14

Routing Within An Autonomous System (RIP, RIPng, OSPF, IS-IS)

14.1 Introduction

The previous chapter introduces the autonomous system concept and examines BGP, an Exterior Gateway Protocol that a router uses to advertise networks within its system to other autonomous systems. This chapter completes our overview of internet routing by examining how a router in an autonomous system learns about other networks within its autonomous system.

14.2 Static Vs. Dynamic Interior Routes

Two routers within an autonomous system are said to be *interior* to one another. For example, two routers on a university campus are considered interior to one another as long as machines on the campus are collected into a single autonomous system.

How can routers in an autonomous system learn about networks within the autonomous system? In the smallest intranets, network managers can establish and modify routes manually. The manager keeps a list of networks and updates the forwarding tables whenever a new network is added to, or deleted from, the autonomous system. For example, consider the small corporate intranet shown in Figure 14.1.

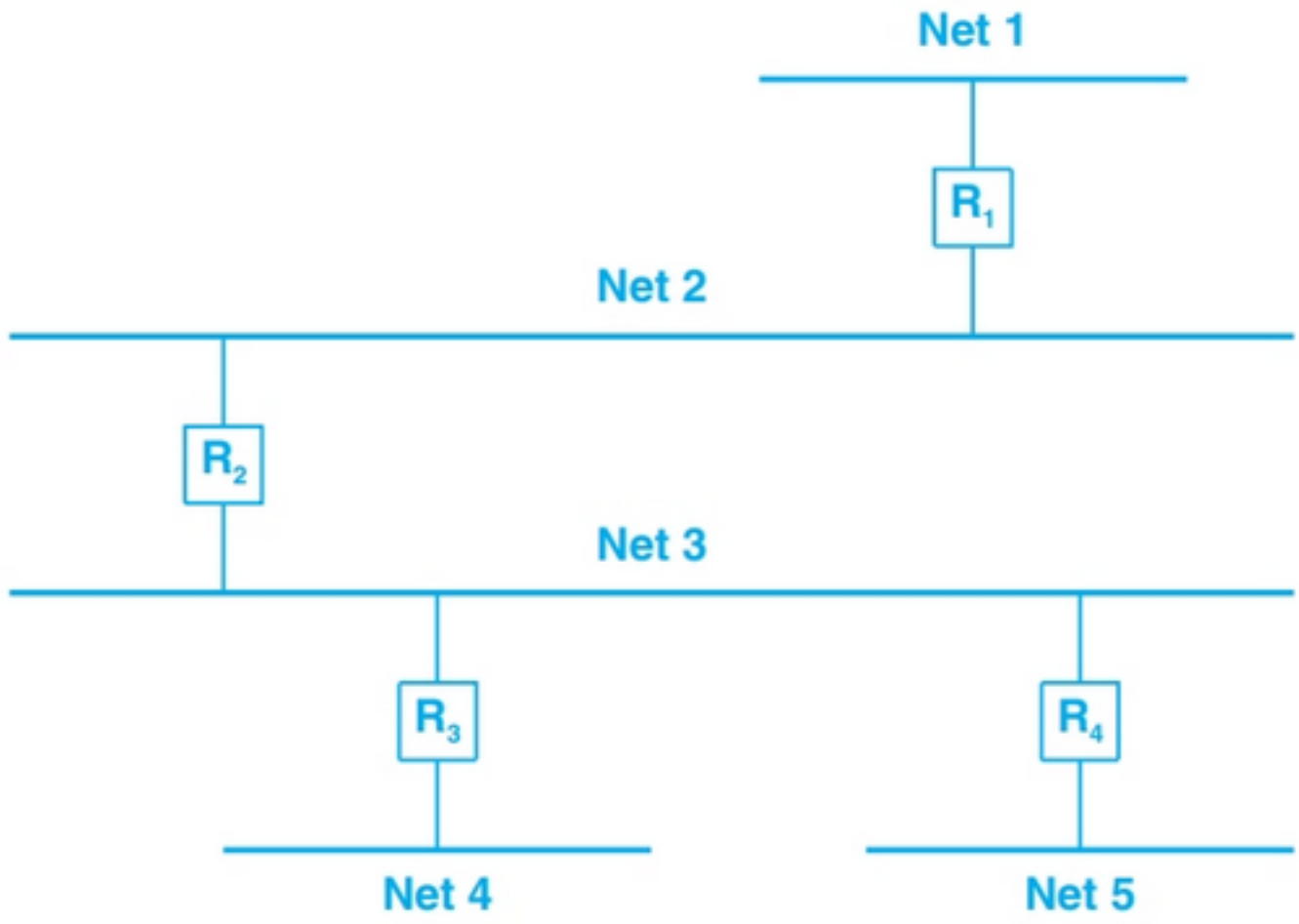

Figure 14.1 An example of a small intranet consisting of five networks and four routers. Only one possible route exists between any two hosts in the example.

Routing for the intranet in the figure is trivial because only one path exists between any two points. If a network or router fails, the intranet will be disconnected because there are no redundant paths. Therefore, a manager can configure routes in all hosts and routers manually, and never needs to change the routes. Of course, if the intranet changes (e.g., a new network is added), the manager must reconfigure the routes accordingly.

The disadvantages of a manual system are obvious: manual systems cannot accommodate rapid growth and rely on humans to change routes whenever a network failure occurs. In most intranets, humans simply cannot respond to changes fast enough to handle problems; automated methods must be used. To understand how automated routing can increase reliability, consider what happens if we add one additional router to the intranet in Figure 14.1, producing the intranet shown in Figure 14.2.

In the figure, multiple paths exist between some hosts. In such cases, a manager usually chooses one path to be a *primary path* (i.e., the path that will be used for all traffic). If a router or network along the primary path fails, routes must be changed to send traffic along an alternate path. Automated route changes help in two ways. First, because computers can respond to failures much faster than humans, automated route changes are less time consuming. Second, because humans can make small errors when entering network addresses, automated routing is less error-prone. Thus, even in small internets, an automated system is used to change routes quickly and reliably.

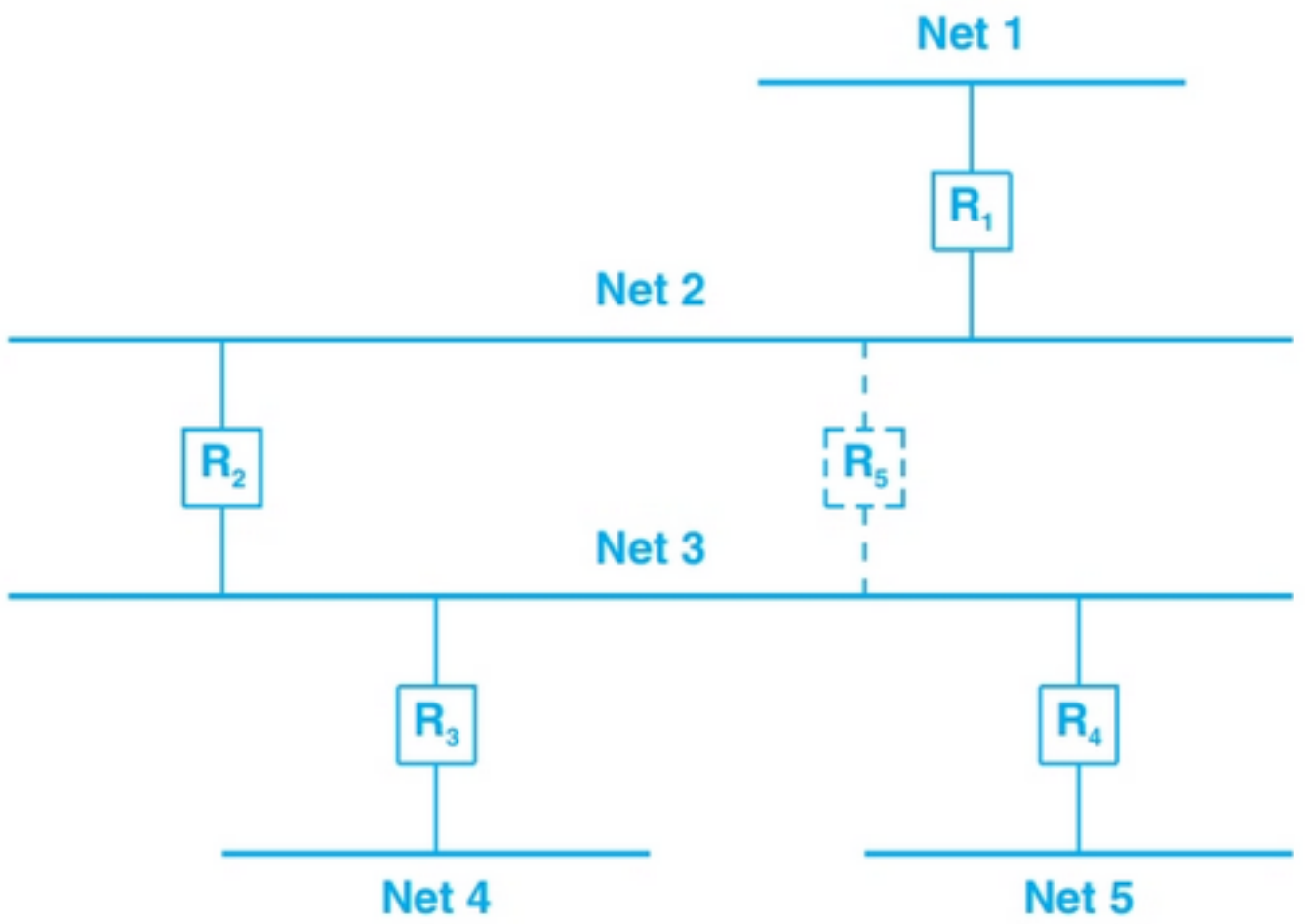

Figure 14.2 The addition of router R_5 introduces an alternate path between networks *2* and *3*. Routing software can quickly adapt to a failure and automatically switch routes to the alternate path.

To automate the task of keeping routing information accurate, interior routers periodically communicate with one another to exchange routing information. Unlike exterior router communication, for which BGP provides a widely accepted standard, no single protocol has emerged for use within an autonomous system or a site. Part of the reason for diversity arises from the diversity in autonomous systems. Some autonomous systems correspond to a large enterprise (e.g., a corporation) at a single site, while others correspond to an organization with many sites connected by a wide area network. Even if we consider individual Internet sites, the network topologies (e.g., degree of redundancy), sizes, and network technologies vary widely. Another reason for diversity of interior routing protocols stems from the tradeoffs between ease of configuration, functionality, and traffic the protocols impose on the underlying networks — protocols that are easy to install and configure may not provide the functionality needed or may impose intolerable load on the networks. As a result, a handful of protocols have become popular, but no single protocol is always optimal.

Although multiple interior protocols are used, a given autonomous system often chooses to limit the number of protocols that are deployed. A small AS tends to choose a single protocol and use it exclusively to propagate routing information internally. Even larger autonomous systems tend to choose a small set. There are two reasons. First, one of the most complex aspects of routing arises from the interaction of protocols. If protocol A is used on some routers and protocol B is used on other routers, at least one router between the two sets must communicate using both protocols and must have a way to transfer information between them. Such interactions are complex, and care must be taken or differences in protocols can lead to unexpected consequences. Second, because routing protocols are difficult to understand and configure, each auton-

omous system must have a staff that is trained to install, configure, and support each individual protocol, as well as software that handles interactions among them. Training can be expensive, so limiting the number of protocols can reduce costs.

We use the term *Interior Gateway Protocol* (*IGP*) as a generic description that refers to any protocol that interior routers use when they exchange routing information. Figure 14.3 illustrates the general idea: two autonomous systems each use a specific IGP to propagate routing information among interior routers. The systems then use BGP to summarize information and communicate it to other autonomous systems.

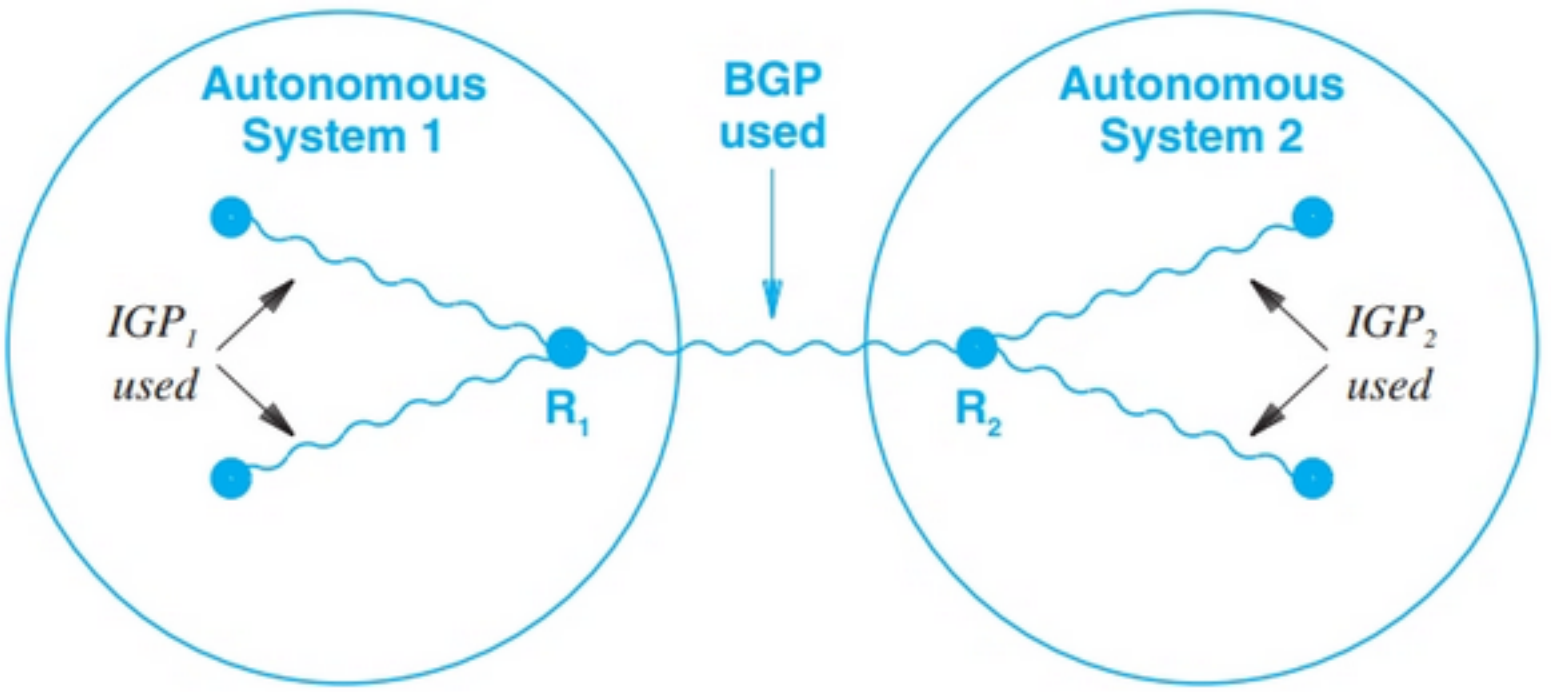

Figure 14.3 Conceptual view of two autonomous systems, each using its own IGP internally, and then using BGP to communicate routes to another system.

In the figure, IGP_1 refers to the interior routing protocol used within autonomous system *1*, and IGP_2 refers to the protocol used within autonomous system 2. Router R_1 will use IGP_1 to obtain routes internally, summarize the information, apply policies, and then use BGP to export the resulting information. Similarly, router R_2 will use IGP_2 to obtain information that it exports. We can summarize the key concept:

> *If multiple routing protocols are used, a single router may run two or more routing protocols simultaneously.*

In particular, routers that run BGP to advertise reachability usually also need to run an IGP to obtain information from within their autonomous system. The next sections describe specific interior gateway protocols; later sections consider some consequences of using multiple protocols.

14.3 Routing Information Protocol (RIP)

14.3.1 History of RIP

The *Routing Information Protocol* (*RIP*) has remained in widespread use since early in the Internet. Originally, RIP was known by the name of an application that implements it, *routed*†. The *routed* software was designed at the University of California at Berkeley to provide consistent routing information among machines on local networks. The protocol was based on earlier research done at Xerox Corporation's Palo Alto Research Center (PARC). The Berkeley version of RIP generalized the PARC version to cover multiple families of networks. RIP relies on physical network broadcast to make routing exchanges quickly, and was not originally designed to be used on large, wide area networks. Vendors later developed versions of RIP suitable for use on WANs.

Despite minor improvements over its predecessors, the popularity of RIP as an IGP does not arise from its technical merits alone. Instead, it is the result of Berkeley distributing *routed* software along with their popular 4BSD UNIX systems. Many early TCP/IP sites adopted and installed RIP without considering its technical merits or limitations. Once installed and running, it became the basis for local routing, and vendors began offering products compatible with RIP.

14.3.2 RIP Operation

The underlying RIP protocol is a straightforward implementation of distance-vector routing for local networks. RIP supports two type of participants: *active* and *passive*. Active participants advertise their routes to others; passive participants listen to RIP messages and use them to update their forwarding table, but do not advertise. Only a router can run RIP in active mode; if a host runs RIP, the host must use passive mode.

A router running RIP in active mode broadcasts a routing update message every 30 seconds. The update contains information taken from the router's current FIB. Each update contains a set of pairs, where each pair specifies an IP network address and an integer distance to that network. RIP uses a *hop count metric* to measure distances. In the RIP metric, a router is defined to be one hop from a directly connected network‡, two hops from a network that is reachable through one other router, and so on. Thus, the *number of hops* or the *hop count* along a path from a given source to a given destination refers to the number of networks that a datagram encounters along that path. It should be obvious that using hop counts to calculate shortest paths does not always produce optimal results. For example, a path with hop count *3* that crosses three Ethernets may be substantially faster than a path with hop count *2* that crosses two satellite connections. To compensate for differences in technologies, many RIP implementations allow managers to configure artificially high hop counts when advertising connections to slow networks.

Both active and passive RIP participants listen to all broadcast messages and update their forwarding tables according to the distance-vector algorithm described in

†The name comes from the UNIX convention of attaching "d" to the names of daemon processes; it is pronounced "route-d".

‡Other routing protocols define a direct connection to be zero hops; we say that RIP uses 1-origin hop counts.

Chapter 12. For example, if the routers in the intranet of Figure 14.2 on page 291 use RIP, router R_1 will broadcast a message on network *2* that contains the pair (*1*, *1*), meaning that it can reach network *1* at distance (i.e., cost) *1*. Routers R_2 and R_5 will receive the broadcast and install a route to network *1* through R_1 (at cost *2*). Later, routers R_2 and R_5 will include the pair (*1*, *2*) when they broadcast their RIP messages on network *3*. Eventually, all routers will install a route to network *1*.

RIP specifies a few rules to improve performance and reliability. For example, once a router learns a route from another router, it must apply *hysteresis*, meaning that it does not replace the route with an equal cost route. In our example, if routers R_2 and R_5 both advertise network *1* at cost *2*, routers R_3 and R_4 will install a route through the one that happens to advertise first. We can summarize:

> *To prevent oscillation among equal cost paths, RIP specifies that existing routes should be retained until a new route has strictly lower cost.*

What happens if a router fails (e.g., the router crashes)? RIP specifies that when a router receives and installs a route in its forwarding table, the router must start a timer for the entry. The timer is reset whenever the router receives another RIP message advertising the same route. The route becomes invalid if 180 seconds pass without the route being advertised again.

RIP must handle three kinds of errors caused by the underlying algorithm. First, because the algorithm does not explicitly detect forwarding loops, RIP must either assume participants can be trusted or take precautions to prevent such loops. Second, to prevent instabilities, RIP must use a low value for the maximum possible distance (RIP uses *16*). Thus, for intranets in which legitimate hop counts approach *16*, managers must divide the intranet into sections or use an alternative protocol†. Third, the distance-vector algorithm used by RIP can create a problem known as *slow convergence* or *count to infinity* in which inconsistencies arise because routing update messages propagate slowly across the network. Choosing a small infinity (*16*) helps limit slow convergence, but does not eliminate it.

14.4 Slow Convergence Problem

Forwarding table inconsistencies and the slow convergence problem are not unique to RIP. They are fundamental problems that can occur with any distance-vector protocol in which update messages carry only pairs of destination network and distance to that network. To understand the problem, consider using a distance-vector protocol on the routers in Figure 14.2 (page 291). To simply the example, we will only consider three routers, R_1, R_2, and R_3, and only consider the routes they have for network *1*. To reach network *1*, R_3 forwards to R_2, and R_2 forwards to R_1. Part (a) of Figure 14.4 illustrates the forwarding.

†Note that the hop count used in RIP measures the *span* of the intranet — the longest distance between two routers — rather than the total number of networks or routers. Most corporate intranets have a span that is much smaller than 16.

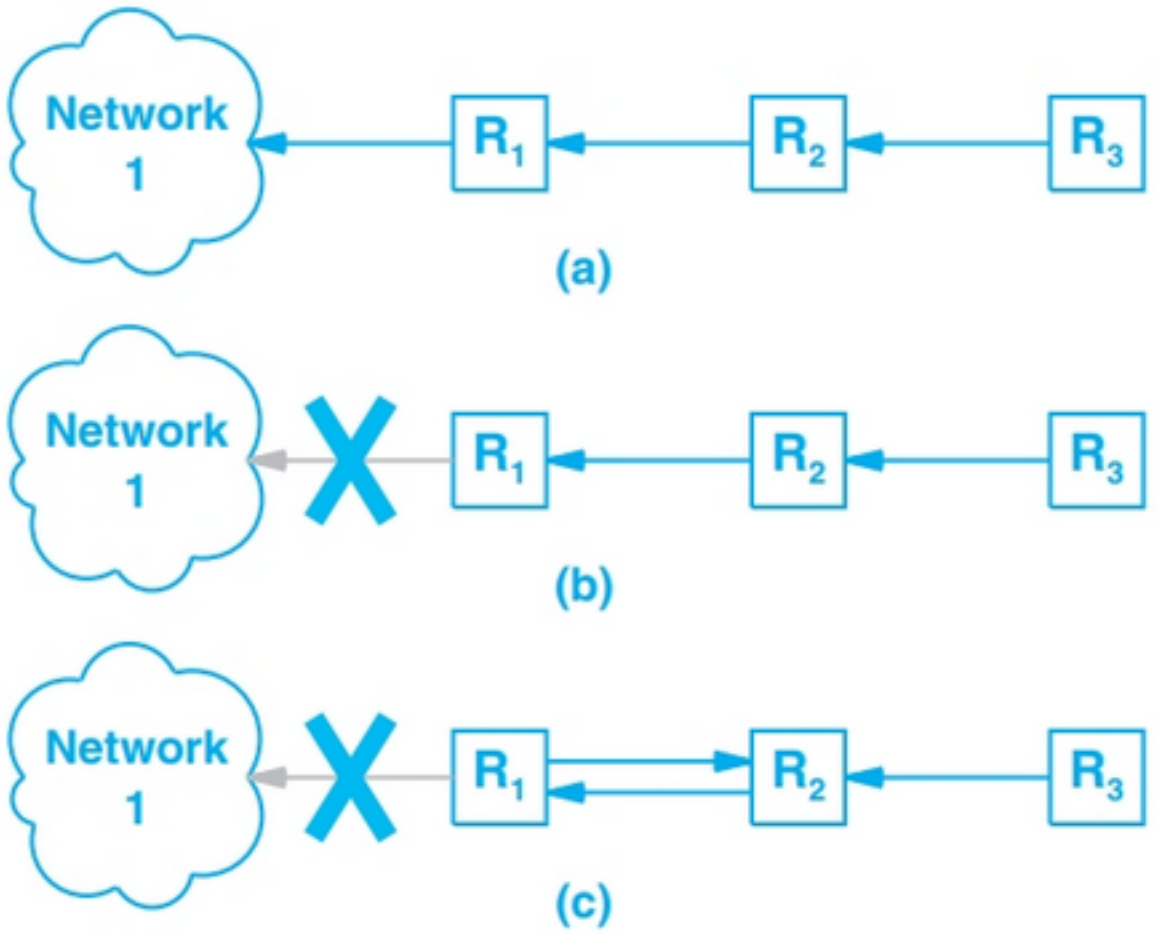

Figure 14.4 Illustration of the slow convergence problem with (a) three routers that have a route to network *1*, (b) the connection to network *1* has failed and R_1 has lost its router, and (c) a routing loop caused because R_2 advertises a route to network 1.

In part (a), we assume all routers are running a distance-vector protocol. We will assume RIP, but the idea applies to any distance-vector protocol. Router R_1 has a direct connection to network *1*. Therefore, when it broadcasts the set of destinations from its FIB, R_1 includes an entry for network *1* at distance *1*. Router R_2 has learned the route from R_1, installed the route in its forwarding table, and advertises the route at distance *2*. Finally, R_3 has learned the route from R_2 and advertises the route at distance *3*.

In part (b) of the figure, we assume a failure has occurred and disconnected R_1 from network 1. Perhaps the connection between router R_1 and network *1* was unplugged or network 1 lost power. The network interface in R_1 will detect the loss of connectivity, and IP will remove the route to network *1* from the forwarding table (or leave the entry, but set the distance to infinity so the route will not be used).

Remember that R_2 broadcasts its routing information periodically. Suppose that immediately after R_1 detects the failure and removes the route from its table, R_2 broadcasts its routing information. Among other items in the broadcast, R_2 will announce a route to network *1* at distance *2*. However, unless the protocol includes extra mechanisms to prevent it, the rules for distance-vector routing mean that R_1 will examine the broadcast from R_2, find a route to network *1*, and add a new route to its table with distance *3* (the distance R_2 advertised plus 1) and R_2 as the next hop.

Unfortunately, part (c) of the figure shows what has happened: R_1 has installed a route for network *1* that goes through R_2, and R_2 has a route that goes through R_1. At this point, if either R_1 or R_2 receives a datagram destined for network *1*, they will route the datagram back and forth until the datagram's hop limit is reached. In other words:

A conventional distance-vector algorithm can form a routing loop after a failure occurs because routing information that a router sent can reach the router again.

The problem persists because the two routers will continue to remain confused about routing. In the next round of routing exchanges, R_1 will broadcast an advertisement that includes the current cost to reach network *1*. When it receives the advertisement from R_1, R_2 will learn that the new distance is *3*. R_2 updates its distance to reach network *1*, making the distance *4*. In the third round, R_1 receives a routing update from R_2 which includes the increased distance. R_1 will increases its distance to *5* and advertise the higher distance in the next update. The two routers continue sending routing update messages back and forth, and the distance increases by *1* on each exchange. Updates continue until the count reaches infinity (16 for RIP).

14.5 Solving The Slow Convergence Problem

A technique known as *split horizon update* has been invented that allows distance-vector protocols, such as RIP, to solve the slow convergence problem. When using split horizon, a router does not propagate information about a route back over the same interface from which the route arrived. In our example, split horizon prevents router R_2 from advertising a route to network *1* back to router R_1, so if R_1 loses connectivity to network *1*, it will stop advertising a route. With split horizon, no forwarding loop appears in the example network. Instead, after a few rounds of routing updates, all routers will agree that the network is unreachable. However, the split horizon heuristic does not prevent forwarding loops in all possible topologies as one of the exercises suggests.

Another way to think of the slow convergence problem is in terms of information flow. If a router advertises a short route to some network, all receiving routers respond quickly to install that route. If a router stops advertising a route, the protocol must depend on a timeout mechanism before it considers the route unreachable. Once the timeout occurs, the router finds an alternative route and starts propagating that information. Unfortunately, a router cannot know if the alternate route depended on the route that just disappeared. Thus, negative information does not always propagate quickly. A short epigram captures the idea and explains the phenomenon:

In routing protocols, good news travels quickly; bad news travels slowly.

Another technique used to solve the slow convergence problem employs *hold down*. Hold down forces a participating router to ignore information about a network for a fixed period of time following receipt of a message that claims the network is unreachable. For RIP, the hold down period is set to 60 seconds, twice as long as a normal update period. The idea is to wait long enough to ensure that all machines receive

the bad news and not mistakenly accept a message that is out of date. It should be noted that all machines participating in a RIP exchange need to use identical notions of hold down, or forwarding loops can occur. The disadvantage of a hold down technique is that if forwarding loops occur, they will be preserved for the duration of the hold down period. More important, the hold down technique preserves all incorrect routes during the hold down period, even when alternatives exist.

A final technique that helps solve the slow convergence problem is called *poison reverse*. Once a connection disappears, the router advertising the connection retains the entry for several update periods, and includes an infinite cost route in its broadcasts. To make poison reverse most effective, it must be combined with *triggered updates*. The triggered update mechanism forces a router to broadcast routing information immediately after receiving bad news. That is, the router does not wait until its next periodic broadcast. By sending an update immediately, a router minimizes the time it is vulnerable (i.e., the time during which neighbors might advertise short routes because they have not received the bad news).

Unfortunately, while triggered updates, poison reverse, hold down, and split horizon techniques all solve some problems, they introduce others. For example, consider what happens with triggered updates when many routers share a common network. A single broadcast may change all their forwarding tables, triggering a new round of broadcasts. If the second round of broadcasts changes tables, it will trigger even more broadcasts. A broadcast avalanche can result†.

The use of broadcast, potential for forwarding loops, and use of hold down to prevent slow convergence can make RIP extremely inefficient in a wide area network. Broadcasting always takes substantial bandwidth. Even if no avalanche problems occur, having all machines broadcast periodically means that the traffic increases as the number of routers increases. The potential for forwarding loops can also be deadly when line capacity is limited. Once lines become saturated by looping packets, it may be difficult or impossible for routers to exchange the routing messages needed to break the cycle. Also, in a wide area network, hold down periods are so long that the timers used by higher-level protocols can expire and lead to broken connections.

14.6 RIP Message Format (IPv4)

RIP messages can be broadly classified into two types: routing information messages and messages used to request information. Both use the same format, which consists of a fixed header followed by an optional list of network and distance pairs. Figure 14.5 shows the message format used with RIP version 2 (*RIP2*), the current version.

In the figure, field *COMMAND* specifies an operation; only five commands are defined. Figure 14.6 lists the commands and the meaning of each.

†To help avoid an avalanche, RIP requires each router to wait a small random time before sending a triggered update.

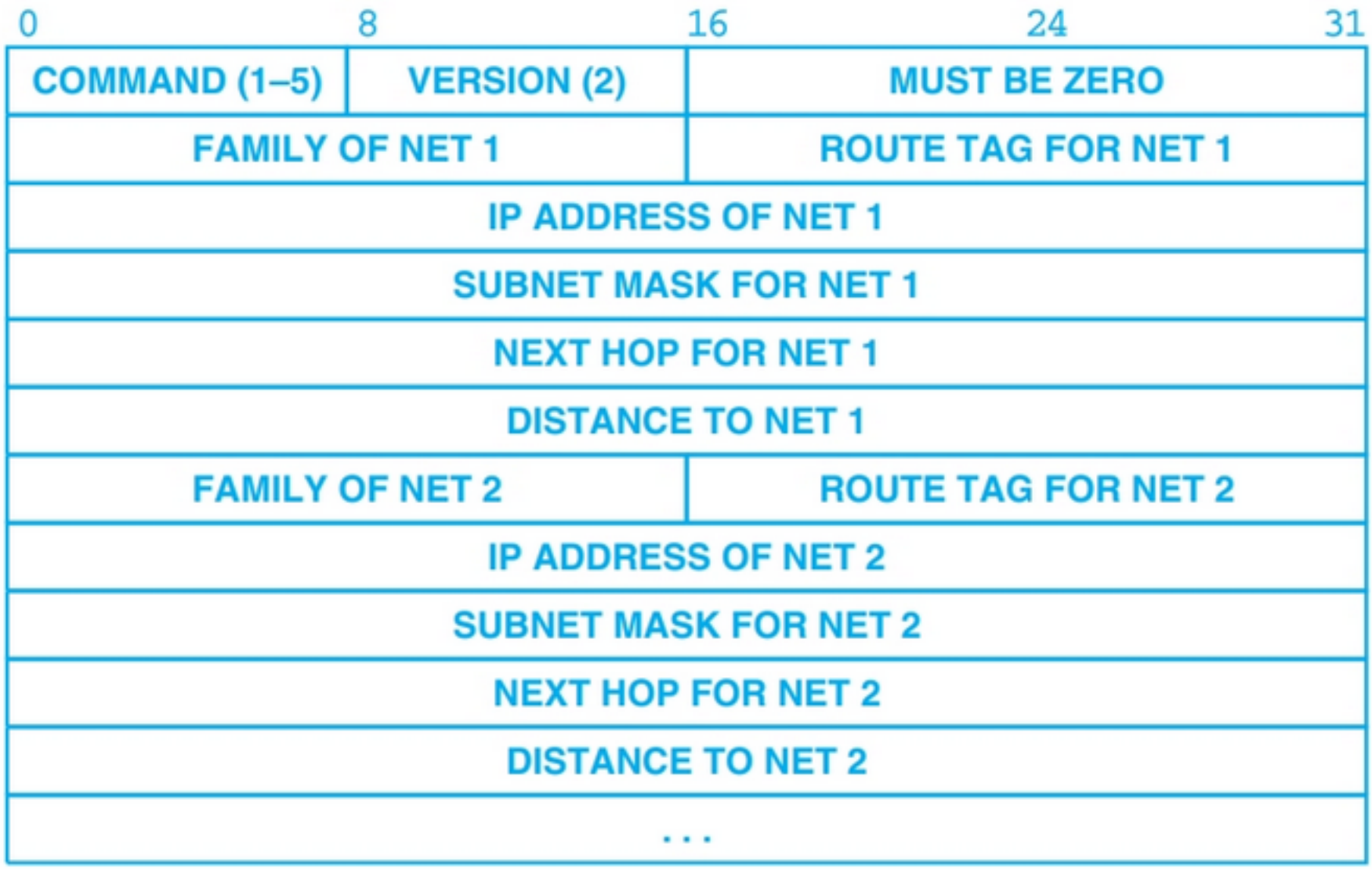

Figure 14.5 The format of an IPv4 version 2 RIP message. After the 32-bit header, the message contains a sequence of pairs, where each pair specifies an IPv4 prefix, next hop, and distance to the destination.

Command	Meaning
1	Request for partial or full routing information
2	Response containing network-distance pairs from sender's forwarding information base
9	Update Request (used with demand circuits)
10	Update Response (used with demand circuits)
11	Update Acknowledge (used with demand circuits)

Figure 14.6 The commands used with RIP. In typical implementations, only command 2 is used.

Although we have described RIP as sending routing updates periodically, the protocol includes commands that permit queries to be sent. For example, a host or router can send a *request* command to request routing information. Routers can use the *response* command to reply to requests. In most cases, a router periodically broadcasts unsolicited response messages. Field *VERSION* in Figure 14.5 contains the protocol version number (*2* in this case), and is used by the receiver to verify it will interpret the message correctly.

14.7 Fields In A RIP Message

Because it was initially used with addresses other than IPv4, RIP uses a 16-bit *FAMILY OF NET* field to specify the type of the address that follows. Values for the field were adopted values from 4.3 BSD Unix; IPv4 addresses are assigned family *2*. Two fields in each entry specify an IPv4 prefix: *IP ADDRESS OF NET* and *SUBNET MASK FOR NET*. As expected, the mask specifies which bits in the address correspond to the prefix. The *NEXT HOP FOR NET* field specifies the address of a router that is the next hop for the route. The last field of each entry in a RIP message, *DISTANCE TO NET*, contains an integer count of the distance to the specified network. As discussed above, RIP uses 1-origin routes, which means a directly connected network is one hop away. Furthermore, because RIP interprets *16* as infinity (i.e., no route exists), all distances are limited to the range *1* through *16*. Surprisingly, the distance is assigned a 32-bit field, even though only the low-order five bits are used.

RIP2 attaches a 16-bit *ROUTE TAG FOR NET* field to each entry. A router must send the same tag it receives when it transmits the route. Thus, the route tag provides a way to propagate additional information such as the origin of the route. In particular, if RIP2 learns a route from another autonomous system, it can use the route tag to propagate the autonomous system's number.

In addition to unicast IPv4 addresses, RIP uses the convention that a zero address (e.g., *0.0.0.0*), denotes a *default route*. RIP attaches a distance metric to every route it advertises, including the default route. Thus, it is possible to arrange for two routers to advertise a default route (e.g., a route to the rest of the internet) at different metrics, making one of them a primary path and the other a backup.

To prevent RIP from increasing the CPU load of hosts unnecessarily, the designers allow RIP2 to multicast updates instead of broadcasting them. Furthermore, RIP2 is assigned a fixed multicast address, 224.0.0.9, which means that machines using RIP2 do not need to run IGMP†. Finally, RIP2 multicast is restricted to a single network.

Note that a RIP message does not contain an explicit length field or an explicit count of entries. Instead, RIP assumes that the underlying delivery mechanism will tell the receiver the length of an incoming message. In particular, when used with TCP/IP, RIP messages rely on UDP to tell the receiver the message length. RIP operates on UDP port *520*. Although a RIP request can originate at other UDP ports, the destination UDP port for requests is always *520*, as is the source port from which RIP broadcast messages originate.

14.8 RIP For IPv6 (RIPng)

It may seem that the *FAMILY OF NET* field in the original design permits arbitrary protocol addresses to be used. However, instead of merely using the existing design, a new version of RIP was created for IPv6. Called *RIPng*‡, the new protocol has an entirely new message format and even operates on a different UDP port than RIP (port *521* as opposed to port *520*). Figure 14.7 illustrates the format.

---

†Chapter 15 describes the *Internet Group Management Protocol*.

‡The suffix *ng* stands for "next generation"; IPv6 was initially named *IPng*.

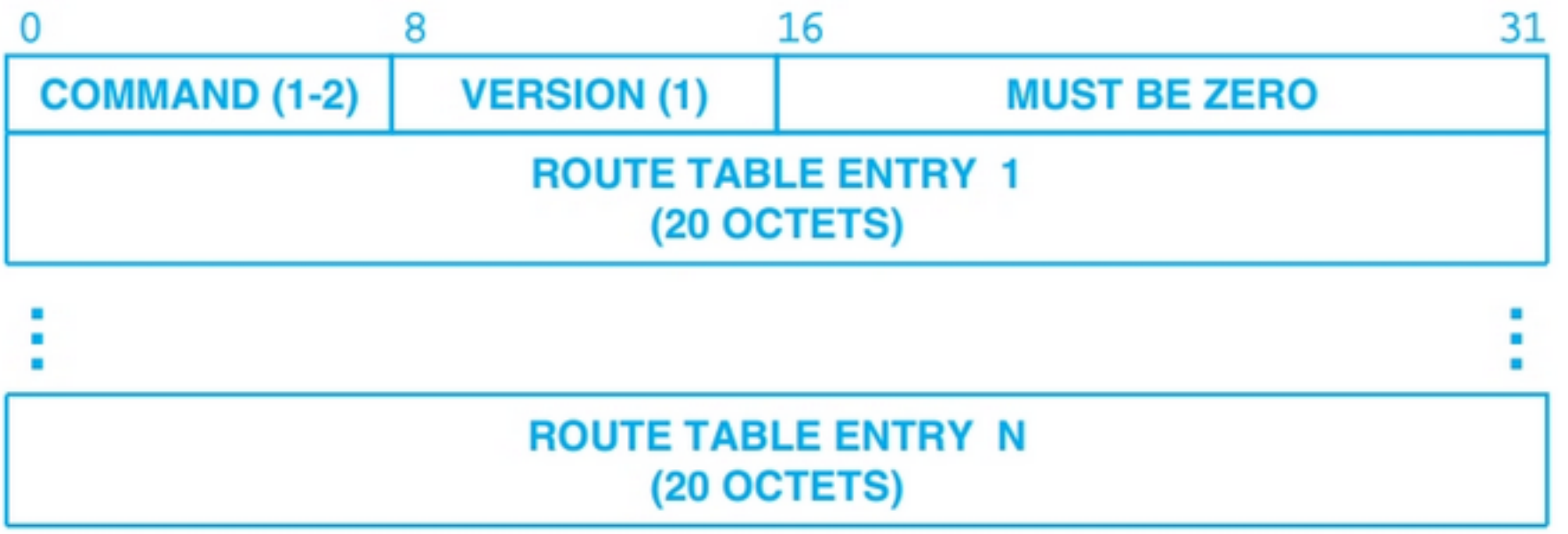

Figure 14.7 The overall format of a RIPng message used to carry IPv6 routing information.

Like RIP2, a RIPng message does not include a size field, nor does it include a count of items that follow; a receiver computes the number of route table entries from the size of the packet (which is obtained from UDP). As the figure indicates, each *ROUTE TABLE ENTRY* occupies 20 octets (i.e., the figure is not drawn to scale). Figure 14.8 illustrates the format of an individual route table entry.

Figure 14.8 The format of each ROUTE TABLE ENTRY in a RIPng message

Observant readers may have noticed that RIPng does not include a field that stores the next hop for a route. The designers were aware that including a next hop in each route table entry would make the message size extremely large. Therefore, they chose an alternative: a route table entry with a metric field of 0xFF specifies a next hop rather than a destination. The next hop applies to all the route table entries that follow until another next hop entry or the end of the message.

Other than the new message format, the use of IPv6 addresses, and the special provision for a next hop, RIPng resembles RIP. Messages are still sent via UDP, and

RIPng still transmits a routing update every 30 seconds and uses a timeout of 180 seconds before considering a route expired. RIPng also preserves the techniques of split horizon, poison reverse, and triggered updates.

14.9 The Disadvantage Of Using Hop Counts

Using RIP or RIPng as an interior gateway protocol restricts routing to a hop-count metric. Even in the best cases, hop counts provide only a crude measure of network capacity or responsiveness. We know that using hop counts does not always yield routes with least delay or highest capacity. Furthermore, computing routes on the basis of minimum hop counts has the severe disadvantage that it makes routing relatively static because routes cannot respond to changes in network load. Therefore, it may seem odd that a protocol would be designed to use a hop-count metric. The next sections consider an alternative metric and explain why hop count metrics remain popular despite their limitations.

14.10 Delay Metric (HELLO)

Although now obsolete, the HELLO protocol provides an example of an IGP that was once deployed in the Internet and uses a routing metric other than hop count. Each HELLO message carried timestamp information as well as routing information, which allowed routers using HELLO to synchronize their clocks. Interestingly, HELLO used the synchronized clocks to find the delay on the link between each pair of routers so that each router could compute shortest delay paths to all destinations.

The basic idea behind HELLO is simple: use a distance-vector algorithm to propagate routing information. Instead of having routers report a hop count, however, HELLO reports an estimate of the delay to the destination. Having synchronized clocks allows a router to estimate delay by placing a timestamp on each packet. Before sending a packet, the sender places the current clock value in the packet as a timestamp, and the receiver subtracts the value from the current clock value. Having synchronized clocks allows delay to be computed without relying on round-trip samples, which means the delay in each direction can be estimated independently (i.e., congestion in one direction will not affect the estimated delay in the other direction).

HELLO uses the standard distance-vector approach for updates. When a message arrives from machine X, the receiver examines each entry in the message and changes the next hop to X if the route through X is less expensive than the current route (i.e., the delay to X plus the delay from X to the destination is less than the current delay to the destination).

14.11 Delay Metrics, Oscillation, And Route Flapping

It may seem that using delay as a routing metric would produce better routes than using a hop count and that all routing protocols should adopt a delay metric. In fact, HELLO worked well in the early Internet backbone. However, there is an important reason why delay is not used as a metric in current protocols: instability.

Even if two paths have identical characteristics, any protocol that changes routes quickly can become unstable. Instability arises because delay, unlike hop counts, is not static. Minor variations in delay measurements occur because of hardware clock drift, CPU load during measurement, or bit delays caused by link-level synchronization. Thus, if a routing protocol reacts quickly to slight differences in delay, it can produce a two-stage oscillation effect in which traffic switches back and forth between the alternate paths. In the first stage, the router finds the delay on path *1* slightly less and abruptly switches traffic onto it. In the next round, the router finds that changing the load to path *1* has increased the delay and that path *2* now has slightly less delay. So, the router switches traffic back to path *2* and the situation repeats.

To help avoid oscillation, protocols that use delay implement several heuristics. First, they employ the *hold down* technique discussed previously to prevent routes from changing rapidly. Second, instead of measuring as accurately as possible and comparing the values directly, the protocols round measurements to large multiples or implement a minimum *threshold* by ignoring differences less than the threshold. Third, instead of comparing each individual delay measurement, they keep a running *average* of recent values or alternatively apply a *K-out-of-N* rule that requires at least *K* of the most recent *N* delay measurements be less than the current delay before the route can be changed.

Even with heuristics, protocols that use delay can become unstable when comparing delays on paths that do not have identical characteristics. To understand why, it is necessary to know that traffic can have a dramatic effect on delay. With no traffic, the network delay is simply the time required for the hardware to transfer bits from one point to another. As the traffic load imposed on the network increases, delays begin to rise because routers in the system need to enqueue packets that are waiting for transmission. If the load is even slightly more than 100% of the network capacity, the queue becomes unbounded, meaning that the effective delay becomes infinite. To summarize:

> *The effective delay across a network depends on traffic; as the load approaches 100% of the network capacity, delay grows rapidly.*

Because delays are extremely sensitive to changes in load, protocols that use delay as a metric can easily fall into a *positive feedback cycle*. The cycle is triggered by a small external change in load (e.g., one computer injecting a burst of additional traffic). The increased traffic raises the delay, which causes the protocol to change routes. However, because a route change affects the load, it can produce an even larger change in

delays, which means the protocol will again recompute routes. As a result, protocols that use delay must contain mechanisms to dampen oscillation.

We described heuristics that can solve simple cases of route oscillation when paths have identical throughput characteristics and the load is not excessive. The heuristics can become ineffective, however, when alternative paths have different delay and throughput characteristics. As an example consider the delay on two paths: one over a satellite and the other over a low-capacity digital circuit (e.g., a fractional T1 circuit). In the first stage of the protocol when both paths are idle, the digital circuit will appear to have significantly lower delay than the satellite, and will be chosen for traffic. Because the circuit has low capacity, it will quickly become overloaded, and the delay will rise sharply. In the second stage, the delay on the circuit will be much greater than that of the satellite, so the protocol will switch traffic away from the overloaded path. Because the satellite path has large capacity, traffic which overloaded the serial line does not impose a significant load on the satellite, meaning that the delay on the satellite path does not change with traffic. In the next round, the delay on the unloaded digital circuit will once again appear to be much smaller than the delay on the satellite path. The protocol will reverse the routing, and the cycle will continue. We say that the routes *flap* back and forth. Such oscillations do, in fact, occur in practice. As the example shows, they are difficult to manage because traffic which has little effect on one network can overload another. The point is:

> *Although intuition suggests that routing should use paths with lowest delay, doing so makes routing subject to oscillations known as route flapping.*

14.12 The Open SPF Protocol (OSPF)

In Chapter 12, we said that a link-state approach to routing, which employs an SPF graph algorithm to compute shortest paths, scales better than a distance-vector algorithm. To encourage the adoption of link-state technology, a working group of the IETF designed an interior gateway protocol that uses the link-state algorithm. Named *Open SPF* (*OSPF*), the protocol tackles several ambitious goals.

- *Open Standard*. As the name implies, the specification is available in the published literature. Making it an open standard that anyone can implement without paying license fees has encouraged many vendors to support OSPF.

- *Type Of Service Routing*. Managers can install multiple routes to a given destination, one for each priority or type of service. A router running OSPF can use both the destination address and type of service when choosing a route.

- *Load Balancing*. If a manager specifies multiple routes to a given destination at the same cost, OSPF distributes traffic over all routes equally.

- *Hierarchical Subdivision Into Areas.* To handle large intranets and limit routing overhead, OSPF allows a site to partition its networks and routers into subsets called *areas*. Each area is self-contained; knowledge of an area's topology remains hidden from other areas. Thus, multiple groups within a given site can cooperate in the use of OSPF for routing even though each group retains the ability to change its internal network topology independently.

- *Support For Authentication.* OSPF allows all exchanges between routers to be *authenticated*. OSPF supports a variety of authentication schemes, and allows one area to choose a different scheme than another area.

- *Arbitrary Granularity.* OSPF includes support for host-specific, subnet-specific, network-specific, and default routes.

- *Support For Multi-Access Networks.* To accommodate networks such as Ethernet, OSPF extends the SPF algorithm. Normally, SPF requires each pair of routers to broadcast messages about the link between them. If K routers attach to an Ethernet, they will broadcast K^2 status messages. Instead of a graph that uses point-to-point connections, OSPF reduces broadcasts by allowing a more complex graph topology in which each node represents either a router or a network. A *designated gateway* (i.e., a *designated router*) sends link-status messages on behalf of all routers attached to the network.

- *Multicast Delivery.* To reduce the load on nonparticipating systems, OSPF uses hardware multicast capabilities, where they exist, to deliver link-status messages. OSPF sends messages via IP multicast, and allows the IP multicast mechanism to map the multicast into the underlying network; two IPv4 multicast addresses are preassigned to OSPF 224.0.0.5 for all routers and 224.0.0.6 for all nodes.

- *Virtual Topology.* A manager can create a virtual network topology. For example, a manager can configure a virtual link between two routers in the routing graph even if the physical connection between the two routers requires communication across multiple transit networks.

- *Route Importation.* OSPF can import and disseminate routing information learned from external sites (i.e., from routers that do not use OSPF). OSPF messages distinguish between information acquired from external sources and information acquired from routers interior to the site.

- *Direct Use Of IP.* Unlike RIP and RIPng, OSPF messages are encapsulated directly in IP datagrams. The value *89* is used in the *PROTO* field (IPv4) or the *NEXT HOP* field (IPv6) in the header to identify the datagram is carrying OSPF.

14.13 OSPFv2 Message Formats (IPv4)

Currently, the standard version of OSPF is version 2. Version 2 was created for IPv4 and cannot handle IPv6. Unlike RIP, where the IETF chose to create an entirely new protocol for IPv6, an IETF working group has proposed that the changes in OSPFv2 for IPv6 merely be incorporated in a new version of OSPF, version 3. We will first examine the version 2 message formats used with IPv4, and then consider the version 3 message formats used with IPv6. To distinguish between them, we will write *OSPFv2* and *OSPFv3*.

Each OSPFv2 message begins with a fixed, 24-octet header. Figure 14.9 illustrates the format.

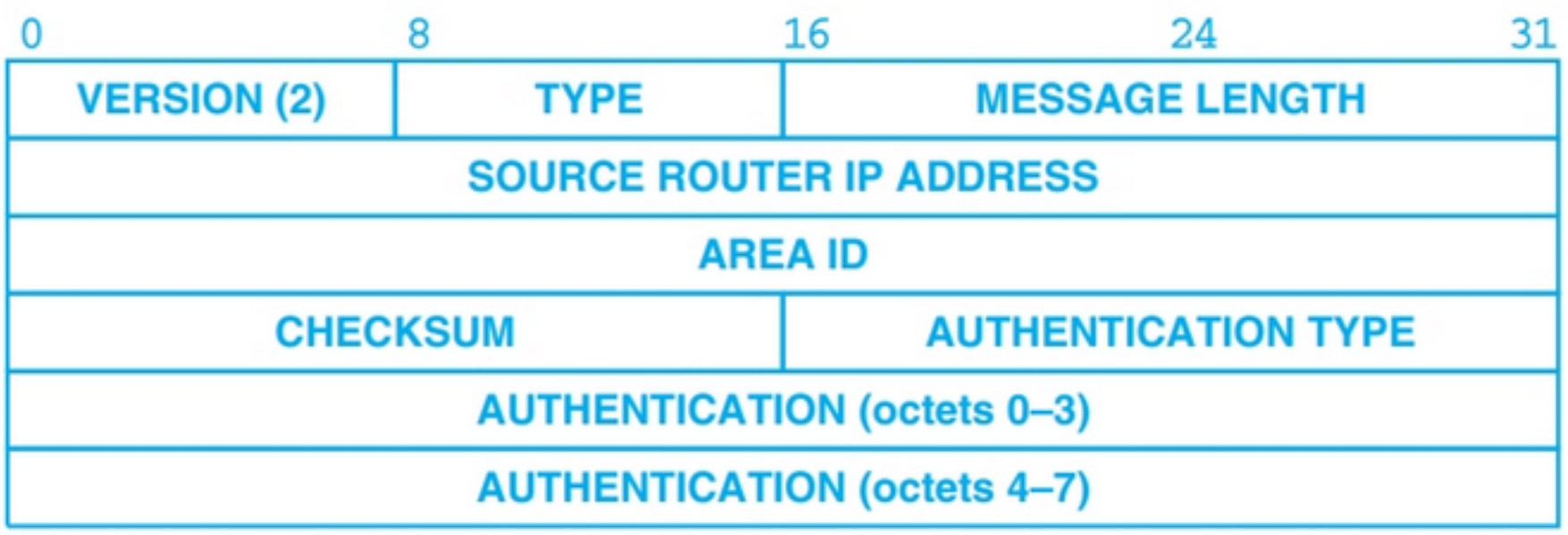

Figure 14.9 The fixed 24-octet OSPFv2 header that appears in each message.

Field *VERSION* specifies the version of the protocol as *2*. Field *TYPE* identifies the message type as one of the following types (which are explained in the next sections):

Type	Meaning
1	Hello (used to test reachability)
2	Database description (topology)
3	Link-status request
4	Link-status update
5	Link-status acknowledgement

The field labeled *SOURCE ROUTER IP ADDRESS* gives the address of the sender, and the field labeled *AREA ID* gives the 32-bit identification number of the area. By convention, Area 0 is the backbone area. Because each message can include authentication, field *AUTHENTICATION TYPE* specifies which authentication scheme is used (e.g., *0* means no authentication and *1* means a simple password is used).

14.13.1 OSPFv2 Hello Message Format

OSPFv2 sends *hello* messages on each link periodically to establish and test neighbor reachability. Figure 14.10 shows the message format. Field *NETWORK MASK* contains an address mask for the network over which the message has been sent. Field *ROUTER DEAD INTERVAL* gives a time in seconds after which a nonresponding neighbor is considered dead. Field *HELLO INTERVAL* is the normal period, in seconds, between hello messages. Field *GWAY PRIO* is the integer priority of this router, and is used in selecting a backup designated router. The fields labeled *DESIGNATED ROUTER* and *BACKUP DESIGNATED ROUTER* contain IP addresses that give the sender's view of the designated router and backup designated router for the network over which the message is sent. Finally, fields labeled *NEIGHBOR IP ADDRESS* give the IP addresses of all neighbors from which the sender has recently received hello messages.

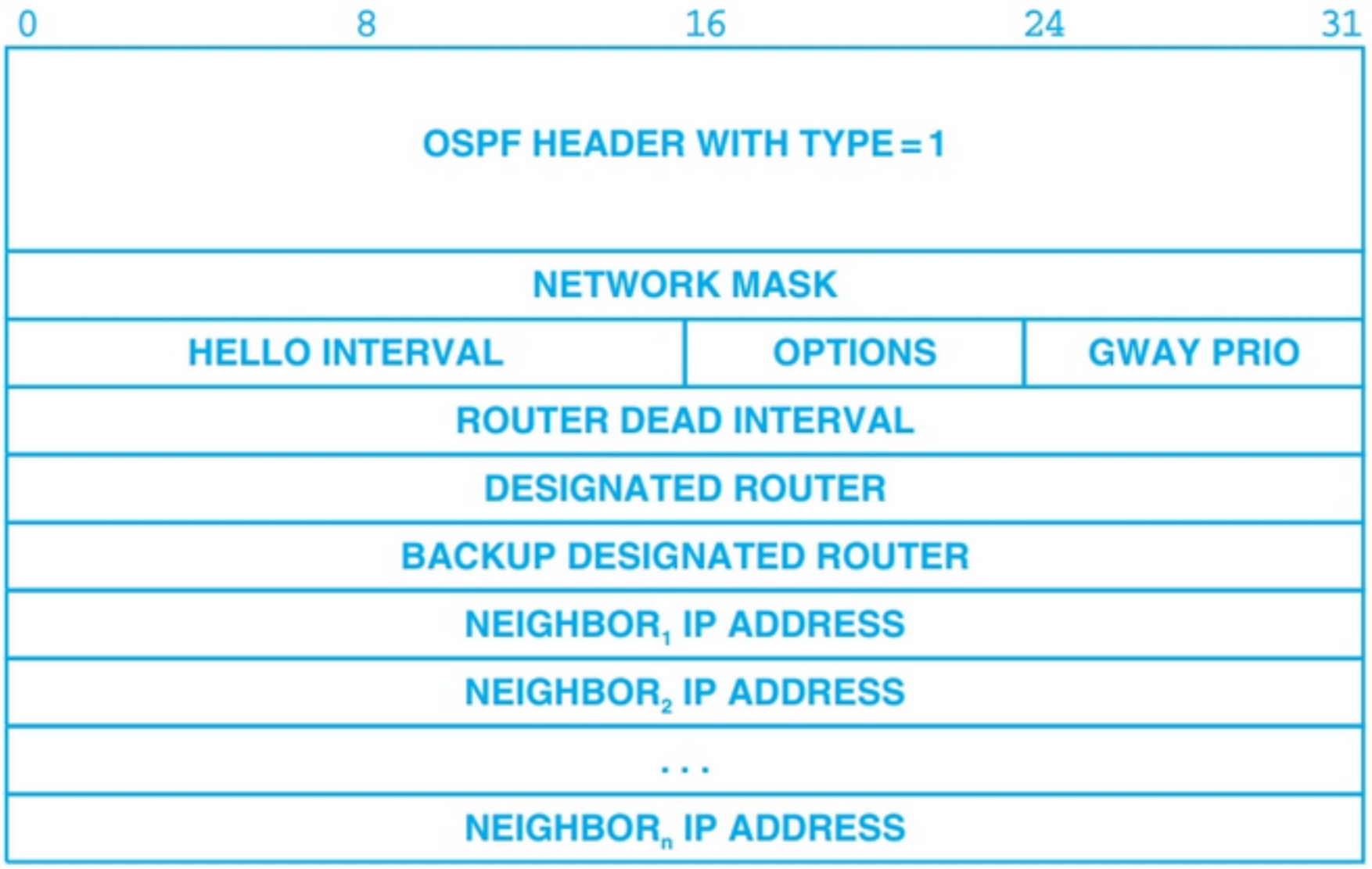

Figure 14.10 The OSPFv2 *hello* message format. A pair of neighbor routers exchanges hello messages periodically to test reachability.

14.13.2 OSPFv2 Database Description Message Format

Routers exchange OSPFv2 *database description* messages to initialize their network topology database. In the exchange, one router serves as a master, while the other is a slave. The slave acknowledges each database description message with a response. Figure 14.11 shows the format.

Because it can be large, a topology database may be divided into multiple messages using the *I* and *M* bits. Bit *I* is set to 1 in the initial message; bit *M* is set to 1 if additional messages follow. Bit *S* indicates whether a message was sent by a master (1) or by a slave (0). Field *DATABASE SEQUENCE NUMBER* numbers messages sequentially so the receiver can tell if one is missing. The initial message contains a random integer *R*; subsequent messages contain sequential integers starting at *R*.

Field *INTERFACE MTU* gives the size of the largest IP datagram that can be transmitted over the interface without fragmentation. The fields from *LS AGE* through *LS LENGTH* describe one link in the network topology; they are repeated for each link.

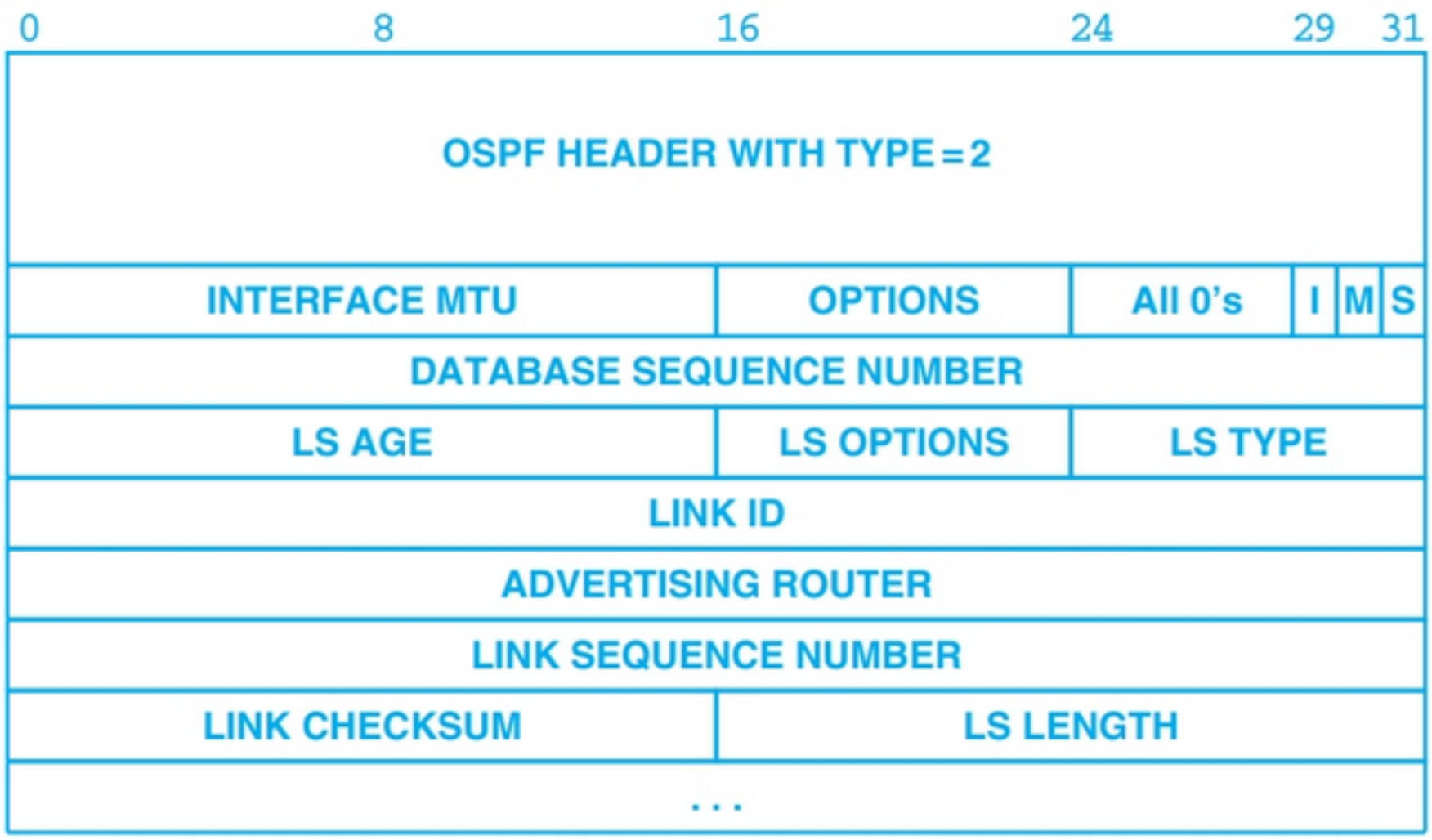

Figure 14.11 OSPFv2 *database description* message format. The fields starting at *LS AGE* are repeated for each link being specified.

Field *LS TYPE* describes the type of a link. The possible values are given by the following table.

LS Type	Meaning
1	Router link
2	Network link
3	Summary link (IP network)
4	Summary link (link to border router)
5	External link (link to another site)

Field *LINK ID* gives an identification for the link (which can be the IP address of a router or a network, depending on the link type).

Field *LS AGE* helps order messages — it gives the time in seconds since the link was established. Field *ADVERTISING ROUTER* specifies the address of the router advertising this link, and *LINK SEQUENCE NUMBER* contains an integer generated by that router to ensure that messages are not missed or received out of order. Field *LINK CHECKSUM* provides further assurance that the link information has not been corrupted.

14.13.3 OSPFv2 Link-Status Request Message Format

After exchanging database description messages with a neighbor, a router has an initial description of the network. However, a router may discover that parts of its database are out of date. To request that the neighbor supply updated information, the router sends a *link-status request* message. The message lists specific links for which information is needed, as shown in Figure 14.12. The neighbor responds with the most current information it has about the links in the request message. The three fields shown in the figure are repeated for each link about which status is requested. More than one request message may be needed if the list of requests is long.

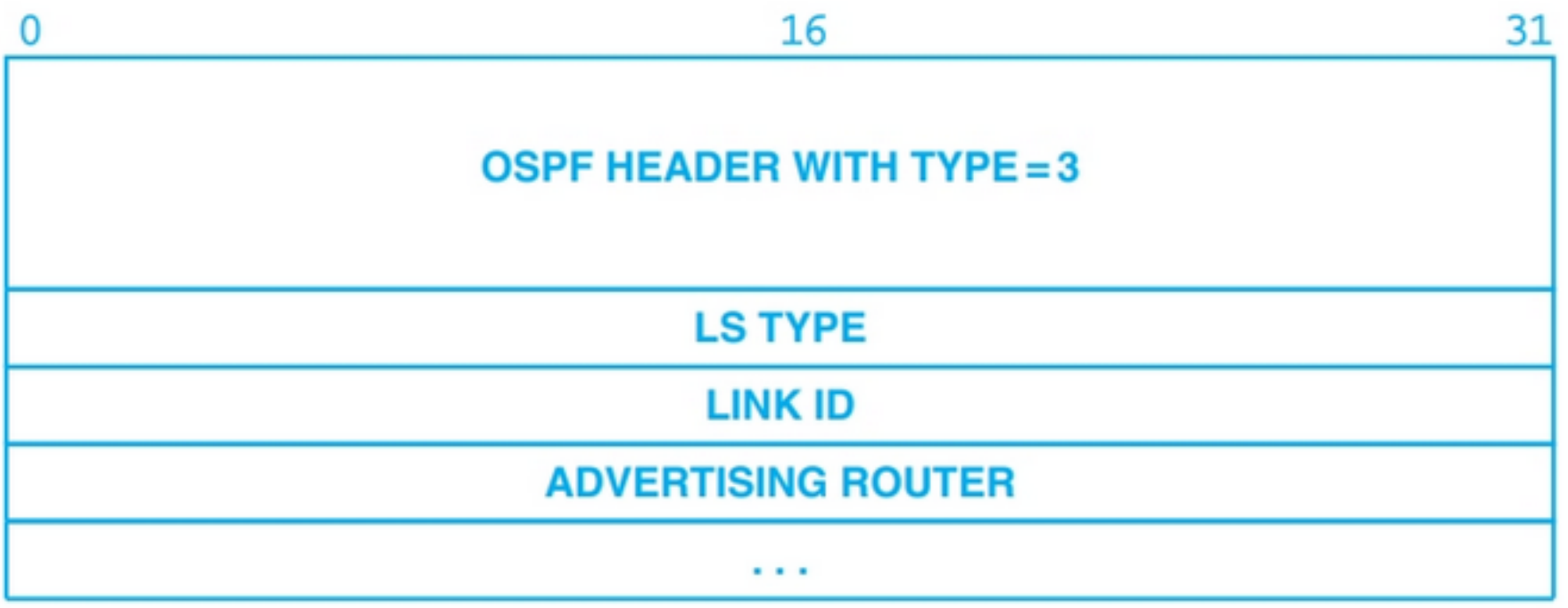

Figure 14.12 OSPFv2 *link-status request* message format. A router sends the message to a neighbor to request current information about a specific set of links.

14.13.4 OSPFv2 Link-Status Update Message Format

Because OSPF uses a link-state algorithm, routers must periodically broadcast messages that specify the status of directly-connected links. To do so, routers use a type *4* OSPFv2 message that is named a *link-status update*. Each update message consists of a count of advertisements followed by a list of advertisements. Figure 14.13 shows the format of link-status update messages.

In the figure, each *link-status advertisement* (*LSA*) has a format that specifies information about the network being advertised. Figure 14.14 shows the format of the link-status advertisement. The values used in each field are the same as in the database description message.

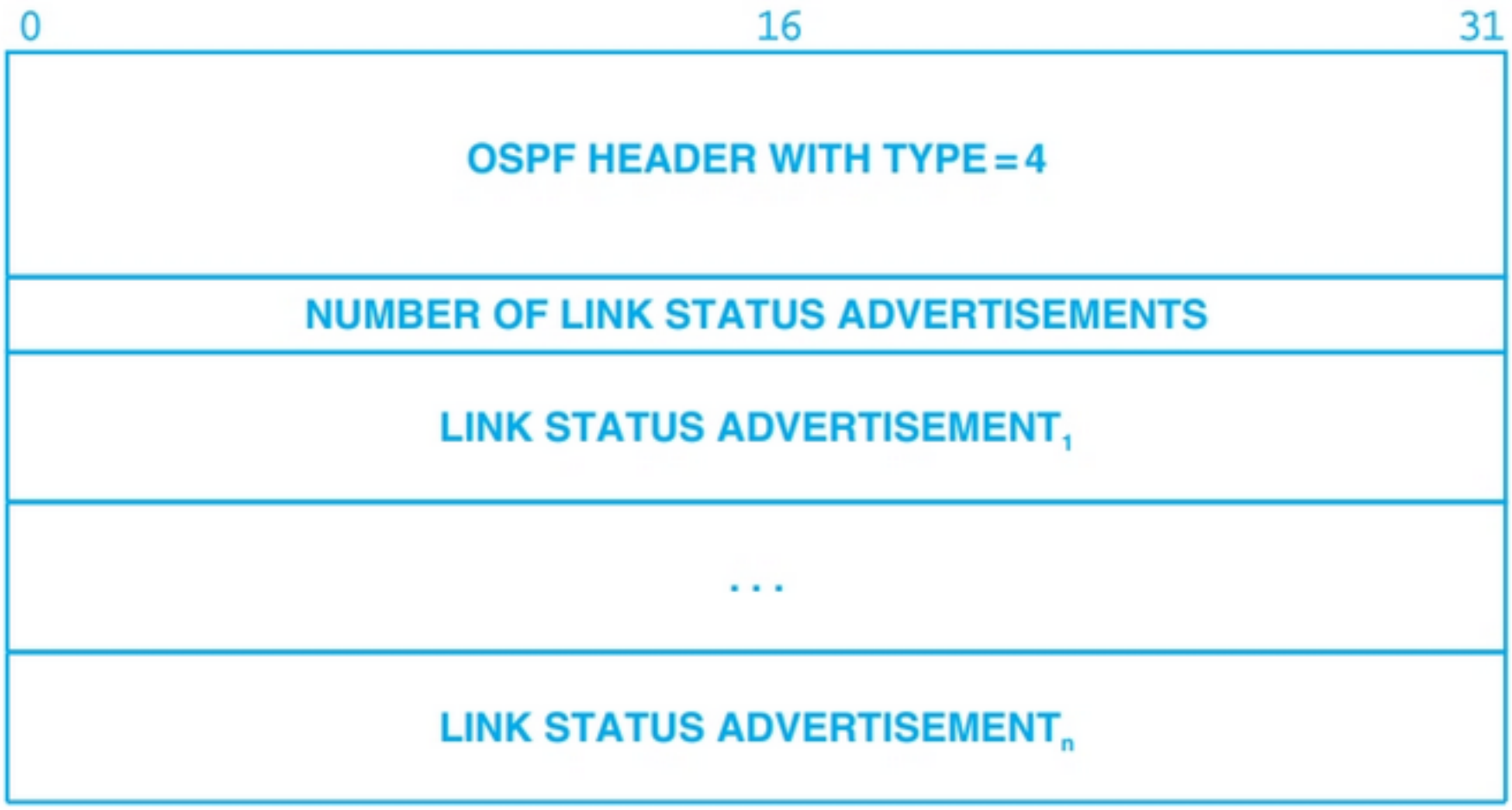

Figure 14.13 OSPFv2 *link-status update* message format. A router sends such a message to broadcast information about its directly connected links to all other routers.

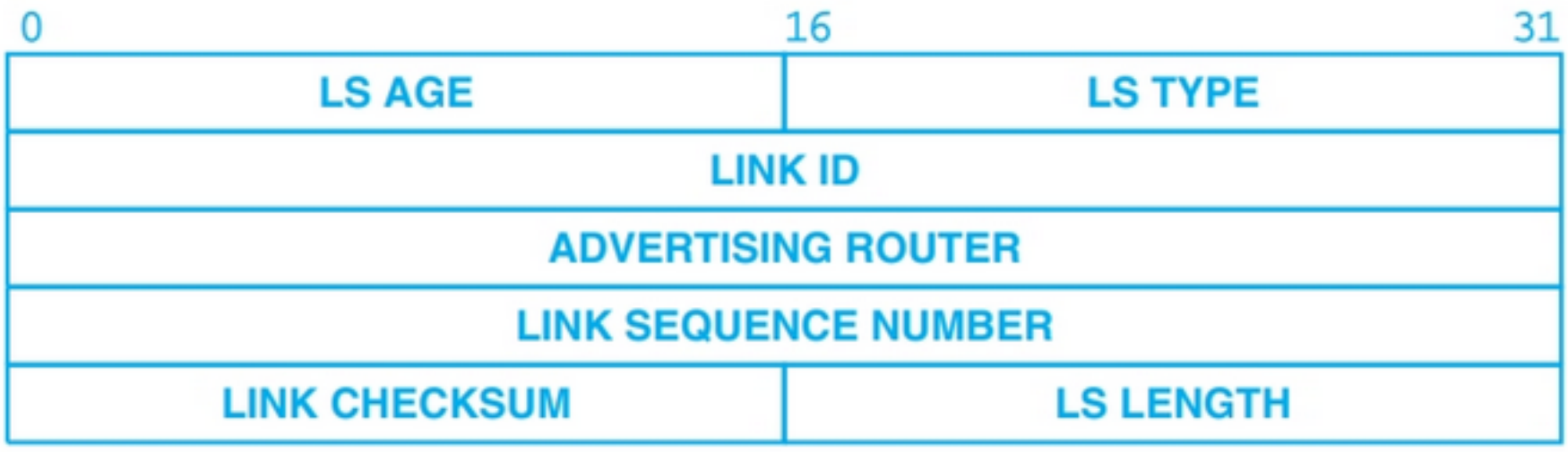

Figure 14.14 The format of an OSPFv2 *link-status advertisement* used in a link-status message.

Following the link-status header comes one of four possible formats to describe the links from a router to a given area, the links from a router to a specific network, the links from a router to the physical networks that constitute a single, subnetted IP network (see Chapter 5), or the links from a router to networks at other sites. In all cases, the *LS TYPE* field in the link-status header specifies which of the formats has been used. Thus, a router that receives a link-status update message knows exactly which of the described destinations lie inside the site and which are external.

14.14 Changes In OSPFv3 To Support IPv6

Although the basics of OSPF remain the same in version 3, many details have changed. The protocol still uses the link-state approach†. All addressing has been removed from the basic protocol, making it protocol-independent except for IP addresses in link-status advertisements. In particular, OSPFv2 used a 32-bit IP address to identify a router; OSPFv3 uses a 32-bit *router ID*. Similarly, area identifiers remain at 32 bits, but are not related to IPv4 addresses (even though dotted decimal is used to express them). OSPFv3 honors IPv6 routing scopes: link-local, area-wide, and AS-wide, meaning that broadcasts will not be propagated beyond the intended set of recipients. OSPFv3 allows independent *instances* of OSPF to run on a set of routers and networks at the same time. Each instance has a unique ID, and packets carry the instance ID. For example, it would be possible to have an instance propagating IPv6 routing information while another instance propagates MPLS routing information. Finally, OSPFv3 removes all authentication from individual messages, and instead, relies on the IPv6 authentication header.

The most significant change between OSPFv2 and OSPFv3 arises from the message formats, which all change. There are two motivations. First, messages must be changed to accommodate IPv6 addresses. Second, because IPv6 addresses are much larger, the designers decided that merely replacing each occurrence of an IPv4 address with an IPv6 address would make messages too large. Therefore, whenever possible, OSPFv3 minimizes the number of IPv6 addresses carried in a message and substitutes 32-bit identifiers for any identifier that does not need to be an IPv6 address.

14.14.1 OSPFv3 Message Formats

Each OSPFv3 message begins with a fixed, 16-octet header. Figure 14.15 illustrates the format.

0	8	16	24 31
VERSION (3)	TYPE	MESSAGE LENGTH	
SOURCE ROUTER ID			
AREA ID			
CHECKSUM		INSTANCE ID	0

Figure 14.15 The fixed 16-octet OSPFv3 header that appears in each message.

Note that the version number occupies the first octet, exactly as in OSPFv2. Therefore, OSPFv3 messages can be sent using the same *NEXT HEADER* value as

†Unfortunately, the terminology has become somewhat ambiguous because IPv6 uses the term *link* in place of *IP subnet* (to permit multiple IPv6 prefixes to be assigned to a given network). In most cases, the IPv6 concept and OSPFv3 concept align, but the distinction can be important in special cases.

OSPFv2 with no ambiguity. Also note that the fixed header is smaller than the OSPFv2 header because authentication information has been removed.

14.14.2 OSPFv3 Hello Message Format

The OSPFv3 *hello* message helps illustrate the basic change from IPv4 addressing to 32-bit identifiers. The goal is to keep the packet size small while separating the protocol from IPv4. As Figure 14.16 illustrates, readers should compare the version 3 format to the version 2 format shown on page 306.

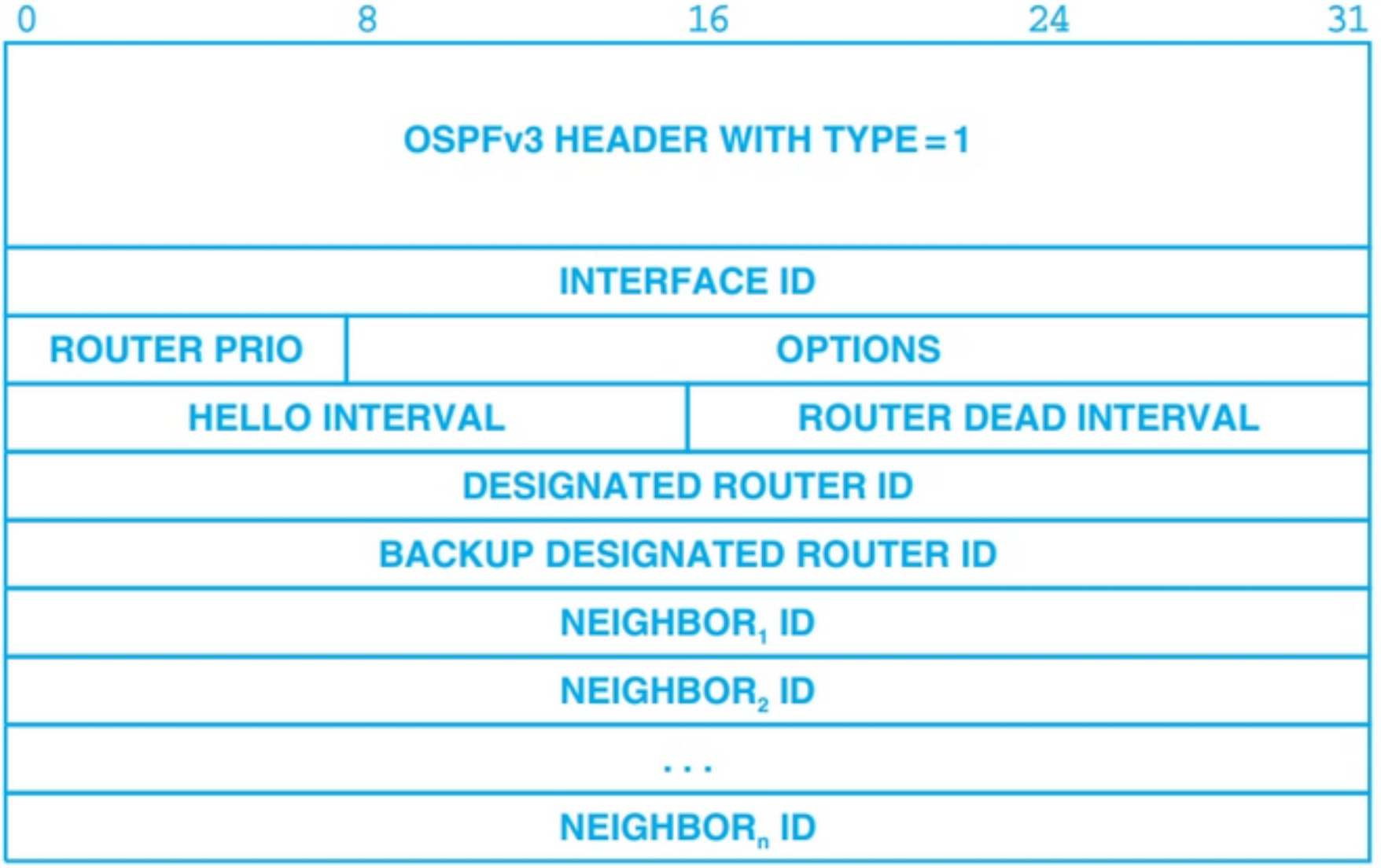

Figure 14.16 The OSPFv3 *hello* message format. All IPv4 addresses have been replaced by 32-bit identifiers.

14.14.3 Other OSPFv3 Features And Messages

OSPFv3 combines and generalizes many of the facilities and features that have been defined for OSPFv2. Consequently, OSPFv3 defines several types of link-status advertisement (LSA). For example, OSPFv3 supports router LSAs, link LSAs, inter-area prefix LSAs, inter-area router LSAs, AS-external LSAa, intra-area prefix LSAs, and *Not So Stubby Area* (*NSSA*) LSAs. Each link-status message begins with a header that is the same as the OSPFv2 header illustrated in Figure 14.14 on page 309, and uses the type field to identify the remaining contents.

The point of providing multiple LSA types is to support large autonomous systems that have a complex topology and complex rules for areas. In particular, Tier-1 provid-

ers use OSPF as an IGP across an intranet that includes a backbone, many regional networks, and many attached networks.

14.15 IS-IS Route Propagation Protocol

About the same time the IETF defined OSPF, Digital Equipment Corporation developed an interior route propagation protocol named *IS-IS*†. IS-IS was part of Digital's *DECnet Phase V* protocol suite, and was later standardized by ISO in 1992 for use in the now-defunct *OSI protocols*. The name expands to *Intermediate System - Intermediate System*, and is equivalent to our definition of an Interior Gateway Protocol.

IS-IS and OSPF are conceptually quite close; only the details differ. Both use the link-state algorithm, both require each participating router to propagate link-status messages for directly-connected routers, and both use incoming link-status messages to build a topology database. Both protocols permit status messages to be multicast if the underlying network supports multicast. Furthermore, both protocols use the Shortest Path First algorithm to compute shortest paths.

Unlike OSPF, IS-IS was not originally designed to handle IP. Therefore, it was later extended, and the extended version is known as *Integrated IS-IS* or *Dual IS-IS*. Because it was extended to handle IP, IS-IS has the advantage of not being integrated with IPv4. Thus, unlike OSPF, which required a new version to handle IPv6, Dual IS-IS accommodates IPv6 as yet another address family. IS-IS also differs from OSPF because IS-IS does *not* use IP for communication. Instead, IS-IS packets are encapsulated in network frames and sent directly over the underlying network.

Like OSPF, IS-IS allows managers to subdivide routers into areas. However, the definition of an area differs from OSPF. In particular, IS-IS does not require an ISP to define Area 0 to be a backbone network through which all traffic flows. Instead, IS-IS defines a router as *Level 1* (intra-area), *Level 2* (inter-area), or *Level 1-2* (both intra-area and inter-area). A Level 1 router only communicates with other Level 1 routers in the same area. A Level 2 router only communicates with Level 2 routers in other areas. A Level 1-2 router connects the other two sets. Thus, unlike OSPF which imposes a star-shaped topology, IS-IS allows the center to be a set of Level 2 networks.

Proponents of OSPF point out that OSPF has been extended to handle many special cases that arise in a large ISP. For example, OSPF has mechanisms to deal with stub networks, not-so-stubby networks, and communication with other IETF protocols. Proponents of IS-IS point out that IS-IS is less "chatty" (i.e., sends fewer messages per unit time), and can handle larger areas (i.e., areas with more routers). Thus, IS-IS is considered a suitable alternative to OSPF for special cases.

---

†The name is spelled out "I-S-I-S".

14.16 Trust And Route Hijacking

We have already observed that a single router may use an Interior Gateway Protocol to gather routing information within its autonomous system and an Exterior Gateway Protocol to advertise routes to other autonomous systems. In principle, it should be easy to construct a single piece of software that combines information from the two protocols, making it possible to gather routes and advertise them without human intervention. In practice, technical and political obstacles make doing so complex.

Technically, IGP protocols, like RIP and Hello, are routing protocols. A router uses such protocols to update its forwarding table based on information it acquires from other routers inside its autonomous system. Thus, RIP or OSPF software changes the local forwarding table when new routing updates arrive carrying new information. IGPs trust routers within the same autonomous system to pass correct data.

In contrast, exterior protocols such as BGP do not trust arbitrary routers and do not reveal all information from the local forwarding table. Instead, exterior protocols keep a database of network reachability, and apply policy constraints when sending or receiving information. Ignoring such policy constraints can affect routing in a larger sense — some parts of the Internet can be become unreachable. For example, if a router in an autonomous system that is running an IGP happens to propagate a low-cost route to a network at Purdue University when it has no such route, other routers that receive the advertisement accept and install the route. Consequently, routers within the AS where the mistake occurred will forward Purdue traffic incorrectly; Purdue may become unreachable from networks within the AS. The problem becomes more serious if Exterior Gateway Protocols propagate incorrect information — if an AS incorrectly claims to have route to a given destination, the destination may become unreachable throughout the Internet. We say that the destination address has been *hijacked*.

14.17 Gated: A Routing Gateway Daemon

A mechanism has been created to provide an interface among a large set of routing protocols, such as RIP, RIPng, BGP, HELLO, and OSPF. The mechanism also includes routing information learned via ICMP and ICMPv6. Known as *gated*†, the mechanism understands multiple protocols (both interior and exterior gateway protocols, including BGP), and ensures that policy constraints are honored. For example, *gated* can accept RIP messages and modify the local computer's forwarding table. It can also advertise routes from within its autonomous system using BGP. The rules *gated* follows allow a system administrator to specify exactly which networks *gated* may and may not advertise and how to report distances to those networks. Thus, although *gated* is not an IGP, it plays an important role in routing because it demonstrates that it is feasible to build an automated mechanism linking an IGP with BGP without sacrificing protection.

Gated has an interesting history. It was originally created by Mark Fedor at Cornell, and was adopted by MERIT for use with the NSFNET backbone. Academic

†The name is short for *gateway daemon*, and is pronounced "gate d."

researchers contributed new ideas, an industry consortium was formed, and eventually, MERIT sold *gated* to Nexthop.

14.18 Artificial Metrics And Metric Transformation

The previous chapter said that ISPs often choose routes for economic rather than technical reasons. To do so, network managers configure routing protocols manually, and assign artificial weights or distances, which the protocol software uses in place of actual weights or distances. Consider a network using RIP. If a manager wants to direct traffic over a path that has more hops than the optimal path, the manager can configure a router to specify that the optimal path is several hops longer. For example, one router can be configured to advertise a directly-connected network as distance 5. Similarly, when using OSPF, each link must be assigned a weight. Rather than base the weight on the capacity of the underlying network, a manager can choose artificial weights that make the protocol software prefer one path over another. In the largest ISPs, the assignment of artificial weights is important because it has a direct and significant relationship to revenue. Therefore, large ISPs often hire talented individuals whose entire job is to analyze routing and choose weights that will optimize revenue.

Software like *gated* helps network managers control routing by offering metric transformations. A manager can place such software between two groups of routers that each use an IGP and configure the software to transform metrics so routing proceeds as desired. For example, there may be a low-cost route used within one group that is reserved for internal use. To avoid having outsiders use the reserved route, software on the border between groups artificially inflates the cost of the route before advertising it externally. Thus, outsiders think the route is expensive and choose an alternative. The point is:

> *Although we have described routing protocols as finding shortest paths, protocol software usually includes configuration options that allow a network manager to override actual costs and use artificial values that will cause traffic to follow routes the manager prefers.*

Of course, a manager could achieve the same result by manually configuring the forwarding tables in all routers. Using artificial metrics has a significant advantage: if a network fails, the software will automatically select an alternate route. Therefore, managers focus on configuring metrics rather than on configuring forwarding tables.

14.19 Routing With Partial Information

We began our discussion of internet router architecture and routing by discussing the concept of partial information. Hosts can route with only partial information because they rely on routers. It should be clear now that not all routers have complete information. Most autonomous systems have a single router that connects the autonomous system to other autonomous systems. For example, if the site connects to the global Internet, at least one router must have a connection that leads from the site to an ISP. Routers within the autonomous system know about destinations within that autonomous system, but they use a default route to send all other traffic to the ISP.

How to do routing with partial information becomes obvious if we examine a router's forwarding tables. Routers at the center of the Internet have a complete set of routes to all possible destinations; such routers do not use default routing. Routers beyond those in ISPs at the center of the Internet do not usually have a complete set of routes; they rely on a default route to handle network addresses they do not understand.

Using default routes for most routers has two consequences. First, it means that local routing errors can go undetected. For example, if a machine in an autonomous system incorrectly routes a packet to an external autonomous system instead of to a local router, the external system may send it back (perhaps to a different entry point). Thus, connectivity may appear to be preserved even if routing is incorrect. The problem may not seem severe for small autonomous systems that have high-speed local area networks. However, in a wide area network, incorrect routes can be disastrous because the path packets take may involve multiple ISPs, which incurs a long delay, and ISPs along the path may charge for transit, which results in needless loss of revenue. Second, on the positive side, using default routes whenever possible means that the routing update messages exchanged by most routers will be much smaller than they would be if complete information were included.

14.20 Summary

The owner of an autonomous system (AS) is free to choose protocols that pass routing information among routers within the AS. Manual maintenance of routing information suffices only for small, slowly changing internets that have minimal interconnection; most require automated procedures that discover and update routes automatically. We use the term *Interior Gateway Protocol* (*IGP*) to refer to a protocol that is used to exchange routing information within an AS.

An IGP implements either the distance-vector algorithm or the link-state algorithm, which is known by the name Shortest Path First (SPF). We examined three IGPs: RIP, HELLO, and OSPF. RIP is a distance-vector protocol that uses split horizon, hold-down, and poison reverse techniques to help eliminate forwarding loops and the problem of counting to infinity. Although it is obsolete, Hello is interesting because it illustrates a distance-vector protocol that uses delay instead of hop counts as a distance metric. We discussed the disadvantages of using delay as a routing metric, and pointed

out that although heuristics can prevent instabilities from arising when paths have equal throughput characteristics, long-term instabilities arise when paths have different characteristics. OSPF implements the link-status algorithm and comes in two versions: OSPFv2 for IPv4 and OSPFv3 for IPv6. IS-IS is an alternative that handles some special cases better than OSPF.

Although routing protocols are described as computing shortest paths, protocol software includes configuration options that allow managers to inflate costs artificially. By configuring costs carefully, a manager can direct traffic along paths that implement corporate policy or generate the most revenue, while still having the ability to route along alternative paths automatically when network equipment fails.

EXERCISES

14.1 What network families does RIP support? Why?

14.2 Consider a large autonomous system using an IGP that bases routes on delay. What difficulty does the autonomous system have if a subgroup decides to use RIP on its routers? Explain.

14.3 Within a RIP message, each IP address is aligned on a 32-bit boundary. Will such addresses be aligned on a 32-bit boundary if the IP datagram carrying the message starts on a 32-bit boundary? Why or why not?

14.4 An autonomous system can be as small as a single local area network or as large as multiple wide area networks. Why does the variation in size make it difficult to define a single IGP that works well in all situations?

14.5 Characterize the circumstances under which the split horizon technique will prevent slow convergence.

14.6 Consider an internet composed of many local area networks running RIP as an IGP. Find an example that shows how a forwarding loop can result even if the code uses "hold down" after receiving information that a network is unreachable.

14.7 Should a host ever run RIP in active mode? Why or why not?

14.8 Under what circumstances will a hop count metric produce better routes than a metric that uses delay?

14.9 Can you imagine a situation in which an autonomous system chooses *not* to advertise all its networks? (Hint: think of a university.)

14.10 In broad terms, we could say that an IGP distributes the local forwarding table, while BGP distributes a table of networks and routers used to reach them (i.e., a router can send a BGP advertisement that does not exactly match items in its own forwarding table). What are the advantages of each approach?

14.11 Consider a function used to convert between delay and hop-count metrics. Can you find properties of such functions that are sufficient to prevent forwarding loops? Are the properties necessary as well?

14.12 Are there circumstances under which an SPF protocol can form forwarding loops? (Hint: think of best-effort delivery.)

14.13 Build an application program that sends a request to a router running RIP and displays the routes returned.

14.14 Read the RIP specification carefully. Can routes reported in a response to a query differ from the routes reported by a routing update message? If so how?

14.15 Read the OSPFv2 specification carefully. How can a manager use OSPF's virtual link facility?

14.16 OSPFv2 allows managers to assign many of their own identifiers, possibly leading to duplication of values at multiple sites. Which identifier(s) may need to change if two sites running OSPFv2 decide to merge?

14.17 Can you use ICMP redirect messages to pass routing information among *interior* routers? Why or why not?

14.18 Read the specification for OSPFv3. What is a *stub area*, and what is a *not so stubby area* (*NSSA*)? Why are the two important?

14.19 What timeout does the OSPFv3 standard recommend for a Hello interval?

14.20 Write a program that takes as input a description of your organization's internet, uses SNMP to obtain forwarding tables from all the routers, and reports any inconsistencies.

14.21 If your organization runs software such as *gated* or *Zebra* that manages multiple TCP/IP routing protocols, obtain a copy of the configuration files and explain the meaning of each item.