CHAPTER 6

IP Dynamic Routing

The network needs to understand how to get packets through from one side to the other. This can be accomplished in several ways. In a simple network with only one or two routers, it is probably most efficient to configure this routing information into the routers manually. However, in a large or complex network, the routers need to learn and update routing information through the network automatically. This is particularly true for networks that offer multiple redundant paths for fault-tolerance purposes.

Dynamic routing protocols give the network a way of healing around link or equipment failures. This is because they can see all of the different paths through a network and pick the best one at any given moment. When one path becomes unusable, another is selected.

The routing of IP packets is always handled by means of a routing table. This table is basically just a list of destination networks and the next hop required to get to these destinations. It may also contain other supplemental information, such as an estimate of how much it costs to use that particular route, and it may contain several different options for directing traffic to some destinations.

This concept of the cost of a route is relatively open and vague. The cost could be a function of any number of variables, such as the number of hops, the net latency of the path, the minimum bandwidth along the path, as well as other less tangible factors. For example, it may be better to avoid a particular route because of a usage charge. Or in some cases the network administrators direct traffic through networks under their direct control, instead of using a possibly shorter path through a foreign network.

It is interesting how these routing tables come into being, how they are updated when the topology changes, and how they avoid problems like loops. Several commonly used methods keep routing tables up-to-date.

The earliest and more popular routing protocols typically used a Distance Vector Algorithm. Then Link State Algorithms became popular. I discuss one popular protocol, Border Gateway Protocol (BGP), that uses a completely different algorithm relying on a Path Vector system.

All of these algorithms fulfill two main functions. First, they allow the routers on the network to keep track of changes in the state of the network that require changing routing tables. Second, they provide a mechanism for eliminating routing loops.

A routing loop is exactly what it sounds like: one router forwards a packet to its next hop to be delivered to the eventual destination. But instead of sending the packet on, the second router just forwards it back to the first one, perhaps via other intermediate routers. This is clearly a serious problem for a network, and it is relatively easy to get such loops when the routers are responsible for figuring out for themselves how to send data through the network. This is why sophisticated algorithms are required to prevent them.

Before discussing the more sophisticated dynamic methods of maintaining routing tables through a network, I start with simple static routing. This discussion helps explain the problems that dynamic routing was developed to solve. Further, it is common in large networks to use a mixture of static and dynamic routing, so it is important to understand where each method is useful.

Static Routing

Conceptually, the simplest method for maintaining routing tables is to configure them into the routers manually. This method naturally relies on the accuracy of the network administrator to avoid problems such as loops. It is also up to the administrator to update the tables whenever something changes anywhere in the network.

Maintaining a large network using static routes presents several problems, however. It is cumbersome and labor intensive. It is impossible to achieve automatic fault tolerance because changing routes in response to a network failure always takes time. It also demands that the network administrator have perfect knowledge of the state of the network.

Furthermore, when building static routing tables on a network of any size, making mistakes that isolate remote parts of the network is remarkably easy. When this happens, technicians may need to visit the remote devices and manually reconfigure them from the console. Put simply, static routing is not an effective way of handling the main routing of a network. However, it still has its place, even in a network that uses a more sophisticated dynamic routing protocol to build its tables. For example, when connecting to external networks, it is often easier to configure a few static routes than to share routing information with the external network. This is particu-

larly true when the external network is always accessed through one point. If this Access point goes down, the network doesn't have a backup path, so there is no point in updating the route to elsewhere.

In fact, it may be worse to try more sophisticated methods. Suppose there is more than one external network, such as an Internet connection and a separate secure connection to a partner organization. If that secure connection becomes unavailable, you probably don't want the packets sent out to the Internet instead. In all likelihood, this would happen if the network exchanged routing information with the partner organization.

This is because the partner network's IP-address range is not part of the internal range. When the dynamic routing information from that network disappears, the global default static route pointing out to the public Internet is used instead. Depending on the specific type of failure and the exact network configuration, the same thing could happen if you used a static route to the partner network. For the purposes of this example, let me assume that the static route was added in a way that allows it to remain in effect even if the link goes down.

You may need to use static routes in networks involving equipment that doesn't support the preferred dynamic routing protocols. In this case, though, you would only use the static routes to get through these isolated parts of the network.

Static routes definitely have their place, but they should be used sparingly. Each time you configure a static route you have to ask whether it would be better if this routing information were learned dynamically. As you will see in the following discussion, once a static route exists on one of your routers, it is relatively easy to use the dynamic routing protocol to distribute this information throughout the rest of the network.

Floating Static Routes

Another important kind of static route is a *floating static route*. This feature is not available on all vendors' routers. A floating static route is like a normal static route, but it isn't always present. That is, if a better route is available, the router will not look at this static route. But if that better route disappears, then the router will revert to the floating static route.

The way it works is simple enough in concept. The floating static route is manually configured in the router, the same as any other static route. But it has an extremely high metric to indicate a high cost associated with this path. If there is any other path available, it will be better than this one and consequently will not be used.

This feature is commonly used in dial-backup situations, for example. When the network is working properly, a remote router receives its routing table via a dynamic routing protocol. However, when there is a failure, the router stops receiving any dynamic routing information, and it flushes all of this dynamic information out of its

routing table. When that happens, the floating static route suddenly starts to look good despite its high metric. The router inserts this route into its routing table, and this triggers the dial-backup process.

The key to floating static routes is that there is a magic metric value. Normally, if a router has a static route, it will use it. On Cisco routers it is conventional to use a metric of 200 or greater for floating static routes, although values as low as 150 appear to work just as well.

Types of Dynamic Routing Protocols

It is customary to describe routing protocols by both their function and the algorithms they employ. Functionally, a routing protocol can be either an Interior Gateway Protocol (IGP) or an Exterior Gateway Protocol (EGP). There are three commonly used routing algorithms. Distance Vector Algorithms are used by RIP, IGRP, and EIGRP. OSPF, on the other hand, uses a Link State Protocol Algorithm to find the best paths through the network. BGP uses a Path Vector Algorithm.

I describe the algorithms in more detail in the context of the actual protocols, but it is necessary to clarify the difference between Interior and Exterior Gateway Protocols before I go on.

Simply put, an Interior Gateway Protocol handles routing within an Autonomous System, and an Exterior Gateway Protocol deals with updating routes between Autonomous Systems. But what is an Autonomous System?

This term replaces the more vague term network. If one organization has a network, that concept is easy to understand. If that network is connected to another organization's network, how many networks are there? Really there is just one big network, since you can send packets from a device on one side to those on the other. Is the public Internet one network, a collection of millions of small networks, or a little of both?

The word network stops having much meaning when you talk about these very large scales. It is actually the administrative boundaries between these networks that matter. Interconnecting two networks allows traffic to flow between them, but this doesn't change the fact that Company A controls the first network and Company B controls the second one.

It has been necessary to introduce the phrase *Autonomous System* (AS) to describe this separation of control. To make things more confusing, once this distinction exists, you can then break up a large corporate network into many ASes.

This brings me back to the original definition of terms. IGPs operate within an AS. You can opt to break up a network into several ASes to isolate your IGPs. It is often possible to make an extremely large or complex network operate more efficiently by splitting it up.

In most cases you can create a stable LAN with only one AS and one IGP. Most IGPs (excluding RIP) can handle all but the largest local or Campus Area Networks with one AS if they are configured properly. In extremely large networks it can become necessary to split up ASes.

There are other situations that force a network designer to interconnect distinct ASes within a smaller network. In some cases, a large enterprise network might be managed by different groups, sharing only a backbone. It's also common to connect ASes of different companies because of mergers or other cooperative business requirements. I include a discussion of BGP in this chapter to deal with these sorts of situations.

The possibility of using several ASes in a network introduces the concept of an Autonomous System Boundary Router (ASBR). These are the routers that interconnect different ASes. This term is most useful to the IGPs, as the ASBR represents a portal to the next AS.

As long as I'm talking about boundaries between hierarchical levels of dynamic routing protocols, another important type of router is an Area Border Router (ABR). This concept will be discussed in depth in the "OSPF" section. OSPF has a built-in hierarchical structure in which each AS is divided up into a number of separate areas. Using areas helps to reduce the amount of routing information that each router needs to maintain. The ABR routers act as portals between these areas.

Throughout this chapter I point out the ways that the different routing protocols contribute to the hierarchical design model favored by this book.

RIP

One of the oldest dynamic routing protocols used by IP is the Routing Information Protocol (RIP). RIP uses a Distance Vector Algorithm. It should be stressed from the outset that RIP is a poor choice for a large network. I include it in this discussion for two reasons. First, it makes a good introduction for readers who might be unfamiliar with dynamic routing protocols. Second, despite its age, it is still common to encounter specialized pieces of network equipment that support RIP as their only dynamic routing protocol. Integrating these devices into a network requires a good working knowledge of RIP.

There are two common versions of RIP called, appropriately enough, RIP-1 and RIP-2. RIP-1 is the original version introduced during the early days of ARPANET (the spiritual predecessor to the modern Internet) and is documented in RFC 1058, although the protocol was a de facto standard long before this RFC was published. RIP-2 is an updated version of RIP that improves several key operational problems with the original version. The current version of the protocol is documented in RFC 2453.

Although it is often useful in isolated pockets of a network, there are several reasons to avoid using RIP on a network-wide basis. It is slow in responding to topology changes. It is only effective in small- to medium-sized networks and breaks down completely if the distance between any two parts of the network involves more than 15 hops. It can also cause serious traffic-overhead problems in a network with a large number of routes, particularly over slow links.

The original RIP implementation was actually made for UNIX hosts because it effectively predated modern routers. Thus, every BSD UNIX operating system has always been equipped with a program called routed (for routing daemon). This is not merely an interesting quirk of history; it also represents one of the most dangerous problems with running RIP: there are end devices that expect to take part in the routing protocol.

Defenders of the routed argue that it helps these end devices find the appropriate routers for their desired destinations. However, if a network is well designed, it should be possible to simply point all end devices to a single, default gateway address that lets them reach all destinations transparently. In a well-designed network there is never any need to run a dynamic routing protocol on end devices.

End devices don't need to know how the network routes their packets. Letting them take part in the dynamic routing protocol doesn't just give these devices unnecessary routing information. It also allows these end devices to affect the routing protocol, even though they aren't in a position to understand the physical topology of the network.

As a result, it is quite easy for an end device running routed to mislead the network about the best paths. In particular, if the end device is configured with a default gateway, it will attempt to tell the rest of the network about this information. Then if something goes wrong on the network—causing real routes to disappear—the routers will look to the next best option. This often turns out to be the end device that broadcasts the false default route. Since a default route is a route to anywhere, all routing in the network suddenly becomes confused.

In most cases there is a simple way to get around this problem. You can configure the routers that connect to end-device segments so that they ignore all RIP information coming from those segments. This means that you can only use specialized router-to-router segments to carry routing protocol information. These segments cannot contain any end devices.

With RIP an end device can listen passively to routing updates without taking an active role in building the routing tables. This is significantly less dangerous. However, if a network is well designed, there should be no need for any end device to see this information. It should be able to get a packet to any valid destination just by forwarding it to its default router. This default router should respond to topology changes faster and more accurately than the end device. So allowing the end device to make important routing decisions is likely less reliable.

RIP Functionality

The main idea behind RIP is that every router maintains its own routing table, which it sends to all of its neighbors periodically. The neighbors update their own tables accordingly. Every route in the table has a cost associated with it, which is usually just the number of hops to the destination.

Figure 6-1 has a small network containing four routers and eight Ethernet segments. Router A knows about the two routes directly connected to it: `10.1.5.0/24` and `10.1.12.0/24`. It also knows that it has two neighboring routers, B and C.

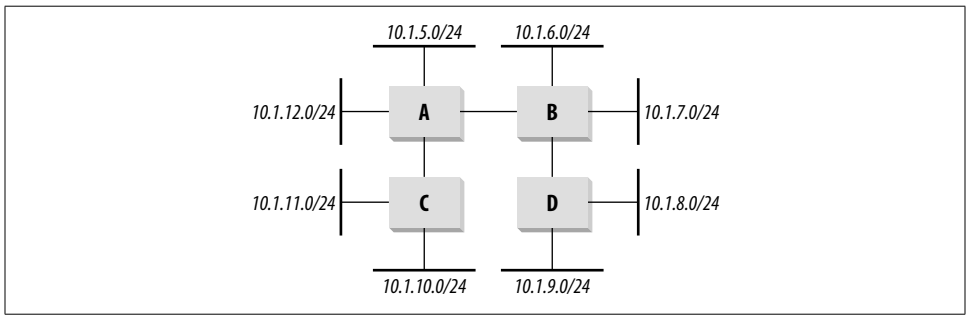

Figure 6-1. Distributing routing information with RIP

When Router B receives information about these routes from Router A, it adds these routes to its own table. For the routes that Router A indicates are directly connected, Router B has to specify some higher cost.

In RIP this cost is called a *metric*. By default, the metric just counts the number of hops. Router A uses a metric of 1 for its directly connected networks, so Router B will increment this metric and show the same entries in its routing table with a metric of 2.

At this point, Router B's routing table is shown in Table 6-1. The table lists the routes in numerical order by destination network because the routers generally display them this way. Router B now sends this same information along to Router D. At the same time, all other routers in the network similarly exchange their routing tables. Clearly, it will take a few rounds of updates before Routers C and D have one another's tables.

Table 6-1. Intermediate routing table for Router B

Destination network	Metric	Next hop
`10.1.5.0/24`	2	Router A
`10.1.6.0/24`	1	Local
`10.1.7.0/24`	1	Local
`10.1.12.0/24`	2	Router A

When the process is complete, Router A's routing table looks like Table 6-2. Note that Router A doesn't have a direct connection to Router D, which owns `10.1.8.0/24` and `10.1.9.0/24`, so it has to direct traffic for these destinations to Router B.

Table 6-2. Final routing table for Router A

Destination network	Metric	Next hop
10.1.5.0/24	1	Local
10.1.6.0/24	2	Router B
10.1.7.0/24	2	Router B
10.1.8.0/24	3	Router B
10.1.9.0/24	3	Router B
10.1.10.0/24	2	Router C
10.1.11.0/24	2	Router C
10.1.12.0/24	1	Local

As long as nothing in the network changes, this routing table remains constant. If something changes, such as a link becoming unavailable, the protocol needs to make it known. To do this, every router sends its current routing table to all of its neighbors every 30 seconds. This update serves two purposes: it allows the routers to ensure that the neighbor routers are all still working, and it makes sure that everybody has the latest routing table.

Suppose the link between Routers B and D suddenly breaks. Router B finds out about this because it stops seeing updates from Router D. But Router B shouldn't react immediately. There may just be a delay or some noise in the network. Perhaps the update message was lost due to congestion caused by a random burst of traffic. There are many reasons why an individual update might not be received. So RIP waits 180 seconds before declaring the routes dead. After the route is considered dead, the protocol will wait another 120 seconds before actually removing it from the table.

This long wait time reflects, in part, the poorer network media available when the protocol was developed. It was not unusual to lose several packets in a row, so the relatively long wait time is a trade-off between wanting to respond quickly to the change in topology and wanting to prevent instability. A busy or noisy link shouldn't cause the routing tables to search continually for new paths. Doing so results in an unstable network.

When a new router is placed on the network, it needs to get a good routing table as quickly as possible. When it first comes up, the router sends out a special request message on all of its active interfaces. Any neighboring routers found on these interfaces respond immediately to this request with a full routing table, rather than

waiting for the regular update cycle. The new router integrates this information into its own routing table, adding information about the new routes that are unique to this new device. Then it turns around and updates its neighbors with the resulting routing table, and the neighbors propagate the new routes throughout the network.

Instead of using the default metric just to count hops to a destination, it can be useful to specify higher values for slower links and lower values for higher bandwidth links. In this way, the network tends to prefer the fastest paths, not just the shortest.

Figure 6-2 shows an example of how this concept might work. Router R1 has connections to both R2 and R3 that can eventually lead it to the destination network, `10.1.6.0/24`. The link to R2 is a standard Ethernet connection, and the link to R3 is a 56Kbps point-to-point serial link. The network should favor the faster link, so the Ethernet connection is given a metric of 3 and the serial connection a metric of 10. Similarly, suppose that the connection from router R2 to R4 is Ethernet, so this too will have a metric of 3. The connections between R2 and R3 and between R3 and R4 are Fast Ethernet, though, so these high-speed links will have a metric of 1.

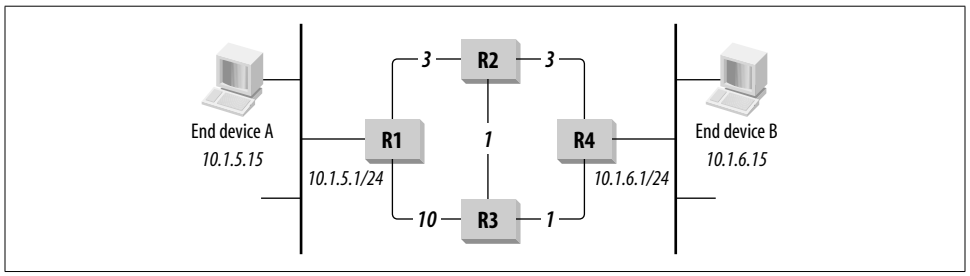

Figure 6-2. Slower links can be configured with higher metrics

There are four possible paths to get a packet from R1 to the destination network `10.1.6.0/24`. It can go from R1 to R2 to R4, with a total metric of six. Or, it can go R1 to R2 to R3 to R4, for a total of five. Similarly, the packet can go R1 to R3 to R4 with a metric of 11. The final possible path is from R1 to R3 to R2 to R4, which has a metric of 14.

So the lowest metric path is the one that goes R1 to R2 to R3 to R4. Because the metric values have been adjusted, the path with the lowest metric is not necessarily the one with the lowest hop count. It is the fastest path, though, which was the point of changing the defaults.

It isn't necessary to give the same metric to all links of a particular type. In fact, you have to be very careful with RIP that you never exceed a total metric of 15 along any valid path. This requirement is too restrictive to establish set rules of different metrics for different media speeds. However this philosophical approach will be quite useful later in the section on "OSPF."

Avoiding Loops

In every dynamic routing protocol, one essential goal is to find the best way to get from A to B. This generally means that every router in the network has to figure out how to get from itself to every place else. Every router has its own routing table that says, regardless of how the packet got here, this is how to get to its ultimate destination. The problem is that, with every device making these decisions on its own, it is possible to wind up with routing loops. Figure 6-3 shows a small network that has four routers. Suppose end device A wants to send a packet to end device B.

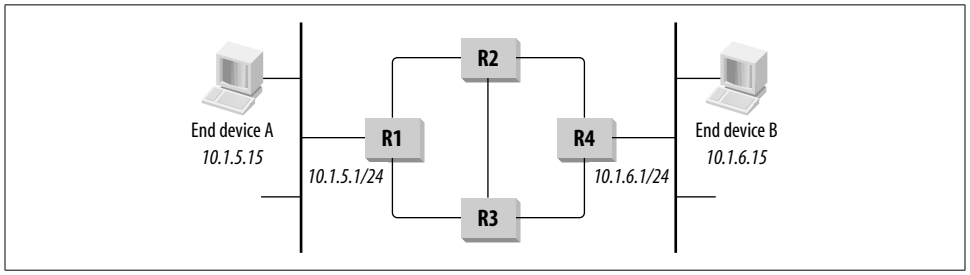

Figure 6-3. Routing loops

A first looks at its own internal routing table for B's address. The destination is not part of its own subnet, so it has to find a route to the destination subnet. Since only one router is on the segment, the end device needs only one default gateway entry in the local routing table. This entry sends everything to router R1.

Now R1 has two options for how to direct the packet. The destination IP address in the packet is `10.1.6.15`, so it looks in its routing table for anything that matches this address. There are clearly two possibilities in the picture, R2 and R3. Suppose it sends the packet to R2.

Then R2 must decide how to get to the destination, and it has three possible paths from which to choose: the ones leading to R1, R3, and R4. If everything is working properly, it should see that the path through R4 is the shortest and use that. Suppose it picks one of the others, though—the one through R3, for example. This might happen if a high cost has been assigned to the link to R4, indicating that it is a slow or expensive link.

The routing tables can start to get into trouble because R3 also has three possible paths. Two of these paths, the ones to R1 and R2, send the packet back where it has already been. If R3 chooses either of these paths, it will create a loop.

Fortunately, IP includes a mechanism to break loops like this so that packets do not circulate indefinitely. Every IP packet has a Time To Live (TTL) field in its header. Originally, when network latency was very high, TTL had a real time meaning, but today it is simply a hop counter. In most IP packets the TTL field starts out with a value of 255. Each time a router receives this packet, it decrements the TTL value

before forwarding it. If the value eventually reaches 0, the packet is discarded. When I talk about multicast networking in Chapter 10, I discuss another useful application of this TTL field.

You should notice a few important things about routing loops. First, they are fundamentally a Layer 3 phenomenon. It doesn't matter whether the network has multiple connections. A loop could happen if R2 forwarded the packet back to R1 so even if there are no physical loops, a network can still have routing loops.

Also realize that a routing loop happens on a per-route basis. The network might have a loop for one route, say `10.1.6.0`, but have no problems with another destination, such as `10.1.5.0`. This is different from a Layer 2 loop, which can take all of the traffic into the spin cycle. Every good routing protocol has several techniques for finding and eliminating loops. One of the main ways that RIP avoids loops is by *counting to infinity*.

The protocol's designers believed that RIP would not converge well in large networks. They estimated that if the distance between any two devices was more than about 15 hops, guaranteeing reliable convergence after a topology change would be difficult. So they somewhat arbitrarily defined infinity as the number 16. This may seem like a small number, but it should be as small as possible if the routers have to count to it quickly.

Look at Figure 6-4. Suppose Router R4 suddenly dies and there is no longer any router available for `10.1.6.0/24`. The protocol somehow has to flush this route out of the tables of the other three routers.

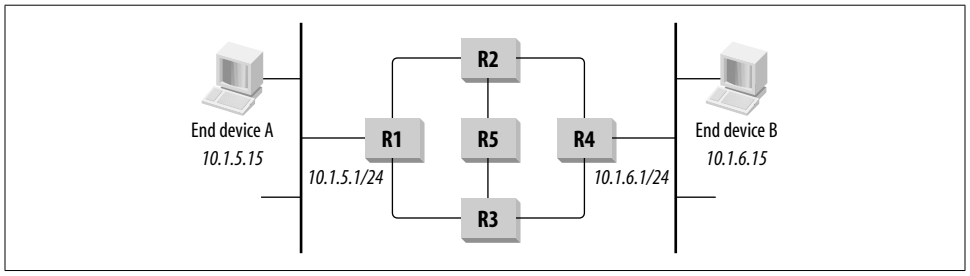

Figure 6-4. Counting to infinity

Routers R2 and R3 will both eventually time out waiting for an update and remove the route they learned from R4. However, Routers R1 and R5 both have routes to get to R4; they can send packets via either R2 or R3.

In the next updates from R1 and R5, R3 will see that they both have routes with a metric of 3 for this destination. R3 will set its own metric to 4 and try to use one of these routes. Meanwhile, R2 will see the same information and do the same thing. Both R2 and R3 will distribute this information back to R1 and R5 in the next cycle.

When R1 and R5 learn that their preferred paths for this destination have suddenly developed higher metrics, they will simply update their own tables, setting the metric to 5. The updated tables are sent back to R2 and R3, which set their metrics for this route to 6 and send it back again. It should be clear why having infinity as small as possible is a good thing. The extinct route will not be removed from the routing table until all routers agree that it is infinitely far away.

RIP has another simple but clever technique for avoiding loops. The protocol stipulates that a router can only send out information about routes that it actually uses. Even if several paths are available, the routers preselect the best one and only worry about the others if this best route becomes unavailable.

For example, look at Figure 6-4 again. Router R2 knows that it can get to the destination network `10.1.6.0/24` through R4. Even though it has heard about alternate routes from both R1 and R3, it never advertises these. Then, when the path through R4 becomes unavailable, it picks either the path through either R1 or R3 and ignores the other.

When R3 also loses its connection with R4, only one possibility remains: R1. However, R1 only advertises the path that it actually uses, which is the one through R2. Eliminating unused—and therefore unnecessary—path options allows the protocol to converge on new paths more quickly when the topology of the network changes.

The *triggered update* is another important feature of RIP helping it converge more quickly. As I described earlier, all of the routing table exchanges normally happen on a timer. However, when a router's interface physically goes down, the router knows for certain that the associated route is no longer available.

Even with static routes, when this happens the router will flush the corresponding route from its tables. When using RIP, the router follows up on this action by immediately telling its neighbors that this route has disappeared. It does this by telling the other routers that this route now has a metric of 16.

Each time a router changes its metric for a particular route, it also executes a triggered update of this information. This allows the information to propagate through the network quickly.

Meanwhile, another router might still have a connection to this network that had been considered worse because of a higher metric. This new information will propagate through the network as it is now better than the other unavailable route.

Split Horizons in RIP

The counting-to-infinity example in the previous section might have seemed slightly more complicated than necessary by including router R5 in the middle. But this was necessary because of another clever feature of RIP called *Split Horizon* that is designed to help destroy loops.

In a regular Split Horizon algorithm, routers simply refrain from passing information back to the router that originally sent it. If R1 and R2 have a link, as they do in Figure 6-4, then R1 will not bother telling R2 about the routes it heard from R2 in the first place.

In fact, RIP employs a slightly modified version of a Split Horizon algorithm called *Split Horizon with Poisoned Reverse*. To understand this, suppose again that R1 and R2 are sharing routing information. When R1 sends its routing table to R2, it includes the routes it received from R2, but it sets the metric to 16 (remember 16 = infinity in RIP). To see how this causes things to converge faster, the reader is invited to repeat the counting-to-infinity example using the network in Figure 6-3, which is the same as 6-4, but without router R5.

Variable Subnet Masks

One of the most serious drawbacks with the original RIP specification was how it handled subnets. In Version 1, RIP assumed that all of the subnets for a given network had the same mask. As I already discussed, the ability to vary subnet masks in a complex network is extremely useful. This ability is called Variable Length Subnet Mask (VLSM). It allows not only more efficient use of the address space, but also makes it much easier to summarize the routes to a particular part of the network.

Removing this restriction was a driving force behind the introduction of RIP Version 2. To accomplish this, the protocol had to be modified so that every subnet address could have its subnet mask specified with it. In the original Version 1 specification, the mask information was not included. Consequently, every subnet of any given network was assumed to have the same mask.

This issue is sufficiently serious that it eliminates RIP Version 1 as a candidate routing protocol in most modern networks. However, there are two alternative options. One is to use RIP Version 2 if the equipment supports it. The other, equally viable option, is simply to restrict the use of RIP to small portions of the network where the condition of equal subnet masks can be satisfied.

RIP Version 2 also has the advantage of using multicast rather than broadcast to send its updates. The advantages of multicast are discussed in depth in Chapter 10. In this case it makes it safer to have end devices on the same network segment as routers that communicate via RIP. Because they are broadcast, RIP Version 1 packets must be examined by every device on the network segment. If there are many routers on the network segment, this can cause CPU loading problems on the end devices, even those devices that don't know or care anything about RIP.

In fact, the only reason that RIP Version 1 is ever used in a modern network is for compatibility reasons with legacy equipment. In most of these cases, the RIP routing information is isolated to local communication between the legacy equipment and a modern router. This modern router then redistributes the RIP routes into a more

appropriate routing protocol. To allow full two-way communication, this router must also summarize the rest of the network into RIP for the legacy equipment to use.

Redistributing with Other Routing Protocols

Another key difference between RIP Versions 1 and 2 is the inclusion of *Route Tags*. A Route Tag is a two-octet field used to indicate routes that come from outside of the RIP AS. These could come from another IGP or an EGP, or they could even specify routes that are statically configured on a router.

RIP does not use the Route Tag information directly while routing packets, but it is included because it is often useful to know from where different routes came. In other routing protocols Route Tags often ensure that traffic remains inside the AS wherever possible. So, any tagged route will have a higher cost than the worst interior route to the same destination.

This is not practical in RIP, however, because of the small range of allowed metrics. Since any route with a metric of 16 is considered unreachable, it is not possible to use this for exterior routes. The low value of infinity in RIP makes it extremely difficult to balance metrics so as to prefer certain paths to others.

So the Route Tag field is included primarily for information and to allow RIP to pass this information to other routing protocols (in particular, BGP) that can use it.

IGRP and EIGRP

In response to the scaling problems of RIP, Cisco developed a proprietary IGP of its own called Interior Gateway Routing Protocol (IGRP). This protocol was later updated and improved, with the result called Enhanced IGRP (EIGRP). Because these protocols are proprietary, they are only implemented on Cisco equipment. As always in this book, I recommend that readers avoid proprietary protocols for compatibility reasons. However, I include this discussion on IGRP and EIGRP because they are remarkably efficient and easy to implement. Furthermore, Cisco has produced not only IP, but also IPX and AppleTalk versions of EIGRP, making multiprotocol networks easier to administer.

IGRP and EIGRP are distance-vector algorithms just like RIP. But they are able to operate on much larger networks while consuming much less bandwidth. There are a number of differences between the original IGRP and the more recent EIGRP. One of the most important is that, like RIP Version 1, IGRP cannot handle Variable-Length Subnet Masks (VLSM). The other main difference is the use of a new algorithm called Diffusing Update Algorithm (DUAL)in EIGRP, which provides better convergence properties.

The enhancements in EIGRP make it a much more useful protocol. I recommend avoiding IGRP in favor of EIGRP wherever possible. In fact, the only place where IGRP is likely to be used is in older networks originally built before the advent of EIGRP. In most cases it is relatively easy to complete an upgrade from IGRP to EIGRP simply by configuring the protocols to redistribute routing information into one another. Then it should be possible simply to move the IGRP/EIGRP dividing line through the network one router at a time. Note that this might require the temporary use of additional static routes because IGRP does not cope well with splitting up networks in ways that cannot be easily summarized. The remainder of this section focuses on EIGRP.

Basic Functionality

An important difference between EIGRP and RIP is how they handle routing updates. While RIP sends out the entire routing table periodically, EIGRP only sends incremental updates. So, if a router has no updates to send, it sends only a tiny HELLO packet to each of its neighbors. This allows EIGRP to consume much less bandwidth than RIP.

The first thing an EIGRP router does when it comes up on the network is send out HELLO packets to establish the neighbor relationships with all of the routers on directly connected networks. As soon as it discovers a new neighbor, the router sends it a query requesting that it send its routing table.

EIGRP is uses a Distance Vector routing algorithm, like RIP. However, unlike RIP, the distances are calculated based on the speed and latency of each path. The latency is found by adding up the round-trip delays of every link in the path. The speed comes from the bandwidth of slowest link. The actual formula used by EIGRP* is:

$$metric = 256(\Sigma(delays)/10 + 10^7/(minimum\ bandwidth))$$

The delays in this equation are measured in microseconds; the bandwidth in kilobits per second.

Figure 6-5 shows a simple example of how these metrics work. Router R1 connects to Router R2 via a 10Mbps Ethernet connection. The delay on this link (including the latencies of the routers themselves) is 2000 microseconds (2 milliseconds). Router R2 connects to Router R3 over a 4Mbps Token Ring with a delay of 3500 microseconds (3.5 ms).

* In fact, this is the simplified version of the formula that results from using the default k values. There are five variables, k*1* to k5, that control the relative weightings of the delay and bandwidth. They also introduce using the reliability and load of the link to control the metric further. However, I advise using the default k parameters to avoid confusion and instability that can result from accidentally choosing poor combinations of values. As always in networking, simplicity is a virtue.

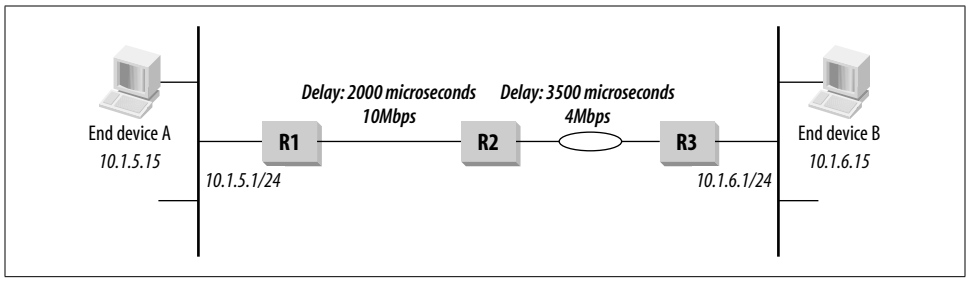

Figure 6-5. Metrics in EIGRP

In calculating the metric to the network `10.1.6.0/24`, Router R1 must first figure out what the lowest bandwidth is. It gets this information from its neighbor, R2. R2 has already found out that the minimum bandwidth is associated with the 4Mbps Token Ring that it uses to connect to R3. So R1 compares this to its own link and uses the lower one. In this way, each router along a multihop path needs only to include one minimum bandwidth for each route. At each successive hop, the next router compares the reported minimum bandwidth with that of its own link and takes the slower one.

For each successive hop, the routers must also keep track of the total delay so far. Each router adds the delay for its own leg to the running total. This greatly simplifies the calculations that each router needs to perform. In this example:

$$metric = 256((2000 + 3500)/10 + 10^7/4000) = 780800$$

This metric is a large number. In RIP the maximum metric is 15. In EIGRP the maximum is $2^{32} = 4{,}294{,}967{,}296$. Clearly, this means that EIGRP can't use the same counting-to-infinity algorithm to get out of loops the way RIP does.

Instead, EIGRP relies on its Split Horizon implementation and on its neighbor relationships to avoid loops. Recall from the RIP discussion that Split Horizon means that the router doesn't advertise itself as a route to any device that it considers closer to the destination. In particular, if a router is using a particular next hop router to get to some destination network, then it never tells the next hop router that it knows how to get to that destination.

RIP used a modified version of Split Horizon in which it does advertise the path to the destination network, but it does so with a metric of infinity so that it is never used. This is called Split Horizon with Poisoned Reverse. EIGRP uses a similar rule for exactly the same reasons.

EIGRP only works with incremental updates rather than distributing the entire routing table. So when a router detects a topology change, it alerts its neighbors to flush this route from their tables. They can then determine a new optimal path. There is no need to count to infinity incrementally before removing the route. The DUAL algorithm eliminates routing loops.

Unlike OSPF, EIGRP does not support the use of areas. The entire AS acts as a single unit. However, EIGRP can use its autosummarization feature to achieve many of the same benefits as OSPF does using areas. In fact, the existence of areas in OSPF forces all route summarization to be done at the Area Border Routers. In EIGRP, however, route summarization can be done at multiple levels.

EIGRP is said to be a Classless routing protocol because it can summarize at any bit in the network address, without being concerned about the class of the address range. Thus, a carefully designed hierarchical network can have a very efficient routing table. Figure 6-6 shows an example of how this might work. This is a hierarchical network design with a large number of Distribution Areas. When allocating the IP addresses downstream from the Distribution Routers, one has to be careful to ensure that the routes can be summarized.

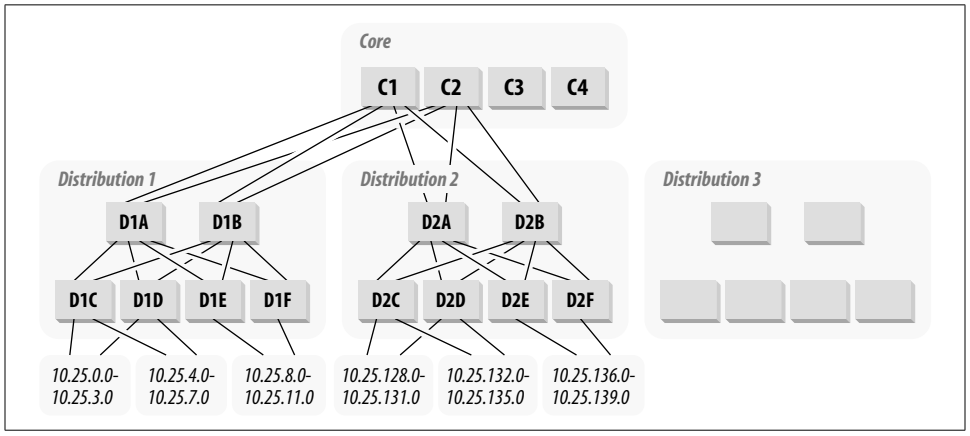

Figure 6-6. Route summarization in an EIGRP network

The most important restriction on this sort of summarization is that you cannot break up any summarized address range across multiple Distribution Areas. In Figure 6-6 a range of subnets from `10.25.0.0` to `10.25.3.0` to is given the first group of LAN segments in Distribution Area 1. These could be subnetted in whatever way is appropriate. But Routers D1C and D1D can summarize the connections to this group as `10.25.0.0/22`. Then you need to be careful only that you don't assign subnets from these ranges anywhere else in the network.

Following the summarization process up the diagram toward the Core, Routers D1A and D1B can have a single summary route for both `10.25.0.0/22` and `10.25.4.0/22`, namely `10.25.0.0/21`. These routes then point to Routers D1C and D1D.

Suppose the designer needs to reserve a large number of subnets for growth in this Distribution Area. He might allow the summary route for the entire area to be `10.25.0.0/17`. Then Distribution Area 2 could be summarized as `10.25.128.0/17`. In this way, each router can have an extremely compact routing table that is easy to update

throughout the network. The routers in the Core don't have to care about whether a particular subnet, such as 10.25.3.5/32, is behind router D1C or D1F. All they have to know is that D1A and D1B take care of a large range of addresses that happens to include this subnet.

The same sort of summarization also happens in the other direction. If Distribution Area 2 is summarized as 10.25.128.0/9, then every router in Distribution Area 1 will see a route for this summary network pointing into the Core. There is no need for these routers to see any more detail.

By default, EIGRP automatically summarizes whenever two different IP networks meet if the networks represent two different Classes. For example, where a section of network containing 172.16.0.0/16 meets another one using 172.17.0.0/16, they will both be summarized.

However, if the subnet addresses are allocated in a good hierarchical scheme, you can configure the routers to summarize other smaller ranges. It is a good idea to do so because every router should have a simple, concise, routing table.

If the network is not planned this way, then summarization doesn't make sense. For example, some of the subnets of 10.25.3.0 are located in Distribution Area 1 and others are in Distribution Area 2, then it is not possible to summarize either one.

EIGRP can also handle multiple routes to a particular destination. The network in Figure 6-6 has two connections from each router to the next level above it. Each of the higher-level routers will see two routes to everything downstream from it. EIGRP allows these routers to keep and use both of these routes.

In the example, there are two paths from Router D1A to 10.25.3.0: one through D1C and the other through D1D. If both of these links have the same metric, then EIGRP uses equal-cost multipath routing. In most cases, the router simply alternates the traffic flows between the different paths. The packets belonging to a particular TCP session are called a flow. So equal cost multipath routing keeps all of these packets on the same path. However, as each new session is established, the router attempts to balance the paths since they have the same cost.

If the packets from several different flows are coming sufficiently quickly, the router sends the second packet out the second path before the first packet has finished departing along the first path. This allows a simple form of load sharing between the paths. However, this form of load sharing is not terribly bandwidth efficient, so you will get considerably less than twice the bandwidth of one path in practice.

Active and Stuck-in-Active Routes

EIGRP uses an interesting technique for keeping its routing tables up-to-date. Even though it only uses the best route, the EIGRP topology table keeps a list of every path to every subnet. This way, if the best path goes away it can select a *feasible*

successor. But if there are no feasible successors when a route disappears, the router puts this route into an "ACTIVE" state and queries its neighbors to find a new path to the destination. If one or more of the neighbors knows a path to this destination network (or a summary route that contains this one), they respond. But if they do not have a route, they in turn query their neighbors.

Sometimes the destination is simply nowhere to be found. This can happen because a failure somewhere in the network has isolated some subnets. Sometimes the process of trying to find a new path can fail to converge. In the ever-expanding chain of queries from one router to the next, each device is waiting for a response. If the network is too large or if it contains too many high-latency sections, it may become difficult for this process to converge.

If the queries for an "ACTIVE" route are not satisfied within the timeout period of a few minutes, the router gives the dreaded "Stuck In Active" message. It then clears the neighbor relationship with the router that failed to respond. This can happen either because the route has disappeared or because a communication problem has broken the chain of queries somewhere in the network. Either way, "Stuck In Active" represents a serious problem, particularly if it happens repeatedly.

When the routers in an EIGRP network issue large numbers of "Stuck In Active" messages, it is important to determine where thing are getting "Stuck." This is by far more serious than the "ACTIVE" problem, which just means that a route is missing. When these messages appear, the network engineer should find out which neighbor relationships are being reset. This could be happening anywhere in the network, not necessarily on or adjacent to the router that reports the "Stuck In Active."

The easiest way to find these problems is to ensure that EIGRP is configured to log neighbor status changes. Then, when the "Stuck In Active" messages appear, attempt to track where the neighbor relationships are flapping. In some cases the neighbors that are changing are randomly dispersed throughout the network. This may indicate that the EIGRP AS has simply become too large to be stable. This can happen particularly when the automatic route summarization features of EIGRP are not used effectively.

Interconnecting Autonomous Systems

EIGRP networks are grouped into Autonomous Systems (ASes). Each router that has EIGRP configured must specify the AS number. All neighbors must be in the same AS to exchange routes.

It is possible to break up an EIGRP network into multiple ASes, but these ASes cannot be directly connected to one another. The problem is that the router sitting on the border between the two ASes is a full member of both. It maintains a distinct topology database for each AS. But consider what happens when a route in one of the ASes disappears. As I discussed in the previous section, that route becomes

"ACTIVE" as EIGRP attempts to find an alternate path. When the router that is a member of both ASes marks this route as "ACTIVE," it queries all of its neighbors for a possible alternate. That includes the neighbors that belong to the other AS. So any ACTIVE route queries from one AS are forwarded over to the other AS. This means that if there are stability problems in one AS, they will be inherited by the other.

The main reason to break up a network into multiple ASes is to help convergence and stability of each of the smaller units. In doing so you must be careful to separate these ASes more effectively by using another protocol between them. Since EIGRP is an IGP, it is natural to use an EGP, such as BGP, to perform this function. However, it can also be effective simply to use another IGP such as RIP or OSPF.

Like RIP, EIGRP lets you tag routes that originate from outside of the AS. But, while RIP made it difficult to use this information to make routing decisions, it is relatively straightforward in EIGRP. This information is commonly used to keep traffic inside the AS if any internal path exists. For example, consider a network with two ASes. It might turn out that the shortest path between two segments in the same AS actually passes through the second AS. But usually this is not desirable. After all, what is the point of breaking up the network into ASes if there is no real division between them? In EIGRP, these route tags ensure that if there are two routes for a particular network, one internal and one external, then the internal one is preferred automatically.

Sometimes you might want to use that external route for administrative or cost reasons. More commonly, there might be two or more different external routes for a particular network. For example, there might be more than one Autonomous System Boundary Router (ASBR) connecting to one or more external ASes. In this case the external route with the best metric may not actually be the administratively preferred path.

For these types of situations, Cisco lets you use policy-based routing to act on these route tags. In the simplest implementation, the routers at the edges of the ASes might just add a large delay to one external route.

Although they are intended to act as IGP, IGRP, and EIGRP are sometimes used themselves as EGPs to interconnect OSPF ASes. EIGRP has excellent route-summarization properties, making it also useful for summarizing routes between ASes.

A real EGP such as BGP has better native filtering properties than EIGRP does. Furthermore, because EIGRP is a proprietary standard, it is probably not appropriate for interconnecting the networks of different organizations.

However, two or three OSPF ASes can be easily interconnected within the same organization. This is particularly true if the network designer intends to share all routes between these ASes. EIGRP is extremely simple to configure, and it works well when redistributing routing information with OSPF. So for purely internal uses like this, it may be easier to use EIGRP than BGP to function as the EGP.

Redistributing with Other Routing Protocols

Cisco has made it very easy to distribute routes between EIGRP and other routing protocols. All you need to do is configure a border router that talks to both protocols. Then, in configuring the two protocols, one just instructs each to redistribute routes from the other. But usually AS boundaries serve multiple purposes. It is usually necessary to restrict what information flows between the two protocols.

At a minimum, you must be careful about how the metrics of the external routes look to each protocol. They will be tagged as external routes, so you can always use policy-based routing if you need to. But, as I discussed in Chapter 3, policy-based routing should be used as sparingly as possible. In fact, it is best if used only at the boundary between the two protocols—that is, only on the ASBR routers.

Routers can also set default metrics for all injected external routes in both ASes. This is useful when the two protocols handle metrics in fundamentally different ways, as with RIP and EIGRP. In this case the network designer might want RIP to show all of the external EIGRP routes with a minimum metric of 5. On the EIGRP side, she might want to specify a large minimum administrative delay for all injected RIP routes. Setting default metrics in this way is often the simplest way to control the routes injected from foreign routing protocols.

The situation becomes more complicated when some routes should not be redistributed. For example, there might be a third AS, perhaps running still another routing protocol. Suppose the network has an EIGRP AS connecting to each of two different OSPF ASes. Then the designer decide must if she wants the EIGRP AS to carry traffic between these other two systems. If not, then the boundary router between OSPF AS number 1 and the EIGRP AS can simply refuse to pass along the routing information for AS number 2.

Cisco makes this easy with the use of distribute lists in the EIGRP configuration.

OSPF

Open Shortest Path First (OSPF) uses a Link State Algorithm for finding the best paths through a network. This is a completely different way of looking at dynamic routing than with the Distance Vector protocols discussed earlier. Version 2 of OSPF is the most recent. It is defined in RFC 2328.

Routers running OSPF don't exchange routing tables with one another. Instead, they exchange information about which networks they connect to and the states of these links. This *state* primarily means whether it is up or down, but it also includes information about its type of interface. Every router in the OSPF area (a term that I define shortly) carries an identical copy of this Link State database. The database of links in the network is then used to create a *shortest-path tree*, from which the routing table is calculated.

A simple example should help to explain these concepts. Figure 6-7 shows a simple network that runs OSPF. For now I avoid any questions of media type and IP addressing. These are all point-to-point links. Arbitrary cost is indicated beside each link in the diagram. Some of the links are faster than others. I use a cost of 1 for all-fast links and 10 for the slow ones.

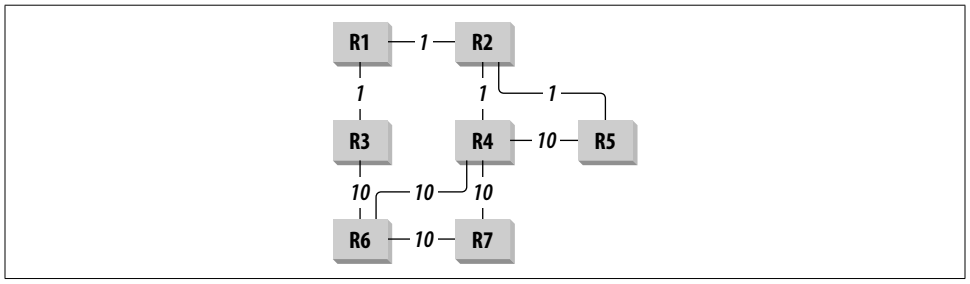

Figure 6-7. A simple OSPF network

Table 6-3 shows the Link State information for this network.

Table 6-3. Link State database

	R1	R2	R3	R4	R5	R6	R7
R1		1	1				
R2	1			1	1		
R3	1					10	
R4		1			10	10	10
R5		1		10			
R6			10	10			10
R7				10		10	

Now OSPF uses this Link State information to construct a shortest-path tree. Even though the Link State information is identical on every router, each one has its own unique shortest-path tree. Figure 6-8 shows the shortest-path tree for Router R6. At first glance it looks like just a redrawing of the same network diagram from Figure 6-8, but it is actually somewhat different.

In particular, although there is a connection from R4 to R7, R6 would never use this link because it has its own links to each of these destinations. Also, the shortest path from R6 to R5 in terms of number of hops goes R6—R4—R5. But the link from R4—R5 is slower than the apparently longer path from R4—R2—R5. Since the shortest-path tree only cares about the links the network will actually use, it shows this more circuitous (but nonetheless shorter in terms of cost) path.

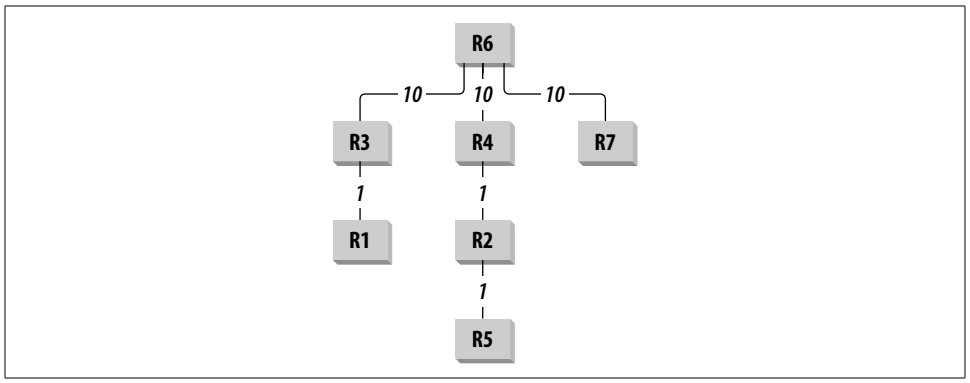

Figure 6-8. Shortest-path tree for Router R6

Every router in the network builds its own shortest-path tree and uses this information to construct its routing tables. Each entry in the routing table indicates the next hop, exactly as it did for the other routing protocols mentioned earlier.

The preceding example was deliberately constructed so that there one would be only one best path to each destination. In any real network this is rarely the case. OSPF provides a mechanism called *equal-cost multipath*. This means that the tree-building algorithm actually discovers and uses these alternate paths. This makes the picture harder to draw, but it works the same way conceptually.

Different vendors have different ways of dealing with equal-cost multipath routing. In most cases there is a configurable maximum number of paths that will be considered. If there are four equal-cost paths to a destination, the router might only use the first two that it discover. Usually this does not cause any problems, but it could result in routing tables that do not look as expected. Consult your router vendor's documentation for details on how it handles equal-cost multipath routing.

OSPF requires that every router in a grouping have the same Link State database. Scaling efficiency dictates that these groupings shouldn't contain more than about 50 routers. This number is far too small to support most large networks. So clearly there must be a mechanism for subdividing OSPF ASes.

This AS has the same meaning as it did for the discussions of RIP, IGRP, and EIGRP. It is a large administrative grouping of routers that all share routing information using a single IGP.

An area is simply a group of routers that all share the same Link State database. The process by which all routers in an area learn the Link State database from one another is called *flooding*.

When a new router connects to the network, it first attempts to establish neighbor relationships with every other router that it can see directly. Most of these neighbors will then become *adjacent*, meaning that they directly exchange Link State information with one another. There are exceptions where routers that are neighbors do not become adjacent, but I discuss this later.

Then the new router sends its current Link State to all of its adjacent neighbors. This Link State information is contained in a Link State Advertisement (LSA). Since every router taking part in this OSPF area needs to see the same Link State database, the neighbors proceed to pass this information along to all of their adjacent neighbors. These neighbors in turn send the new information to their neighbors and so on until every router in the area has updated its database. Meanwhile, the new router also receives the current Link State database from its neighbors. Very quickly every router in the area obtains the new database. They then must recalculate their shortest-path trees and the resulting routing tables.

The fact that every router in an area must have an identical copy of the Link State database poses an important scaling problem with OSPF. The more routers there are in the area, the more different links each router has, and the more memory the Link State database will consume. This is actually the smaller problem, though. A more serious scaling problem comes from the difficulty in calculating the shortest-path tree as the area becomes more and more complicated.

The usual rule of thumb is that no area should contain more than 50 routers. In a simple network design where every router's shortest-path tree is easily calculated, this number can be pushed up. This is particularly true if the routers are all configured with faster processors and extra memory.

However, it is a good idea to keep OSPF areas small and simple. This helps ensure that the network can respond quickly and accurately to topology changes.

In general, routers that are neighbors are also adjacent. But there are places where this is not the case. The exceptions happen for broadcast media like Ethernet segments and Token Rings, as well as for Nonbroadcast Multiple Access (NBMA) media. ATM and Frame Relay networks can be implemented as NBMA, as can some types of wireless networks.

If a broadcast medium such as an Ethernet segment contains several routers, then every router is a neighbor to every other router. This effectively forms a mesh of relationships. As I mentioned earlier in this book, meshes do not scale well. So OSPF allows routers on broadcast and NBMA networks to simplify their relationships by electing a Designated Router (DR) for the segment. They also elect a Backup Designated Router (BDR) to take over if the DR fails. Then every other router on the segment becomes adjacent to only the DR and BDR. This changes the mesh into a star.

The DR handles all flooding of Link State information for the segment. This router does not take on any special role in routing, however. The DR function is only used to make exchange of Link State data more efficient.

If the DR becomes unreachable for any reason, then the BDR automatically takes over for it and becomes the new DR. It remains in this role until it also fails. So in many networks the DR is just the router that has been up the longest. But this is not always desirable. For administrative reasons, sometimes a network designer wants to restrict which routers take on these functions. In this case, it is possible to set an OSPF priority on every router connected to the broadcast or NBMA medium to control the election process.

The router with the highest priority is elected as the DR, and the second highest becomes BDR. However, this election only happens if there is no DR, either because it's a new network or because the DR has failed.

Frequently there are routers that the network engineer does not want as DR for the segment. In this case the priority is simply set to zero.

Area Types

OSPF allows the designer to break up the AS into a number of smaller areas. Between these areas are Area Border Routers (ABR). An ABR controls the flow of routing information between the different areas, while maintaining distinct Link State databases for each.

There are two main types of areas and a number of subcategories. The main distinction is whether an area is capable of acting as a Transit area.

A Transit area carries traffic that originates in a different area (or a different AS) and is destined for still another area. These external destinations may be other areas, or they may even be other ASes, perhaps running different routing protocols. Conversely, a non-Transit area is one that can only carry traffic that either originates or terminates in that area.

The main reason for this distinction has to do with how external routes are summarized. If an area uses summary and default routes for everything external, then other areas can't use it to get to external or other areas. It simply doesn't have sufficient information to allow this kind of flow-through. So a Transit-capable area is one that does little or no summarization.

There are three common options for how this summarization can be done. They are called Stub, Not-So-Stubby, and Totally Stub.

A Stub area is one that uses a summary route for everything outside of the AS. If the whole network is contained in one AS, perhaps with a single default route to the Internet, then a Stub area provides very little benefit. However, Stub areas can be quite efficient in networks that have a large number of external routes.

If any of the routers in the area connect to a different AS, then the area cannot be Stub. However, it is possible to use a Not-So-Stubby Area (NSSA) for this purpose.

NSSA are defined in RFC 1587. This option allows for the summarization of some external routes but not others. If there is a router internal to the NSSA that connects to the external AS, then those external routes are not summarized. Any external routes that originate in a different area are summarized.

Finally, a Totally Stub area summarizes everything from outside of the area. So even routes that are internal to the AS but originates in a different area appear only as summary routes. This can be useful for portions of a network where routers have limited resources. It is also useful when a large number of the links in the area are slow or have high latencies. In these cases the area cannot transmit large amounts of routing information. So it makes sense to summarize everything from outside of the area.

Not all vendors implement Totally Stub areas. This feature originated with Cisco and is not included in any of the RFC documents that define the OSPF standard. Some other vendors have also implemented Totally Stub areas, however. As with all non-standard options, it should be used with caution. All of the routers in any given area must agree on the type of area. So if some routers are not capable of operating in a particular mode, they may be unable to participate in the area.

NSSA and Stub areas, on the other hand, are implemented by nearly every router vendor.

Route summarization in this discussion is similar to how it was used with EIGRP. In a normal non-Stub area, OSPF distributes routing information on every individual subnet, including those in external regions of the network. A summary reduces this information to a small number of routes that describe large ranges of addresses.

For this to work, it must be possible to reach every valid address in this range through a single Access point. In the case of summary routes for networks outside of the AS, the destination must point to the Autonomous System Boundary Router (ASBR). For summary routes of networks in other areas (or where the ASBR is in another area), every router in the area will simply direct traffic to the Area Border Router (ABR).

Because it is a Classless routing protocol, OSPF uses a system of the longest possible match when looking at summary routes. Suppose a packet has a destination of `10.2.3.5`. The router forwarding this packet will look in its routing table to see how to deal with it. It might have a default route of `0.0.0.0`, which it will use as a catch-all in case it can't find a better match. It might also have a summary route for `10.0.0.0/8`. Again, if it can't find a better match, it will use this one. If there are several possible matches, the router will always use the one with the longest mask, which will be the most specific route. In this example, if there is a route for `10.2.3.4/30`, this will be better than any either `10.0.0.0/8` or `0.0.0.0/0`.

Note also that the ABR routers summarize in both directions. The routes from outside of the area are summarized when they are distributed into the area. Similarly, the internal area routes are summarized when the ABR presents them to the rest of the network. So if an area has a summary route of `10.1.4.0/22`, then it is up to the ABR to distribute this summary information to the neighboring area. If it is summarizing this way, then it does not distribute any of the specific routes for this area.

Just as ASes are defined by numbers, areas also have numeric identifiers. Every AS must have at least one Transit-capable area called area 0.

Areas are sometimes called by a single number, and sometimes by numbers written out in the same format as IP addresses. So Area 0 is sometimes written as `0.0.0.0`. It is usually a good idea to have the default route for an AS connected to Area `0.0.0.0`. But this has nothing to do with this naming convention. In fact, the numerical identifiers for areas (except for Area 0) are completely arbitrary. Since every area must connect directly to Area 0, and only to Area 0, there need not be any relationship between the names of different areas.

However, it can make administration and troubleshooting simpler if areas have meaningful names. Some organizations make their area names identical to the summary of networks inside the area. So, if an area can be summarized with the route `10.1.16.0/22`, then the area might be called `10.1.16.0`.

Other organizations choose their area designations to represent administrative information. For example, they might have a group of areas belonging to each of several different divisions of the organization. One of these divisions—Engineering, for example—might be called `5.1.10.0`. Then the Engineering OSPF areas would be called `5.1.10.1`, `5.1.10.2`, and so forth. Meanwhile, the Marketing division might have `10.2.5.1`, `10.2.5.2`, and so forth.

The numbers are completely arbitrary, so it is up to the network designer to come up with a scheme that is meaningful to the organization.

Area Structures

Every AS must have an Area 0. Every other area in the AS must connect directly to Area 0 and no other area. In other words, every OSPF AS is a star configuration. So OSPF lends itself well to hierarchical network design.

The routers that connect one area to another Area Border Routers (ABR). Every ABR straddles the line between Area 0 and at least one other area. Figure 6-9 shows an example of how this works.

There are three areas in this picture. Area 0.0.0.0 is called the backbone or Core area. There are six routers in this area. Four of these routers are ABRs, and the other two are purely internal. Routers that are purely internal to Area 0 are called backbone routers, so I have named them BBR 1a and BBR 1b.

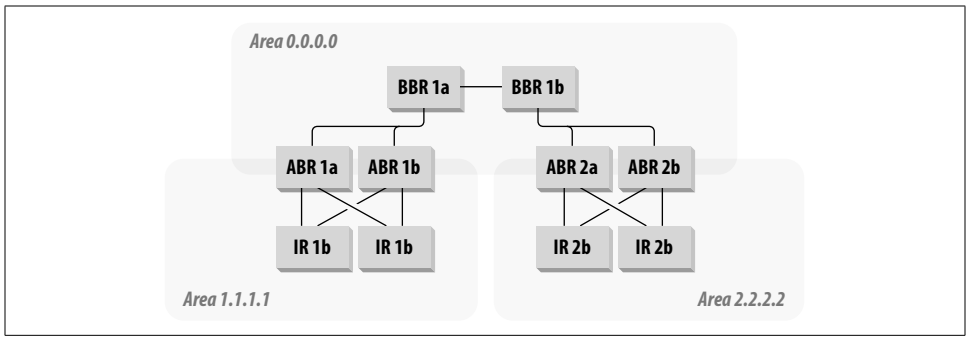

Figure 6-9. Layout of areas in an OSPF AS

There are two other areas indicated in this picture, Area `1.1.1.1` and Area `2.2.2.2`. Each of these areas connects to the backbone area through a redundant pair of ABR routers. Each area also contains two other routers. Routers in nonbackbone areas that are not ABRs are called Internal Routers (IR). Most of the routers in the network will wind up being IRs.

This figure shows multiple ABRs connecting to each area. This is important because it affects how summarization is done. Suppose Area `1.1.1.1` is a Stub area. Then all of the IR routers inside this area will see two types of routes. Any route that originates inside the AS will be a full route with no summarization. But every route from other ASes will be summarized into a default route such as `172.16.0.0/14` or `0.0.0.0/0`.

This summary route will then be distributed by Link State flooding to every router in the area. In the example, both of the internal routers in this area are directly attached to both of the ABRs. So they will see equal-cost multipath routes for these summary routes.

Suppose Area `2.2.2.2` is not a Stub area. Then every router in this area will see all of the full routes originating with every router in the network. They will only see the Link State database for routers in their own area, but they will see routes for everything else.

Figure 6-9 showed two ABR routers for each area. This was done to remove the single point of failure that a single ABR would represent. But it presents a special problem for OSPF to deal with. The two ABR routers must present the same information to the Core. To ensure that they are in synch, it is important to always mesh the ABRs in any one area. The same issues will be even more applicable when talking about ASBRs later in this chapter. This is because ASBRs ensure that summary routes are correct for an entire AS. ABRs only have to summarize an area, but they still need to keep the routing information up-to-date.

In many networks there is an additional reason for meshing the ABRs for any given area. It is common for every IR in an area to have connections to a pair of ABRs. Then if one of these links fails, the second ABR will take over all of the traffic. However, if the ABRs summarize the area routes when passing them to the Core, then the Core does not need to know about this failure inside the area. So, if traffic from the Core to the IR used the ABR with the failed link, the ABR-to-ABR link provides a new path to the IR. Otherwise, every little change in a remote area will cause changes in the routing tables of the backbone. The backbone area should see only routing changes that result from serious problems.

This diagram shows a pair of ABR routers that connect Area 0 to each of the non-Core areas. In fact, if the ABR routers have relatively powerful processors and lots of memory, they act as ABR for a number of non-Core areas.

One could, for example, have a single ABR router with three high-speed interfaces. The first interface connects to Area 0, the second interface connects to Area `1.1.1.1`, and the third to Area `2.2.2.2`. This router then acts as ABR to both areas. There need be no particular relationship between Area `1.1.1.1` and Area `2.2.2.2` in this case. The point is just to economize on the number of ABR routers required.

There are no theoretical limits on how many areas an ABR can support. But there are relatively strict practical limits imposed by CPU performance and memory capacity. Most modern routers can readily handle two areas plus Area 0. Some powerful devices can be ABR for 4 or 5 areas with relative ease. Ask your hardware vendor for guidance before attempting to support multiple areas through a single ABR router. It may require a memory or CPU upgrade.

So far, the benefits to summarization have concerned efficient use of resources. But summarization has another key benefit. If you don't summarize, then you must propagate every route through the network. In particular, if the ABRs don't summarize into Area 0, then they must propagate every individual route into Area 0. This is usually not a problem, but every time a link changes state, the route *flaps*—that is, a Link State advertisement is flooded through the area. When this information crosses into another area, such as Area 0, it also has to update the routing tables in this area.

Normally, this is not a problem. But suppose the circuit that connects a number of routers to an ABR is faulty. Every time this circuit goes up and down, the ABR must send out Link State advertisements for all of the routes that have changed state. If it happens too frequently, it can cause stability problems in the network Core. So summarization is not just a resource issue; it is also a stability issue.

I have one final comment on OSPF Area structures. In vendor documentation and even in the OSPF RFC, you frequently read about Virtual Links. These are effectively routing tunnels that allow physically remote routers to become adjacent neighbors. This is sometimes used when a router needs to be in one area, but is physically located in another.

For example, a network might consist of a chain of four areas in a row. The first area is Area 1, the second is Area 0, and the last two are Areas 2 and 3. Area 0 connects Areas 1 and 2 properly, but there is a problem in getting to Area 3. One solution is to configure a virtual link from Area 0 to the router that connects Areas 2 and 3 together. Then this router becomes an ABR for both of these areas.

It should now be clear that needing to use virtual links is a symptom of a bad design. It is far too easy for a virtual link to break and partition an area. When the area that breaks is Area 0, this is disastrous. I strongly caution against using virtual links. They may make otherwise impossible configurations possible, but they will never make a network stable.

Interconnecting Autonomous Systems

Just as with RIP and EIGRP, it is possible to join OSPF Autonomous Systems. This could happen because two otherwise separate networks need to talk to one another. Or it could be that one AS has to be divided into two or more pieces. Routers that connect one AS to another are called Autonomous System Boundary Routers (ASBR).

Technically, an ASBR can be placed in any non-Stub Area or in any NSSA. However, in a hierarchical design it is usually preferable to place the ASBR routers in Area 0. In principle, routers from anywhere in the AS will want to connect to the ASBR and the network beyond it.

However, if the ASBR is in one of the non-Core areas, then traffic from a different area must travel a potentially large distance to get to the ASBR. This tends to be rather inefficient. Also, if there are multiple ASBR routers connecting to several different ASes, all via different areas, then it can be very difficult to know which default `0.0.0.0/0` route is the best one. However, if the ASBR routers are all located in Area 0, it becomes much easier to keep tight control over external routing.

Finally, if the designer is building a hierarchical network design, then it should be hierarchical at all levels, not just within the OSPF AS. So this concept leads to the idea of a central EGP Core that interconnects a number of IGP ASes. In this view the most natural place to connect the OSPF and EGP clouds is in the Core of the OSPF AS, Area 0.

The one important exception to this is using static routes or a foreign routing protocol such as RIP to accommodate network gear that doesn't support OSPF. It is not uncommon to encounter legacy equipment in outlying portions of the network. In this case it is essentially unavoidable: you need to have an ASBR in an area other than Area 0.

It is important to make sure that this area is a Transit Area. It can be either a non-Stub Area or NSSA. The choice between these two options depends mainly on how much other routing information comes from outside of the AS. If the AS has very few

external routes, then a non-Stub Area is simpler and therefore preferable. But if there are many external routes, then an NSSA should use router resources more efficiently.

Strictly speaking, since OSPF is an IGP, you should interconnect ASes using an EGP such as BGP. It is possible to use another IGP for this purpose, however. IGRP and RIP actually work relatively well for this purpose. However, it is usually not a good idea to interconnect two ASes running the same IGP without some other protocol in the middle. This is essentially to control the flow of IGP information.

I mentioned previously that it is a bad idea to connect two EIGRP ASes directly. OSPF behaves somewhat better in this regard. But it is still good practice to use a foreign protocol in the middle to help control how routes are distributed between the ASes.

There are two reasons for splitting up an OSPF AS. First, it might have grown so large that it no longer converges quickly after a link failure. This is relatively rare, however. More frequently a designer might want to split up an AS to help isolate regions of instability. Whenever a link fails, the route associated with this link must be updated throughout the AS. If the ABR routers for the area containing the failed link give Area 0 summary rather than detailed routing information, then there is nothing to update. But if every route is listed in detail, then this detailed information must be rigorously updated whenever it changes.

Now consider an AS that contains a mixture of LAN and WAN areas. Suppose that a WAN area contains a Frame Relay cloud with a single circuit supporting hundreds of remote sites. If this circuit fails, the ABR for this area must update Area 0 with all of these individual routing updates. When the circuit comes back up, all of the routes must be updated again.

If this happens frequently, it can make the routers in Area 0 extremely busy recalculating their routing tables. That can result in Area 0 itself becoming unstable. So some network designers like to separate their WAN components into one or more distinct ASes that are separate from the more stable LAN components.

Redistributing with Other Routing Protocols

The simplest example of redistributing other routing information into OSPF is the use of static routes. This is effectively the same as redistributing from one AS into another. Every route that does not come from within the AS and is not generated by the standard Link State advertisements is considered an external route and is tagged as such.

When an external route is injected by an ASBR, a cost is associated with it. This need not be a real indication of the number of hops or the speed of links on the outside of the ASBR. In fact, you only need to be careful with the costs of external routes when there are two or more different ASBRs offering connections to the same network. In this case, OSPF adds its own internal costs to each hop through the network.

To reliably control which external path is used in this scenario, all ASBR routers that connect to the external network should be located together in Area 0. Then if one ASBR is preferred, it injects the route with the best cost. If this is not done—for example, if two ASBR routers with the same external routing information are located in different Areas—then predicting which one will be used is difficult. Some routers may use one, and others may use the other ASBR. This may be desired. But it is simpler and easier to maintain if all ASBR routers are located in Area 0.

In fact, OSPF uses two different types of external routes. An arbitrary router inside an AS looking at a Type 1 external route sees a metric equal to the cost for that route at the ASBR, plus the cost required to get to the ASBR. For Type 2 external routes, on the other hand, the internal portion of the cost is ignored.

If there are two ASBR routers injecting the same Type 1 route with the same metric, then each internal router chooses the closer ASBR. But if it is a Type 2 route, then it always picks the same ASBR, regardless of which one is closer. The ASBR it picks will be the one with the best external cost. If the metrics for two Type 2 routes are equal, then the internal distance is used to break the tie.

Where both Type 1 and Type 2 routes exist for a particular network, the internal routers will always select the Type 1 route.

A special case is the injection of an external route that overlaps with address range of the AS. This is generally dangerous. But there are times when a static route must be used because OSPF is not naturally aware of the route. This might happen, for example, if there is foreign equipment in the network that does not run OSPF.

OSPF will always use the most specific route first. So, ven if there is a lower cost route that includes the subnet mentioned in the static route, the specific route will be used. For example, suppose a route to `192.168.5.0/24` is distributed through the normal Link State process. This could be distributed either as a summary route or as a normal route. Suppose there is one particular host, `192.168.5.16/32`, that is connected differently, perhaps through a PPP or SLIP connection directly to a router port. Then this router could inject this host route (a host route has a mask of `255.255.255.255`) with the appropriate metric for this medium. OSPF would then use this host route properly for this specific device and the network route for everything else in the segment. This should work even if the host route has a higher cost than the network route.

IP Addressing Schemes for OSPF

OSPF relies on route summarization to work efficiently. Unlike EIGRP, which allows route summarization at any point, OSPF only summarizes at ABR and ASBR routers. So where EIGRP can benefit from highly sophisticated addressing schemes that summarize on many levels, OSPF can use somewhat simpler IP addressing schemes.

Each AS must be completely summarized by a simple network/mask combination. As mentioned previously, it is always possible to inject external routes that overlap with the internal range. But this should be avoided because it is confusing. If multiple ASes are used, they should all have their own clearly summarized ranges. Then, each area within each AS should be composed strictly of a summarized subgroup from the AS address range.

For example, suppose you have a network with two ASes. The first uses the range `10.1.0.0/16`, and the second uses `10.2.0.0/16`. This will make it easy for the ASBR routers to summarize the links that connect them. Then the areas within the first AS may have address ranges that look like `10.1.0.0/22`, `10.1.4.0/22`, `10.1.8.0/21`, `10.1.16.0/21`, and so forth. Note that these ranges are not all the same size. There is no reason to restrict areas to summarize the same way as one another.

If you fail to create clearly summarized address ranges at the ASBR and ABR boundaries, OSPF has to work much harder than it would otherwise. This is extremely inefficient. It is also very difficult for human engineers to diagnose problems when there is no simple and clear pattern to the IP addresses.

OSPF Costs

Earlier in this chapter I indicated that the OSPF cost values are arbitrary. They are used to select the best paths through a network. So, in general, faster links will be configured to have lower costs. In fact, if you assume the same latency for every type of link (which is not true in reality), then you can define the cost to be inversely proportional to the bandwidth.

This leads to one of the most popular methods for setting OSPF costs. You can take a reference bandwidth as the fastest link in the network and make its cost 1. Then every slower link has a cost that is just the reference bandwidth divided by the slower link's bandwidth. If your reference bandwidth is a Gigabit Ethernet link in the network's Core, then every Fast Ethernet (100Mbps) link will have a cost of 10, 10Mbps Ethernet links will have 100, and a T1 (1.544Mbps) will cost 6476.

This is a relatively good system, but it has one critical flaw that makes it unworkable in many networks. The maximum value for an OSPF cost is 65,535. In fact, it is important to avoid coming anywhere close to this value because a path that includes such a link plus any number of faster links could easily have a total cost greater than the maximum. When this happens the entire path becomes unusable. This is effectively the same problem as when a RIP metric exceeds 15.

The problem is that many networks include too large a range of bandwidths. Suppose, for example, that the fastest link in the network is a Gigabit Ethernet link, and the slowest is a 9.6kbps dialup line. If the Gigabit link has a cost of 1, this implies that the 9.6kbps line must have a cost of 104,166, which is considerably larger than

65,535. This problem becomes worse in a network with a 10Gbps link in its Core, because then even relatively common 56kbps circuits have excessively high costs.

Let's revisit the reasoning behind this standard linear rule to adapt it to these real networks. The range of bandwidths available forces many network designers to use a non-linear rule. Certainly, the faster links must have lower costs than slower ones. But do links that are one-tenth as fast really need to bear a cost that is 10 times as high? This would make the net cost of a path passing through nine Fast Ethernet links better than the cost of a single 10Mbps Ethernet link. Is this realistic?

The main problem is what nonlinear method to use to include the bandwidth factor. What it really comes down to is deciding how many hops through high-speed links equals one slow hop.

Clearly, an important factor is the latency of each of these links. The latency for a short 10Mbps Ethernet is roughly governed by the length of time required to wait for carrier and inject the packet. Time of flight to the farthest part of the segment is less than the time to transmit the entire packet (or else the segment will suffer from late-collision problems). The same is true for 100Mbps Ethernet, but because the carrier frequency for 100Mbps Ethernet is 10 times as fast, the latency should be roughly one-tenth as long.

Adding more hops to the path also increases the latency because each router in the path takes some time to process the packets. In the case of Ethernet and Fast Ethernet, the amount of work is almost exactly the same. So assume that each router adds roughly the same additional latency as a Fast Ethernet segment does. Then passing through N Fast Ethernet hops will add N link delays plus N–1 router delays, for a total of 2N–1. This implies that the break-even point based on latency alone will be when 2N–1 = 10, or N = 5.

Now consider how the bandwidth should scale neglecting latency effects. Nominally, if a link is 10 times as fast, then an application can send 10 times as much data through it. But this assumes that this application is the only one using this link. In fact, the faster links usually aggregate traffic from a number of slower links. The amount of competition for the bandwidth on some remote link depends on the network design and traffic patterns. Generally speaking, these links have some constant utilization for which many devices compete, plus excess capacity that they can use fairly freely.

Putting these factors together suggests a simple formula with the cost inversely proportional to the square root of the nominal bandwidth. Note that a great deal of hand waving went into finding an appropriate formula. It is balanced so that a Fast Ethernet link is roughly three times as good as a 10Mbps Ethernet link. Similarly, Gigabit Ethernet links are roughly three times as good as Fast Ethernet. This simple rule scales the same way throughout the entire range. Best of all, it results in usable cost numbers for the slowest links in a network, as shown in Table 6-4.

Table 6-4. Suggested OSPF cost values for different media types.

Medium	Nominal bandwidth	Cost in 1/bandwidth model	Cost in 1/square root model
9.6kbps line	9.6kbps	1,041,666[a]	1020
56kbps line	56kbps	178,571[a]	422
64kbps line	64kbps	156,250[a]	395
T1 Circuit	1.544Mbps	6,476	80
E1 Circuit	2.048Mbps	4,882	69
T3 Circuit	45Mbps	222	14
Ethernet	10Mbps	1,000	31
Fast Ethernet	100Mbps	100	10
Gigabit Ethernet	1Gbps	10	3
10 Gigabit Ethernet	10Gbps	1	1
4Mbps Token Ring	4Mbps	2,500	50
16Mbps Token Ring	16Mbps	625	25

[a] These costs are all higher than the maximum cost value of 65,535, and they would be adjusted in practice.

Table 6-4 also includes the costs that result from using the more common model in which cost is inversely proportional to bandwidth. In both cases I adjusted the costs so that the fastest link, the 10Gigabit Ethernet, has a cost of 1.

Both of these models are just basic suggestions for starting points. The network designer should carefully consider the OSPF costs of every link in the network to ensure that they are appropriate. Poor choices of values can lead to serious traffic routing problems. But, as with all network design problems, simple consistent rules will usually result in a more stable network.

It is particularly important to include room for growth. If there is even a remote chance that your network will one day include Core links that are faster than the 10 Gigabit Ethernet speed suggested in this table, make sure to scale all of the cost values up accordingly. Then the new fastest link will have a cost of 1, and all of the other links will be correspondingly more expensive. Making this change after a network is built can be time consuming and highly disruptive.

There is an interesting exception to the preceding comments. OSPF areas are configured so that only Area 0 carries traffic from one area to another. If a packet starts in any area besides Area 0 and then leaves that area, then it cannot return to the area in which it started. If it then passes into another area, then it must have its ultimate destination in that area. So the problem of selecting the best path breaks up into components. First, OSPF needs to find the best path through the originating area. If the destination is in the same area, then it needs the best path to the destination. But if the destination is in some other area, then all it cares about is finding the best path to Area 0.

Once the packet is in Area, 0 OSPF needs to find the best path within this area. It may terminate in Area 0, or it may lead to an ABR for another area. Finally, in the destination area it needs to find the best path to the final device. But the point is that the protocol does not need to know the entire path from end to end unless the path is contained entirely in one area. It just needs the best path to the next ABR.

Consequently, it doesn't matter if you use different costing rules in different areas. For example, Area 0, being the Core of the network, might contain several 10 Gigabit Ethernet links. But it is unlikely that this area will contain anything slower than a T1 circuit. So you can use one set of costs appropriate to this range of bandwidths. Similarly, a destination area might contain a number of remote WAN sites connected via 56kbps circuits. But as long as the fastest links in this area are 100Mbps Fast Ethernet, you can use a consistent set of costs based on 100Mbps bandwidth. However, as with all aspects of network design, it is preferable to have a single common rule that applies everywhere. So this exception is best used only as an interim measure while readjusting metrics throughout an AS.

BGP

Border Gateway Protocol (BGP) is currently in its fourth version, which is defined in RFC1771. Although the Core protocol has not changed since 1995, there have been some additions to it.

BGP is an EGP. All of the other routing protocols that I have discussed so far in this chapter are IGP. In the usual configuration, IGP protocols function purely within an AS, while EGP protocols are used to interconnect ASes. The main exception to this rule is that sometimes an IGP can be used in a limited function to link together two ASes running a different protocol. For example, you can link two OSPF ASes using EIGRP or RIP. However, using a real EGP offers many important advantages.

BGP is by far the most popular EGP. There is an earlier EGP protocol called, confusingly enough, EGP. But BGP is much more robust and offers many more useful features for policing traffic flows. Most importantly, BGP Version 4 allows fully classless routing.

Classless routing is important because there are many cases where organizations want either to subnet or to supernet their address ranges. I have already discussed subnetting. Supernetting is a similar idea, except that it allows an organization to group together a number of contiguous smaller class networks.

For example, suppose an organization uses four unregistered Class C networks, `192.168.4.0/24`, `192.168.5.0/24`, `192.168.6.0/24`, and `192.168.7.0/24`. They could distribute routing information to these four addresses by means of the supernet route `192.168.4.0/22`. Similarly, two different organizations might opt to share the Class B range `172.19.0.0/16`. So one could use `172.19.0.0/17`, and the other `172.19.128.0/17`. They could then exchange all of their routing information with a single simple summary.

This feature, called Classless Interdomain Routing (CIDR), is becoming relatively common throughout the Internet. This is because the growth of Internet participation has led to a drastic shortage of IP addresses. So the IETF has been forced to get creative with its address allocation. Since BGP is the primary routing protocol for interconnecting organizations on the Internet, it has become completely classless.

The Internet presents some interesting challenges to a routing protocol. There are some regions of the Internet that share a common backbone. However, internationally the Internet is best viewed as a completely arbitrary collection of interconnected networks.

A large number of individuals and organizations connect to the Internet through one or more Service Provider networks. These Service Provider networks in turn connect with one another and with high-speed Internet backbone networks that are themselves essentially just fast Service Provider networks. In addition, there are a number of educational and governmental organizations that behave almost like Service Provider networks by acting as interconnection points for a number of other networks.

Internally, each Service Provider network may use any routing protocol to distribute routing information. Also internally, these networks form one or more ASes. When connecting to other networks—either client networks or other Service Providers—these networks use BGP to share routing information.

So BGP must share routing information between ASes. It must also summarize information about what routes lie behind each AS. Devices on my network need to get to some distant part of the world by first passing through my Service Provider. My Service Provider needs to know that it can reach this network through some other Service Provider, and so forth, until my packet finally reach its destination.

In short, BGP functions not router to router, but AS to AS. It resolves loops and finds the shortest path in AS-sized chunks.

BGP also has another more complex role to play in AS-to-AS routing. Some organizations might want to take part in the Internet (or, for that matter, any shared IP network). But they might not be willing to act as a conduit for traffic between other organizations. So BGP has a filtering function that allows considerable control over what routes are distributed to which AS neighbors.

BGP uses a different routing algorithm than either RIP or OSPF. Like RIP, it only keeps track of information about the nearest hops and the routes that can be reached through them. But unlike RIP, it doesn't use a simple metric to decide which path is the best. Instead, it maintains information about the entire path to the destination. The method for doing this is called a Path Vector Algorithm.

This means that every route that BGP knows about is accompanied not by a simple number representing cost or distance, but by the actual path. It is easy to avoid loops when you can look at the whole path and see that the same intermediate step appears more than once. The path is not a sequence of routers, but a sequence of ASes, which is why it is called AS_PATH.

If Autonomous Systems exchange routing information using BGP, then each one must have one or more routers that speak both BGP and the IGP. These routers are called Autonomous System Boundary Routers (ASBR). Figure 6-10 shows how three ASes might be connected.

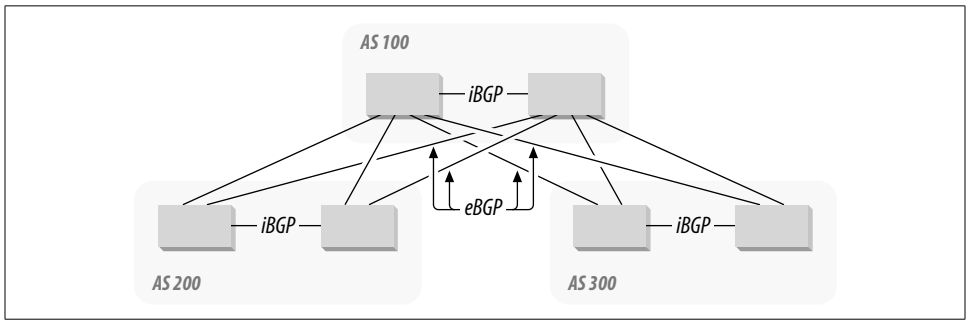

Figure 6-10. Interconnecting three ASes using BGP

Figure 6-10 shows two BGP ASBR routers in each AS for redundancy. These ASes could be running any combination of IGP protocols, such as OSPF, RIP and EIGRP. The two ASBR routers inside each AS communicate with one another using iBGP, the interior protocol. ASBR routers in different ASes use eBGP, the exterior protocol. This is an important distinction because two routers that provide access to each AS must present a unified picture of what is inside. This means they share a common view of the interior of the AS, and they also share the most up-to-date information about all of their AS neighbors.

You have to configure a full mesh of iBGP connections between all of the ASBR routers in each AS. Every ASBR connection, whether iBGP or eBGP, uses a TCP session to exchange routing information. This TCP session is not discovered, but must be manually configured on each router, and it then remains permanently active.

However, as mentioned earlier in this book, fully meshed networks do not scale very well. So there have been some additions to the BGP protocol that aim to relax this requirement. The most important of these protocol additions are Route Reflection (discussed in RFC 2796) and BGP AS Confederations (RFC 3065). These documents are both relatively recent, so not all router vendors have implemented their recommendations. Even those that have incorporated these options have only done so recently, so older equipment may not support them.

There are many different ways of connecting ASes. An AS may have only one ASBR and connect only to one other AS. In this case the routing information that it conveys to the rest of the world only concerns its own IP addresses. This is similar to an AS that has multiple Access points but does not allow traffic to pass through it. These are both called nontransit ASes.

A third option is an AS that has multiple Access points and allows traffic to pass through it. This is called a transit AS. In Figure 6-10, there is no connection between AS 200 and AS 300. To allow devices in these two networks to communicate with one another, AS 100 must pass along routing information received from each to the other. However, AS 200 and AS 300 only need to pass along their own summary routing information into AS 100.

A useful feature of BGP is the ability to restrict what routing information is conveyed. This in turn has the effect of restricting whether a particular AS is used for transit between other ASes. So, for example, an AS might be configured to provide transit services for some external networks, but not others. This can be done either per-network or per-AS. It might only pass transit information to some of its downstream neighbors.

Autonomous System Numbers

Since BGP uses AS Numbers to identify routing elements, there must be rules for how these AS Numbers are allocated. If BGP is to work on the public Internet, then clearly two organizations can't both use—for example, AS Number 100. A conflict in AS Numbers is as serious as a conflict in IP addressing. AS Numbers detect routing loops, so they must be globally unique.

The standard rules for allocating AS Numbers are defined in RFC 1930. These rules apply to all IP networks and to all routing protocols. The range from 64,512 to 65,534 (and possibly also 65,535) is reserved for private networks. These AS Numbers cannot be advertised on the public Internet. This is similar to the private use of unregistered IP address ranges such as `10.0.0.0/8`. So it makes a great deal of sense to use AS Numbers from this range particularly for any AS that uses unregistered IP addresses. This way neither the addresses nor the AS Numbers will ever be in danger of reaching the public Internet.

The AS Numbers from 1 through 22,527 have been divided up among three main international Internet standards organizations* to allocate to networks that connect to the public Internet. Of the remaining numbers, 0 and the range from 22,528 through 64,511 are currently held in reserve by the IANA for future purposes. There is some inconsistency between IANA documents and RFC 1930 in the availability of AS Number 65,535. The IANA indicates that this number is reserved, while RFC 1930 lists it as part of the unregistered range. So it is probably best to avoid using 65,535 to avoid possible future compatibility problems.

* In the Americas, Caribbean, and sub-Saharan Africa, ARIN (American Registry for Internet Numbers, *http://www.arin.net*) is responsible for allocating all AS numbers. In Asia and the Pacific region, this is done by APNIC (Asia Pacific Network Information Centre, *http://www.apnic.net)*. RIPE NCC (Réseaux IP Européens Network Coordination Centre, *http://www.ripe.net*) allocates these numbers for Europe.

However, in most cases it is easiest to just contact your Internet Service Provider (ISP) for assistance with registering AS numbers. They should be able to help in deciding whether officially registered AS numbers are required, or whether it is possible to get away with using only unregistered numbers.

Where to Use BGP

BGP is useful anywhere two or more ASes need to exchange routing information dynamically. If the information never changes, then it is considerably simpler to just use a static route.

Many factors lead to networks requiring continuously updated routing information. For example, there might be more than one way to get to a distant AS. Figure 6-11 shows four ASes. To get from AS 100 to AS 400, a packet can go through either AS 200 or AS 300. It might have an administrative reason for preferring one of these paths, but if the preferred path becomes unavailable, it will need to switch to the other.

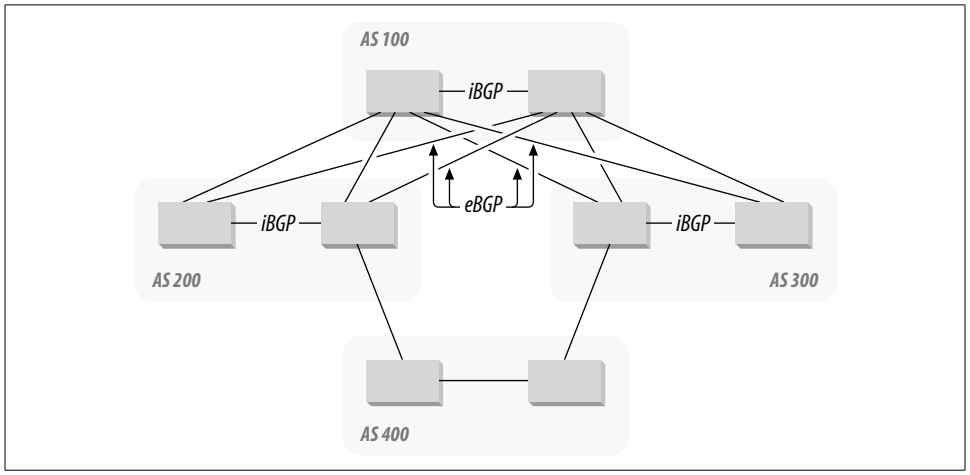

Figure 6-11. A simple network having multiple AS paths

However, there is a simpler reason for needing to use a dynamic EGP protocol in Figures 6-10 and 6-11. Both of these cases have multiple ASBR routers. For example, there are two ASBR routers in AS 100 and AS 200. There are then four paths between these two ASes. A static route would not allow the network to use these paths for redundancy.

BGP is unnecessary in the simple example of one AS connecting to the Internet via a single Service Provider. In this case the ISP can easily handle all inbound traffic with a single static route that summarizes the block of registered addresses for this client network. All outbound traffic is handled similarly by directing the route `0.0.0.0/0` to the Service Provider's network. This is the configuration that most organizations use to connect to the public Internet. So most of these organizations do not need to use BGP for their Internet connections. Consequently, they do not need to register an AS Number.

BGP becomes useful to the client, however, when the network uses two or more different ISPs. Then they can configure BGP to give redundancy in the Internet connection. In this case the network needs a registered AS Number, even if the client network is not configured to allow transit from one ISP to another. In this case the designer will want to configure BGP to distribute the routes for the internal AS only. This will prevent the client network from becoming transit capable.