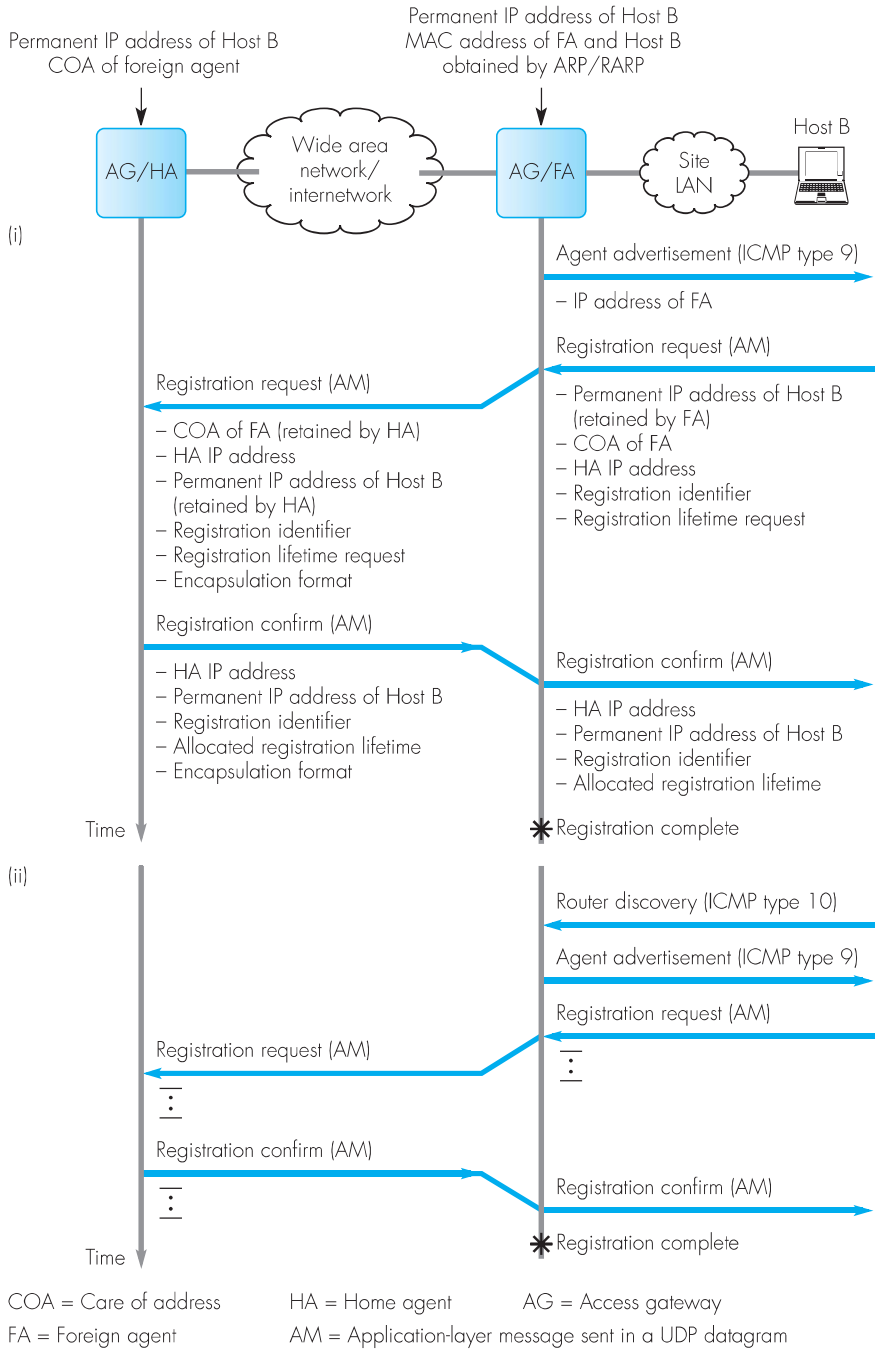


Figure 6.33 Mobile IP – registration of a mobile host (Host B) with the FA and HA.

message. The IP address of the foreign agent is then found in the *Router address* field of the ICMP message. The information in the extension message includes a *Registration identifier* field, a *Registration lifetime* field and one or more **care-of-address** (**COA**) fields. *The Registration* identifier is a 64-bit field and is present in each ICMP message that relates to a single registration operation. There is a fixed time duration for which the registration is valid and this is indicated in the *Registration lifetime* field.

In this example, the COA is that of the foreign agent. Host B then registers with the foreign agent by returning a *Registration request message* that contains the parameters shown in the figure and some security information that includes a time-stamp indicating the time the visiting host was registered. On receipt of this, the FA enters the permanent IP address of host B and the COA of the FA into its routing table and forwards the message to the HA. On receipt of this, the HA enters the permanent IP address of host B and the COA of the FA into its routing table. It then returns a *Registration confirm message* back to the FA with the contents as shown in the figure. At this point the registration is complete.

In the second method – called **agent solicitation** – the mobile host does not wait for an *agent advertisement message* and instead broadcasts an ICMP message with a *Type* value of 10. This indicates that it is an ICMP **router discovery** message. On receiving this, the foreign agent returns an agent advertisement ICMP message to the visiting host directly and the host then proceeds as in the first method. Once the registration of the mobile host with the home and foreign agents has been carried out, the exchange of packets between host A and host B can start. The route followed by each packet is shown in Figure 6.34.

Each IP packet relating to the session contains in its header the IP address of the correspondent host – host A – and the permanent IP address of the mobile host – host B. Hence when the packet is to be sent to the home agent this can be done directly – shown as (1) in the figure. However, when the packet is sent from the home agent to the foreign agent – (2) – it is necessary to encapsulate the packet inside another packet that has the IP address of the foreign agent in the destination address field and the IP address of the home agent in the source address field. On receipt of the packet, the foreign agent simply extracts the original packet from the received packet and then sends it over the visited network using the MAC address of host B obtained using ARP – (3). Similarly, when host B replies, the foreign agent receives a copy of the packet using the MAC address of the FA and detects that the destination IP address in the packet header is that of host A and the source address is the permanent IP address of host B. It therefore sends the packet directly to host A unchanged – (4).

As we can see in the figure, the above procedure is relatively inefficient since the route followed by each pair of packets that are exchanged involves four hops. There are a number of ways of improving this but, in general, these involve the home agent of the correspondent network and hence involve additional procedures.