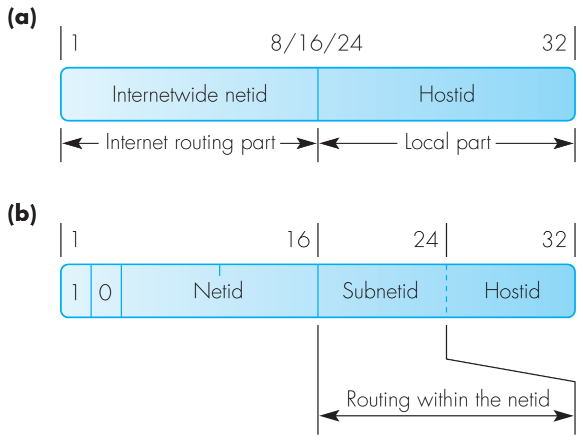


Figure 6.6 Subnet addressing: (a) address structure; (b) example.

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.

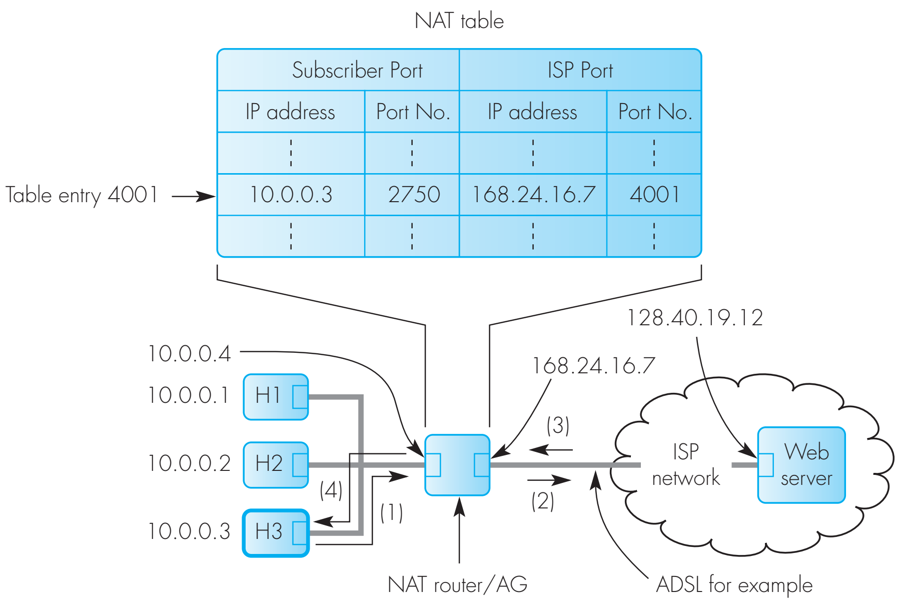
The aim of the network address translation (NAT) scheme is for each access network to be allocated just a single IP address or, for a large site, a small number of addresses. The allocated address is then used by all the hosts for communicating outside their local access network. For communications within 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)



NAT table

Subscriber Port		ISP Port	
IP address	Port No.	IP address	Port No.
⋮	⋮	⋮	⋮
10.0.0.3	2750	168.24.16.7	4001
⋮	⋮	⋮	⋮

(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

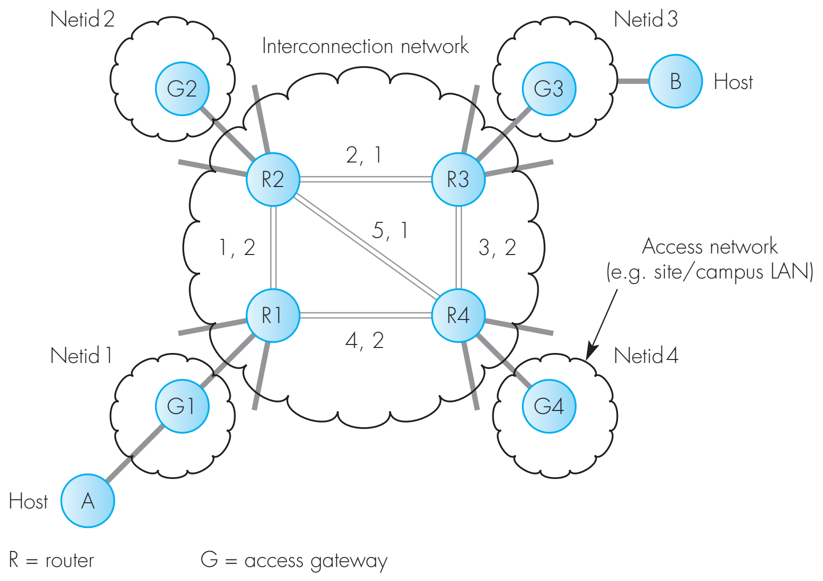


Figure 6.8 Example internetwork topology.

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?

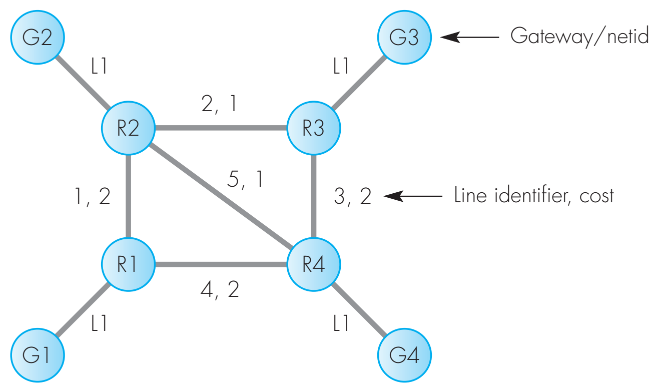
In relation to the last point, we can accept that a simple protocol can be used to enable each gateway to inform the router to which it is attached of the netid of the access network. The first two points, however, are both parts of the routing algorithm associated with the interconnection network. There 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.

(a)



(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.

HALS_C06.qxd 16/11/04 1:31 pm Page 341

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

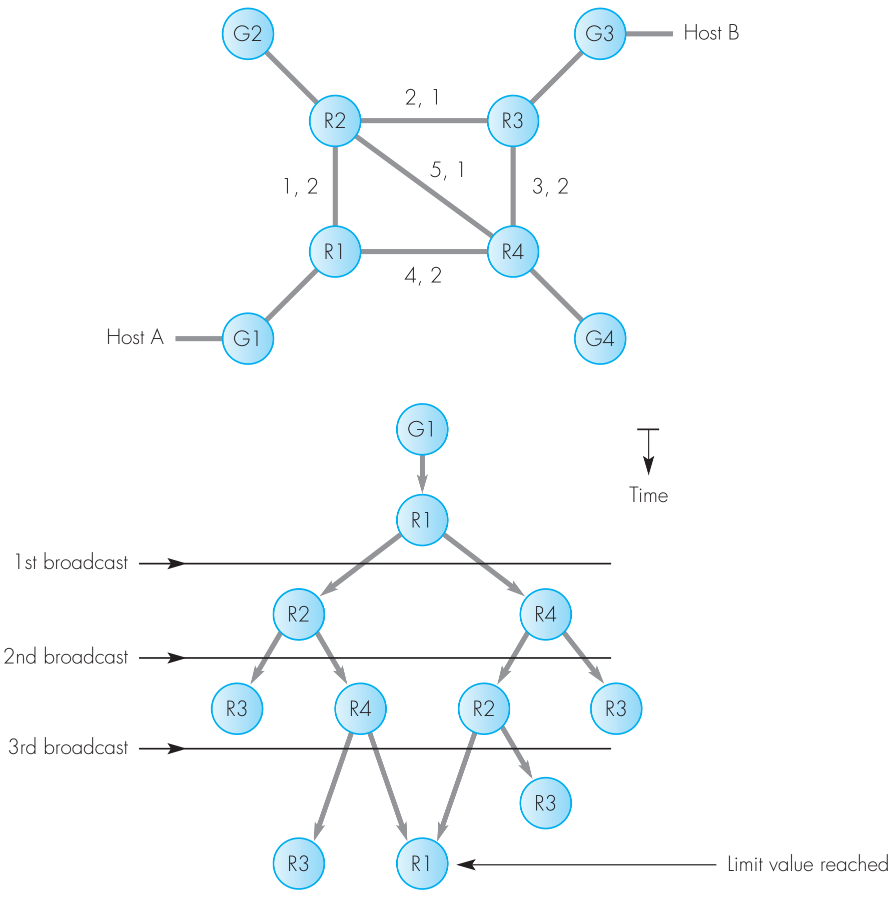


Figure 6.10 Flooding example.

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.

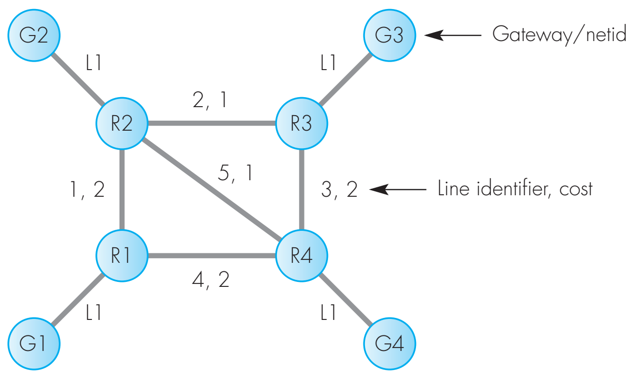
6.5.3 Distance vector routing

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)

R2 R4 → R1:

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

R1 R3 R4 → R2:

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

R2 R4 → R3:

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

R1 R2 R3 → R4:

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

R2 R4 → R1:

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

R1 R3 R4 → R2:

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

R2 R4 → R3:

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

R1 R2 R3 → R4:

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

Figure 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

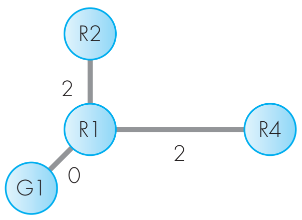
R4:

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

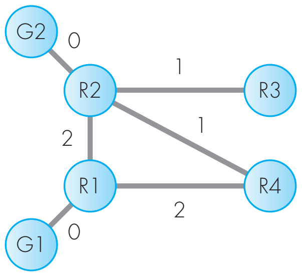
(b)

Topology build-up by R1:

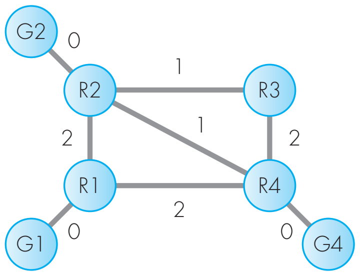
(i) Initial:



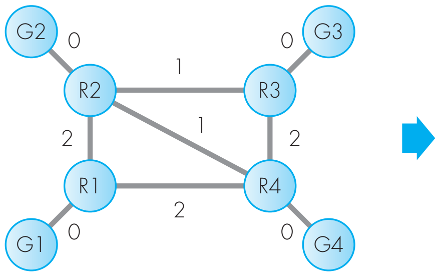
(ii) After connectivity information from R2:



(iii) After connectivity information from R4:

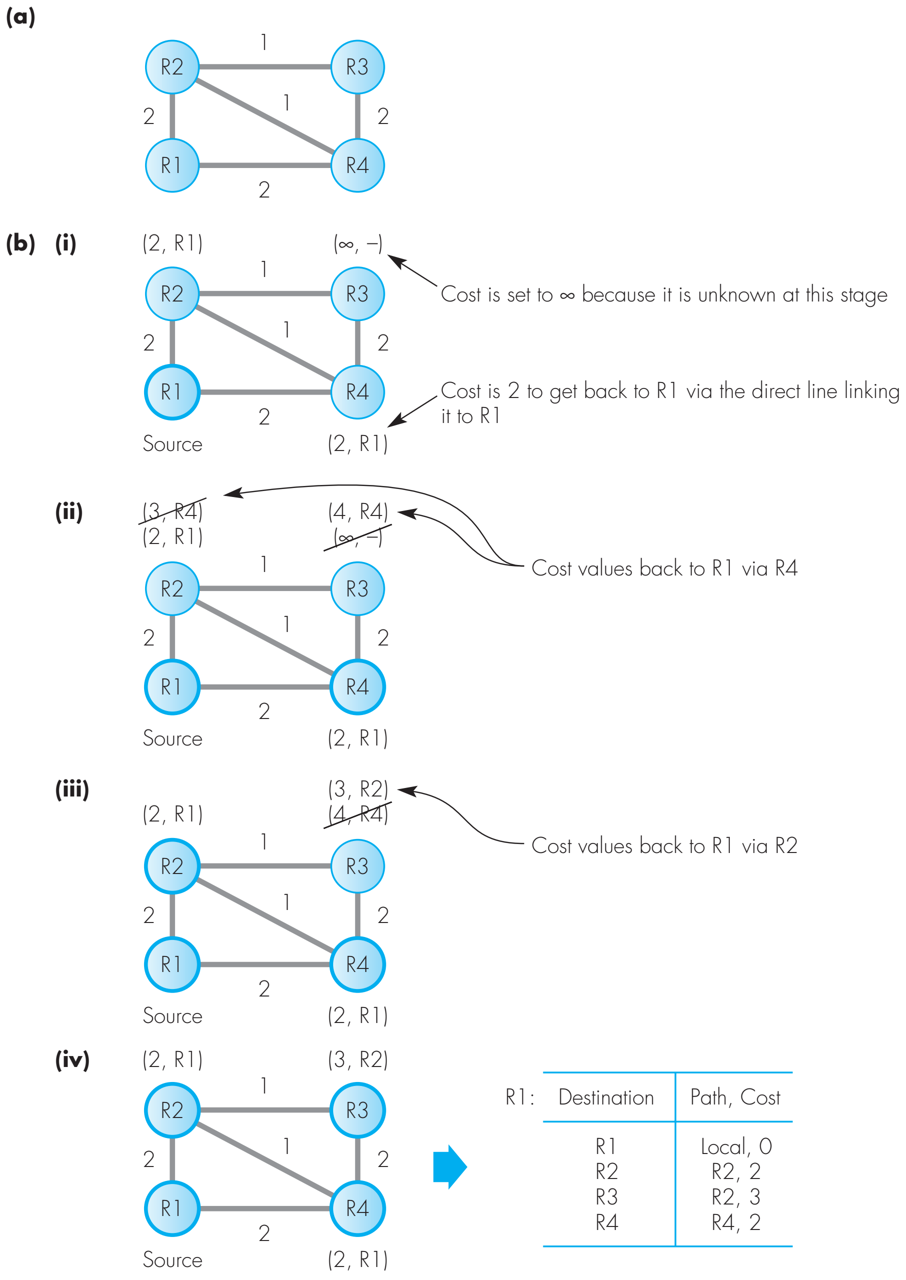


(iv) After connectivity information from R3 via R2:



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.


(a)
R2
R3
R1
R4
1
1
2
2
2
(b) (i)
(2, R1)
(∞, –)
Cost is set to ∞ because it is unknown at this stage
Cost is 2 to get back to R1 via the direct line linking it to R1
Source
(2, R1)
(ii)
~~(3, R4)~~
(2, R1)
(4, R4)
~~(∞, –)~~
Cost values back to R1 via R4
(iii)
(3, R2)
~~(4, R4)~~
Cost values back to R1 via R2
(iv)
(2, R1)
(3, R2)
R1:

Destination	Path, Cost
R1	Local, 0
R2	R2, 2
R3	R2, 3
R4	R4, 2

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

HALS_C06.qxd 16/11/04 1:31 pm Page 348

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.

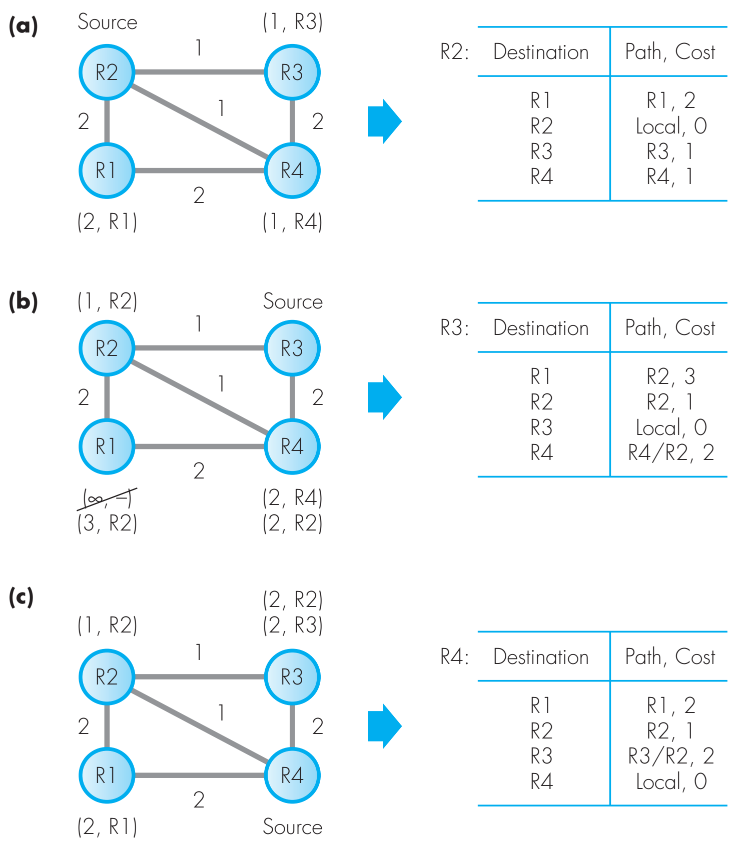
In Figure 6.14 we show the same procedure applied first with R2 as the source – part (a) – then with R3 – part (b) – and finally with R4 – part (c). From these derivations we can make some observations about the algorithm:

- 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



R2:

Destination	Path, Cost
R1	R1, 2
R2	Local, 0
R3	R3, 1
R4	R4, 1

R3:

Destination	Path, Cost
R1	R2, 3
R2	R2, 1
R3	Local, 0
R4	R4/R2, 2

R4:

Destination	Path, Cost
R1	R1, 2
R2	R2, 1
R3	R3/R2, 2
R4	Local, 0

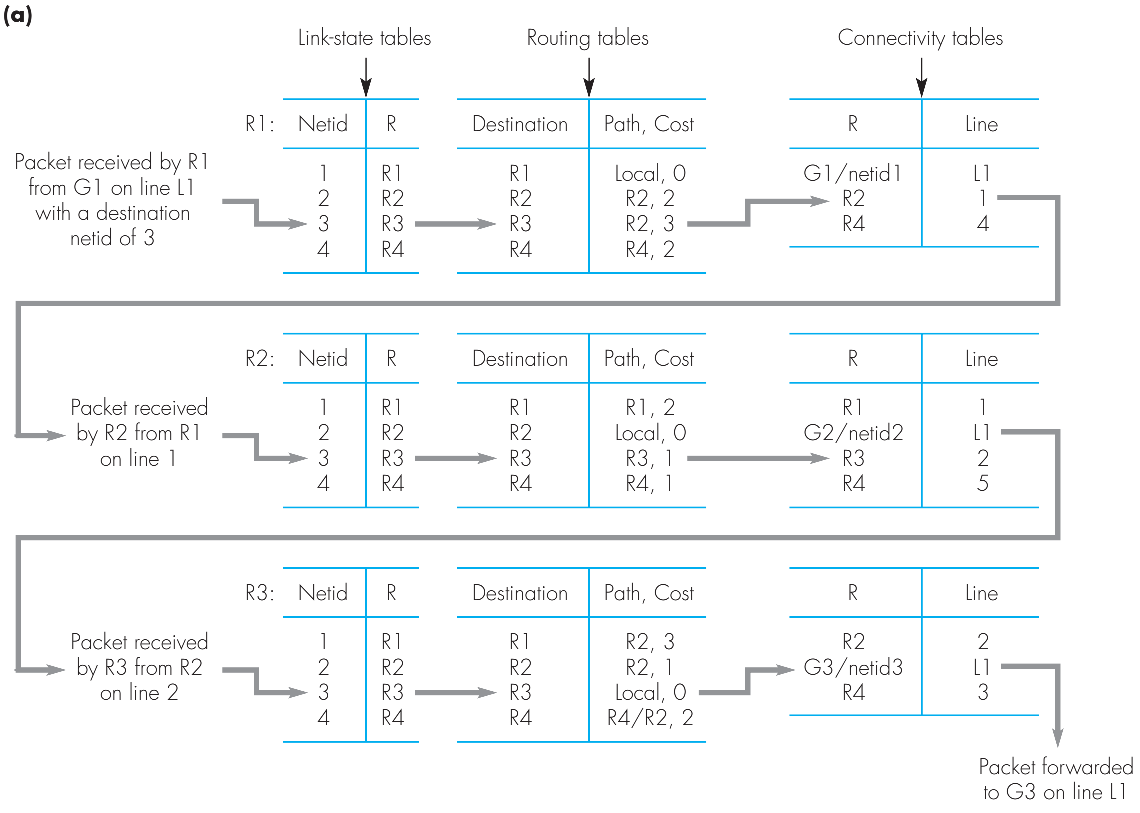
Figure 6.14 Shortest path derivations: (a) by R2; (b) by R3; (c) by R4.

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

(a)


Link-state tables
Routing tables
Connectivity tables
Packet received by R1 from G1 on line L1 with a destination netid of 3
Packet received by R2 from R1 on line 1
Packet received by R3 from R2 on line 2
Packet forwarded to G3 on line L1

R1:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	Local, 0
R2	R2, 2
R3	R2, 3
R4	R4, 2

R	Line
G1/netid1	L1
R2	1
R4	4

R2:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	R1, 2
R2	Local, 0
R3	R3, 1
R4	R4, 1

R	Line
R1	1
G2/netid2	L1
R3	2
R4	5

R3:

Netid	R
1	R1
2	R2
3	R3
4	R4

Destination	Path, Cost
R1	R2, 3
R2	R2, 1
R3	Local, 0
R4	R4/R2, 2

R	Line
R2	2
G3/netid3	L1
R4	3

(b)

1. Datagram received by R1 from G1 on line L1
2. R1 uses its set of routing tables to determine the least path cost is via routers R2, R3, and writes this list into an options field in the datagram header.
3. R1 uses its own connectivity table to forward the packet to the first router in the list, R2, using line 1.

$\downarrow$

4. On receipt of the packet, R2 reads the second router from the list, R3, and uses its own connectivity table to forward the packet on line 2.

$\downarrow$

5. On receipt of the packet, R3 determines it is for one of its local gateways and uses its own connectivity table to forward the packet to G3 on line L1.

Figure 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.

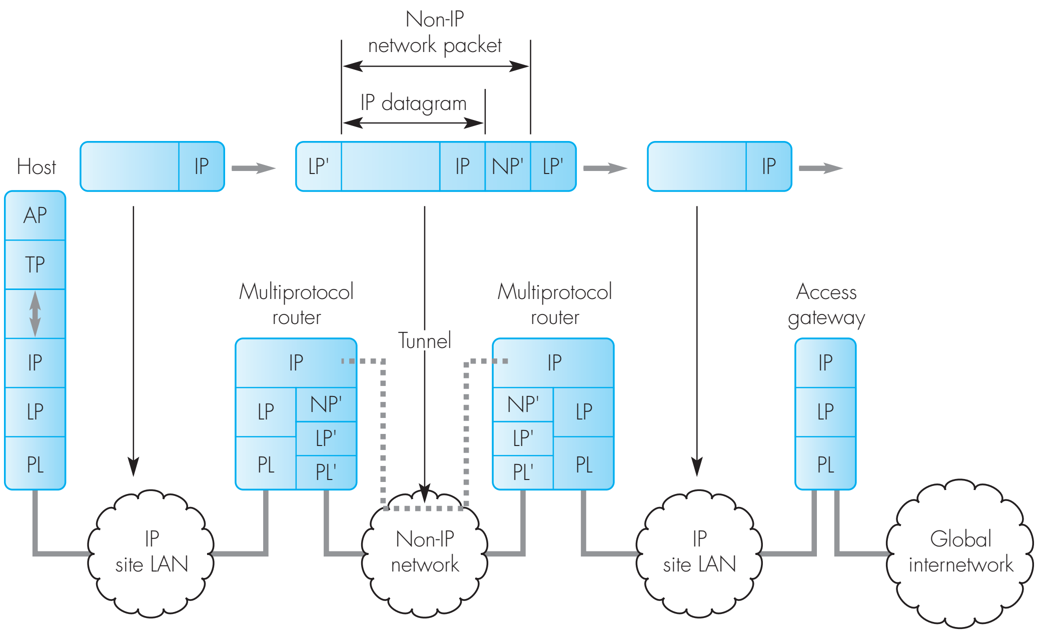


Figure 6.16 Tunneling example.

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

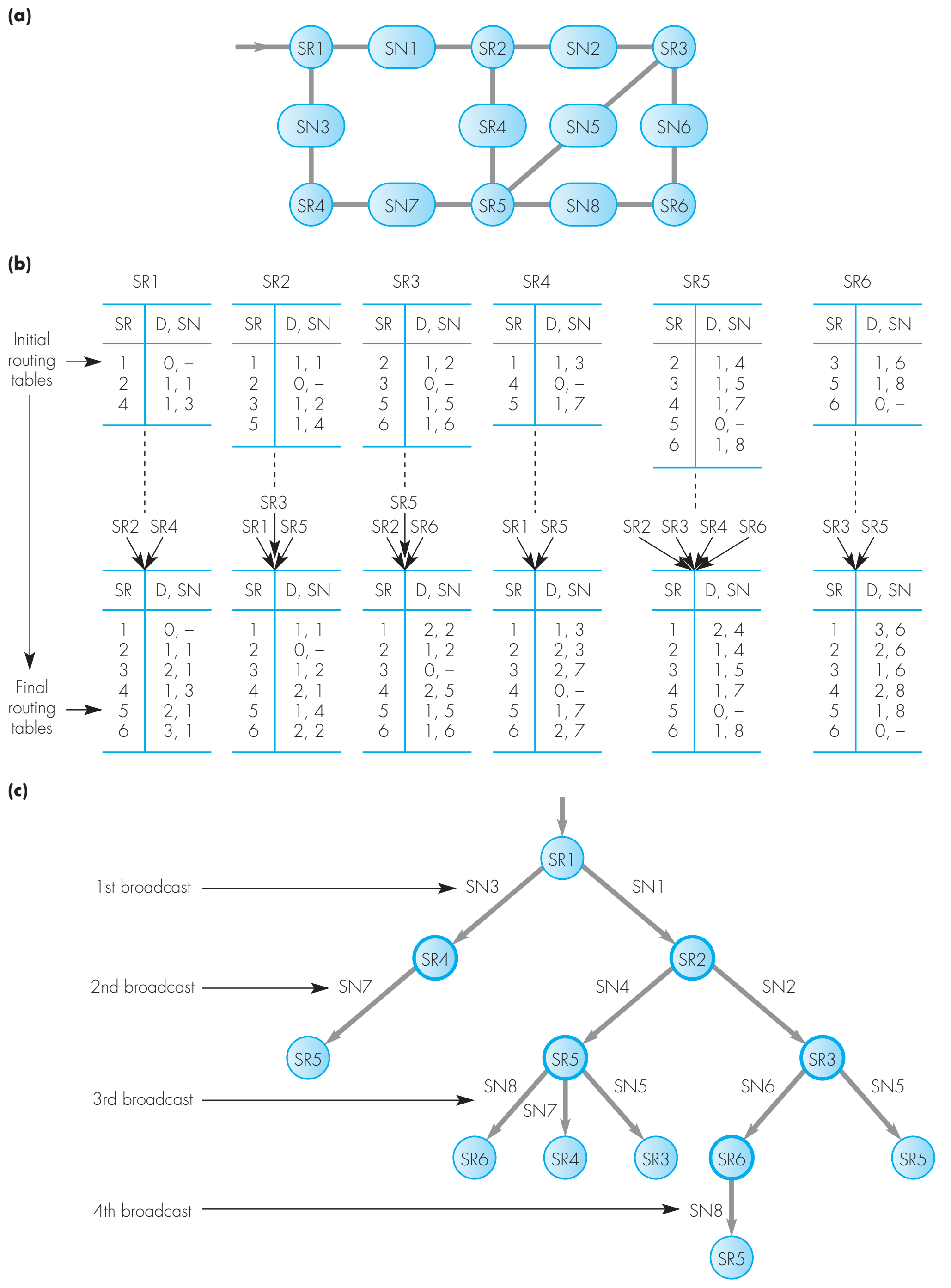
This algorithm is used primarily with networks that use the distance vector (DV) algorithm to route unicast packets. To explain the operation of the algorithm we use the network topology shown in Figure 6.17(a). We assume that subnet router 1 (SR1) also acts as the (single) access gateway for the network 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.



Initial routing tables

SR1

SR	D, SN
1	0, –
2	1, 1
4	1, 3

SR2

SR	D, SN
1	1, 1
2	0, –
3	1, 2
5	1, 4

SR3

SR	D, SN
2	1, 2
3	0, –
5	1, 5
6	1, 6

SR4

SR	D, SN
1	1, 3
4	0, –
5	1, 7

SR5

SR	D, SN
2	1, 4
3	1, 5
4	1, 7
5	0, –
6	1, 8

SR6

SR	D, SN
3	1, 6
5	1, 8
6	0, –

Final routing tables

SR1

SR	D, SN
1	0, –
2	1, 1
3	2, 1
4	1, 3
5	2, 1
6	3, 1

SR2

SR	D, SN
1	1, 1
2	0, –
3	1, 2
4	2, 1
5	1, 4
6	2, 2

SR3

SR	D, SN
1	2, 2
2	1, 2
3	0, –
4	2, 5
5	1, 5
6	1, 6

SR4

SR	D, SN
1	1, 3
2	2, 3
3	2, 7
4	0, –
5	1, 7
6	2, 7

SR5

SR	D, SN
1	2, 4
2	1, 4
3	1, 5
4	1, 7
5	0, –
6	1, 8

SR6

SR	D, SN
1	3, 6
2	2, 6
3	1, 6
4	2, 8
5	1, 8
6	0, –

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:

HALS_C06.qxd 16/11/04 1:31 pm Page 358

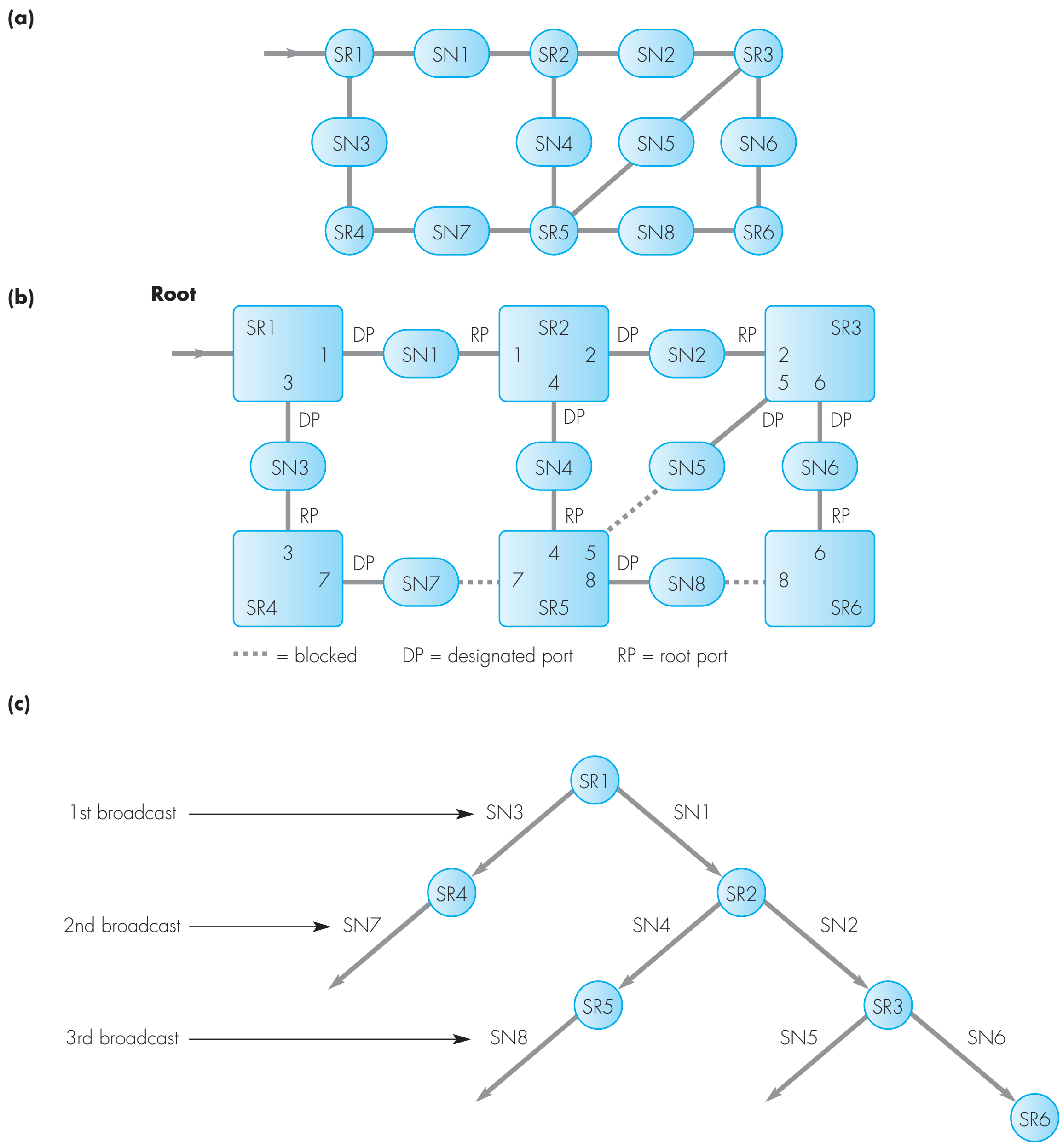


Figure 6.18 Spanning tree broadcast: (a) network topology; (b) spanning tree derived by each subnet router; (c) broadcast sequence.

HALS_C06.qxd 16/11/04 1:31 pm Page 359

- 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

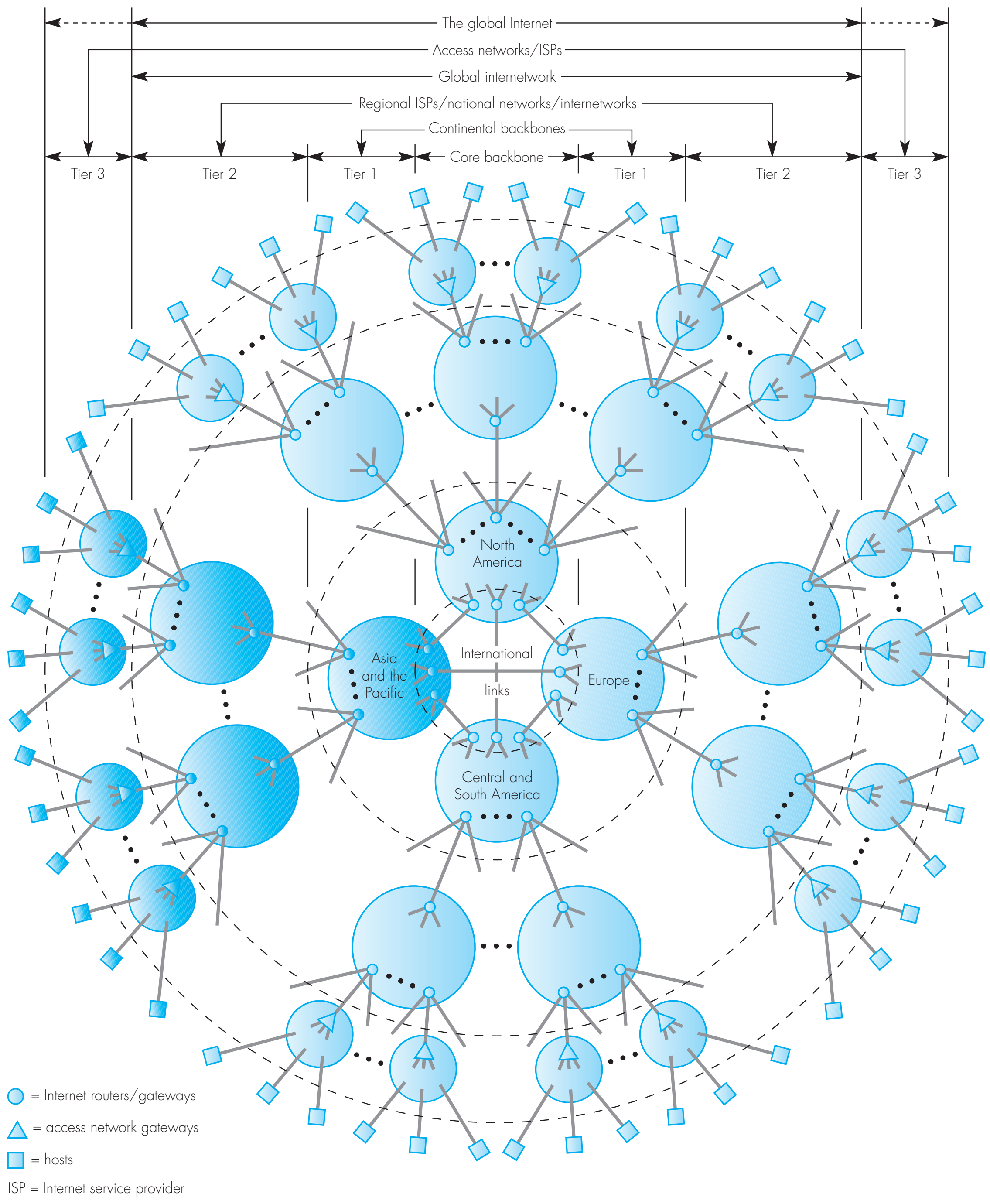


Figure 6.19 General architecture of the global Internet.

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

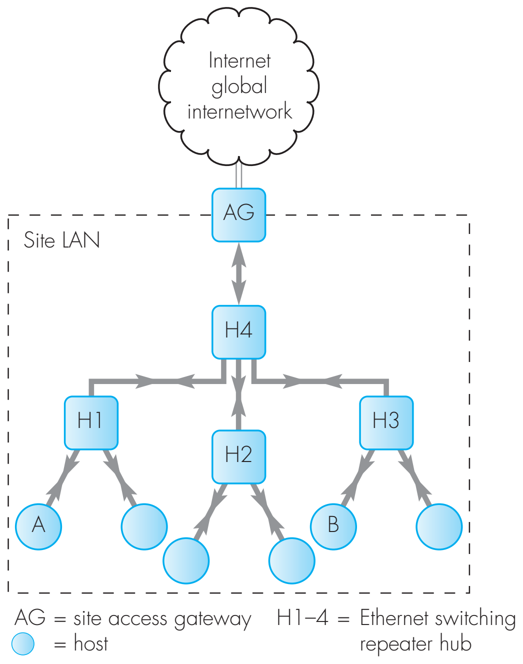

Internet global internetwork
AG
Site LAN
H4
H1
H2
H3
A
B
AG = site access gateway
= host
H1–4 = Ethernet switching repeater hub

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