CHAPTER 8

Scaling

Scalable may well be one of the most overused words in the network designer's lexicon. It applies to protocols, to software, to operating systems, to hardware architecture, and to networks. Scalable means, simply enough, the ability of the relevant entity (select one from the list just given) to get much bigger than it presently is without reducing performance, stability, or accuracy; making your customers angry; or getting you fired.

You have already encountered in this book a number of features that make OSPF and IS-IS scalable from the smallest to the largest networks. The most prominent feature for both protocols is areas. By dividing a network into multiple areas, you can control in each area the scope of flooding, the size of the link state database, and the complexity of the SPF calculations. By bounding these three fundamental link state functions, you constrain the demand OSPF and IS-IS put on router memory, router processing cycles, and link bandwidth, respectively.

You have also encountered features or extensions that can increase scalability, such as configurable refresh timers and increased metric sizes. This chapter delves further into features and extensions that can increase the scalability of the protocols.

8.1 SPF Enhancements

Chapter 2 gave you an idea of how an SPF algorithm works. And while that overview is enough to give you an understanding of the SPF algorithm, it is too simplistic for a practical link state protocol implementation. In fact, even if you follow the OSPF and IS-IS standards, you will get an implementation that is workable (if naïve) for small networks, but that lacks the stability, scalability, and accuracy necessary to survive in large-scale networks. For router vendors wanting to market to the largest network operators, sophisticated OSPF and IS-IS implementations are essential. Demonstrably robust implementations are a competitive advantage, and as a result, many of the enhancements to the SPF algorithm—and in fact the

SPF algorithm itself—can be closely guarded corporate secrets. Therefore, it is impossible to detail many vendor-specific enhancements, but the general areas of enhancement can be examined.

8.1.1 Equal-Cost Multipath

The basic Dijkstra calculation described in Chapter 2 uses the following steps:

1. The router adds itself to the tree database as the root of the tree. It shows itself as its own neighbor, with a cost of 0.
2. All entries in the link state database describing links from the root to its neighbors are added to the candidate database.
3. The cost from the root to each node in the candidate database is calculated. The link in the candidate database with the lowest cost is moved to the tree database, along with the cost from the root. If two or more links are an equally low cost from the root, choose one. If any entries are left in the candidate database with a link to the neighbor just moved to the tree, those entries are deleted from the candidate database.
4. The router ID of the neighbor on the link just added to the tree is examined. Entries originated by that neighbor are added to the candidate database, except for entries in which the ID of the neighbor is already in the tree database.
5. If entries remain in the candidate database, return to Step 3. If the candidate database is empty, terminate the calculation. At this time, every router in the network should be represented as a neighbor on one of the links in the tree database, and every router should be represented just once.

However, there is an inefficiency in Step 3. Namely, "If two or more links are an equally low cost from the root, choose one. *If any entries are left in the candidate database with a link to the neighbor just moved to the tree, those entries are deleted from the candidate database*." An example is needed to understand why this is an inefficiency. Figure 8.1 shows a small network and its link state database. What is important about this network is that unlike the network in Figure 2.15 that was used for the SPF example in that chapter, all of the links in this network have the same cost. This example will show the SPF calculation at R1.

Steps 1 and 2 are shown in Figure 8.2. R1 adds itself to the tree database, the links to its neighbors are added to the candidate database, and the costs from the root to the destination nodes are calculated.

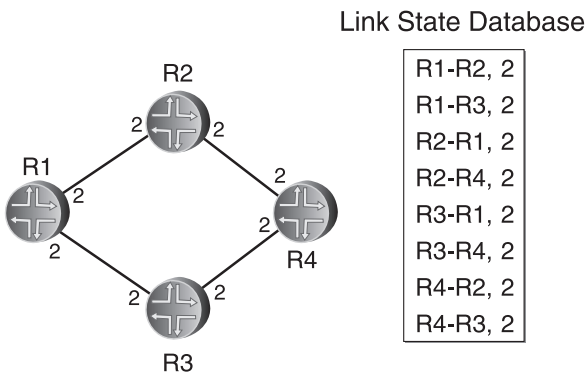

Figure 8.1 All links in this network have a cost of 2.

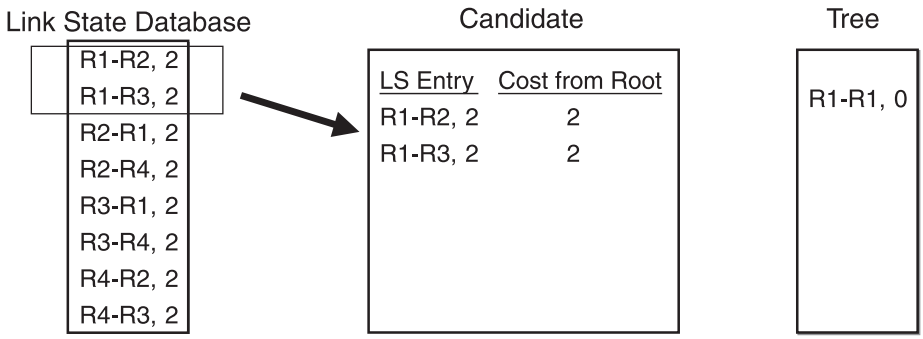

Figure 8.2 The links from R1 to its neighbors are added to the candidate database.

Step 3 is performed in Figure 8.3. The costs to the root are equal, so one of the entries is randomly chosen and moved from the candidate database to the tree database. The inefficiency in Step 3 will not be apparent until the end of the SPF calculation.

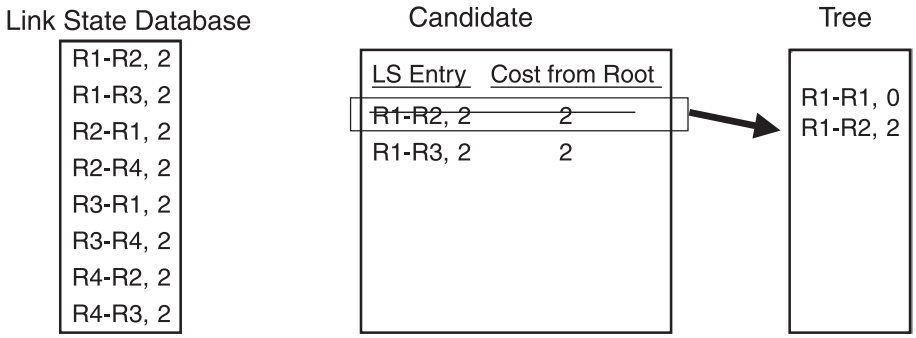

Figure 8.3 The link from R1 to R2 is moved to the tree database.

Step 4 is performed in Figure 8.4. R2 was added to the tree database, so all the links from R2 to its neighbors are examined in the link state database. R1 is already in the tree database, so only the link from R2 to R4 is added to the candidate database. The cost from the root to R4 is 4.

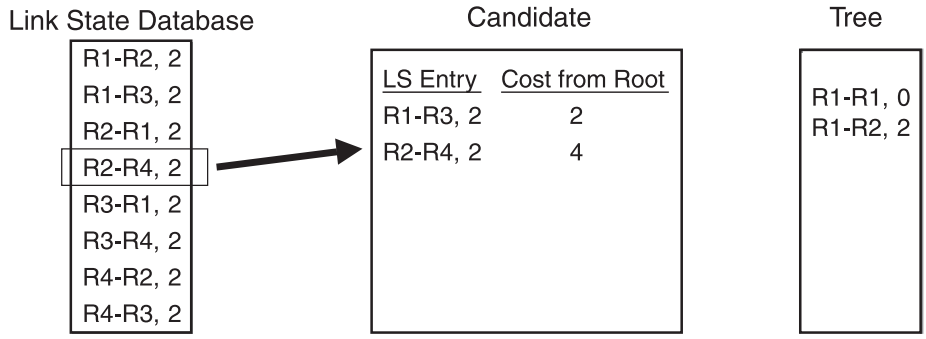

Figure 8.4 The link from R2 to R4 is added to the candidate database.

There are entries remaining in the candidate database, so, as Step 5 prescribes, Figure 8.5 shows a return to Step 3. The cost to root associated with the R1-R3 link is the lowest, so that link is moved from the candidate database to the tree database.

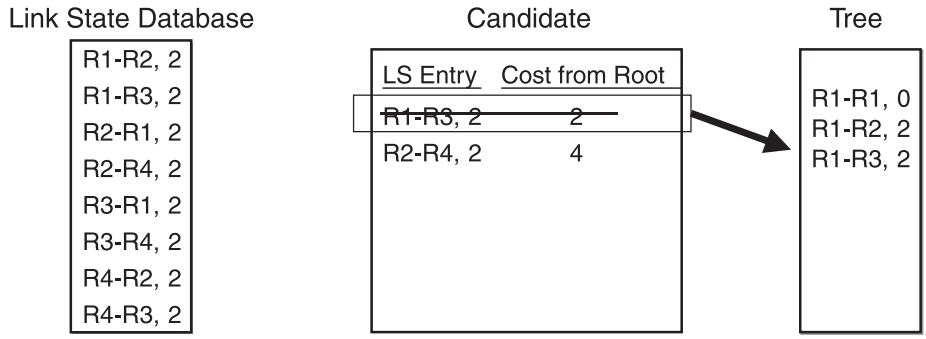

Figure 8.5 The R1-R3 link is moved to the tree database.

With R3 added to the database, the links to its neighbors are examined in the link state database. R1 is already in the tree database, so only the R3-R4 link is added to the candidate database in Figure 8.6. The cost from the root to R4 is 4.

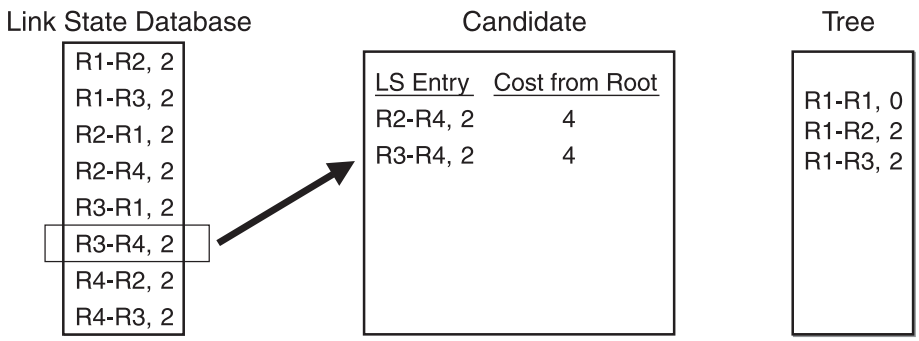

Figure 8.6 The R3-R4 link is added to the candidate database.

In Figure 8.7, the cost from the root of both entries in the candidate database is 4, so one (R2-R4) is randomly chosen and moved to the tree database.

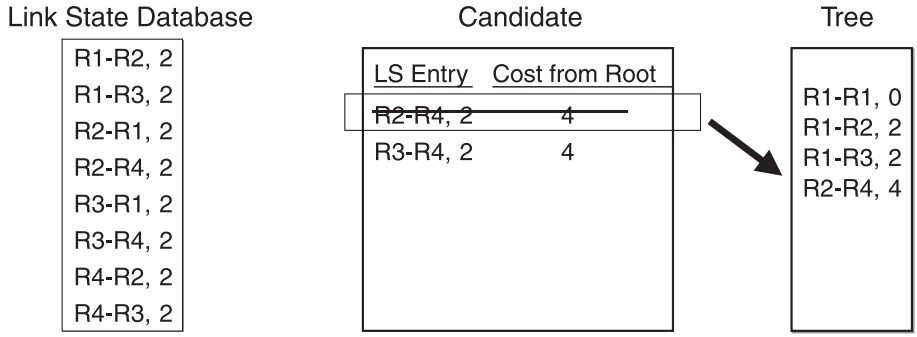

Figure 8.7 The R2-R4 link is moved to the tree database.

In Figure 8.8, the link state database is examined for R4's links to its neighbors: There are two, R4-R2 and R4-R3. However, both R2 and R3 are already in the tree database, so there are no new entries to add to the candidate database. But there is still an entry in the candidate database, and Step 5 says the SPF calculation stops only when the candidate database is empty. This is the reason for the part of Step 3 that says, "If any entries are left in the candidate database with a link to the neighbor just moved to the tree, those entries are deleted from the candidate database." If the "leftover" entries in the candidate database are not removed, the SPF calculation cannot end.

But the inefficiency of this rule also reveals itself at this point. If multiple equal-cost paths exist, choosing only one of the paths means all traffic will be routed on that path, possibly congesting it, while other equally good paths are ignored. We can modify Step 3 so that if there are multiple links in the candidate database to the same node and equally low cost, all the links can be moved from the candidate database to the tree database. In Figure 8.9,

moving the R3-R4 link to the tree database empties the candidate database so that the SPF can stop. An enhancement to the router's forwarding processes can then be made that takes advantage of these multiple equal-cost paths by spreading traffic across all of them. This is *equal-cost multipath* (ECMP), or *load balancing*.[1]

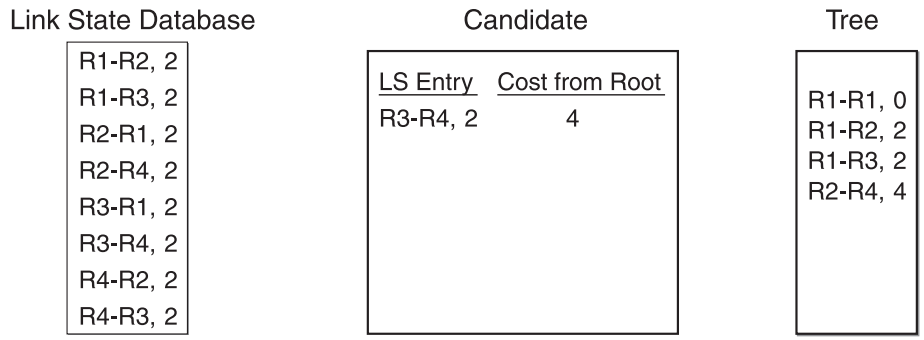

Figure 8.8 The R3-R4 link remains on the candidate database and must be removed before the SPF calculation can end.

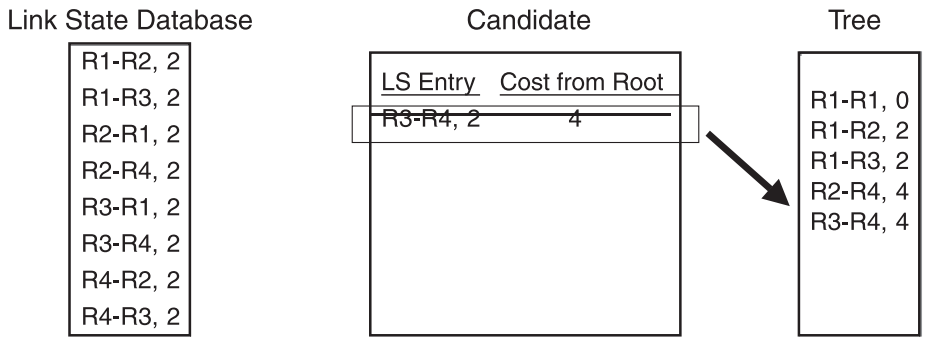

Figure 8.9 By moving the R3-R4 link to the tree database rather than just deleting it from the candidate database, an SPF tree is created in which two equal-cost paths from R1 to R4 exist.

Figure 8.10 shows one situation in which ECMP can apply. In this topology, the next hop from Router A to subnet 10.1.1.0 is Router B. But there are three links to Router B, each with a cost of 50. Router A can balance the traffic going to 10.1.1.0—and any other destination for which Router B is the next hop—across the three links.

[1] *Load balancing* is a more generic term that can also refer to data processing, such as spreading traffic across multiple Web servers. ECMP is more specific to the forwarding of network traffic.

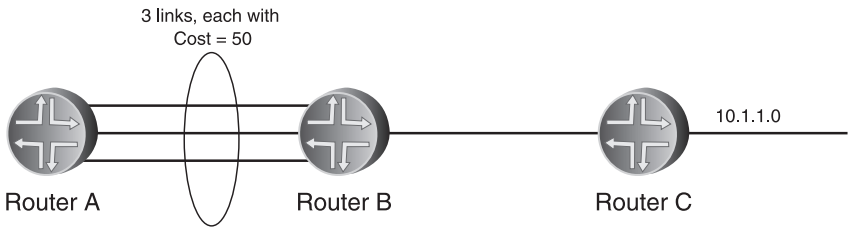

Figure 8.10 ECMP allows Router A to utilize all three links to Router B for traffic going to subnet 10.1.1.0.

Figure 8.11 shows another application of ECMP. Here, Router A has two next hops—Router B and Router C—to subnet 10.2.2.0. Routes to the destination through either next hop have the same cost because all four of the links shown have the same cost. So in this case, Router A can select both next hops and balance the traffic to 10.2.2.0—and any other traffic to destinations that pass through Router D—between Routers B and C.

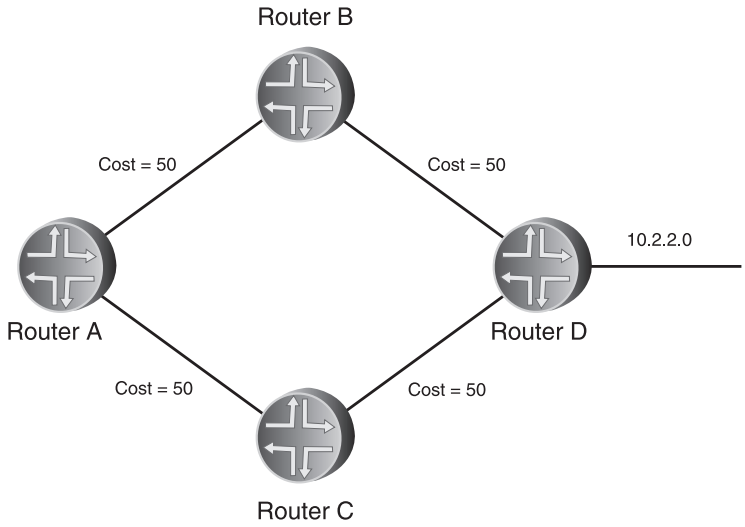

Figure 8.11 ECMP allows Router A to use two equal-cost routes to subnet 10.2.2.0.

Given multiple equal-cost paths—either to the same next-hop router as in Figure 8.10 or through separate next-hop routers as in Figure 8.11—a router can balance traffic across them in several ways. One approach is *per-packet* load balancing, in which the router forwards packets round-robin or randomly over each path as shown in Figure 8.12. The problem with this approach is that different links, although of equal cost, might not be of equal delay. Link propagation, router latency and buffering, and link MTUs requiring fragmentation might differ from one path to another. As a result, packets can become reordered at the receiving end. This can cause problems for TCP, which when it sees out-of-sequence packets can request retransmits. The result is reduced performance and wasted bandwidth.

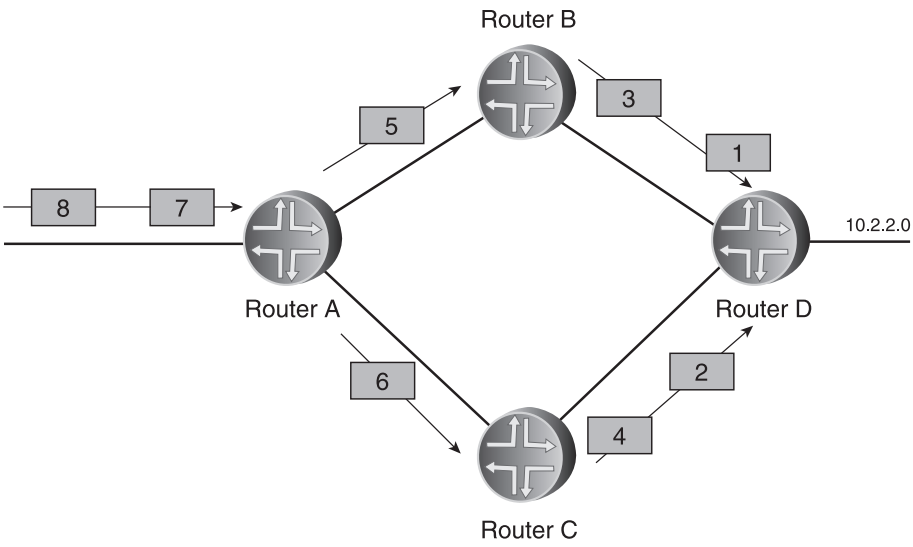

Figure 8.12 Per-packet load balancing distributes packets individually over equal-cost paths.

A better approach is *per-destination* load balancing, in which packets are distributed across multiple paths by destination, as shown in Figure 8.13. In this example, Router A knows it has two equal-cost paths to the destinations 10.1.1.0 and 10.2.2.0. It chooses Router B as the next hop for the route to 10.1.1.0, and Router C for the route to 10.2.2.0. As Router A discovers more routes to destinations reachable through Router D, it continues to alternately assign Router B and Router C as next hops. The functional difference is that per-packet load balancing assigns all ECMP next hops to each route, whereas per-destination load balancing selects, either round-robin or randomly, one next hop from the set of ECMP next hops for each route.

Although per-destination load balancing is an improvement over per-packet, it is still not very good. If substantially more packets are sent to one destination than to another, the overall bandwidth utilization will be uneven. For example, suppose Router E in Figure 8.14 is providing Internet access. When it advertises BGP routes to the other routers in the figure, the routes are given a next-hop address of 10.255.0.1, Router E's loopback address. This is common practice; Router E is advertising that it is the next hop for any Internet destinations. When Router A receives these BGP advertisements, it does a lookup of 10.255.0.1 in its IGP routes to find out how to reach the BGP next hop. If the router is performing per-destination load balancing, it selects either Router B or Router C as the "next hop to the next hop"—that is, the IGP next hop to the BGP next hop. The potential problem is that if traffic passing through Router A to the Internet is relatively heavy compared to other traffic passing through Router A to Router D, the loading across the equal cost paths can be severely unbalanced.

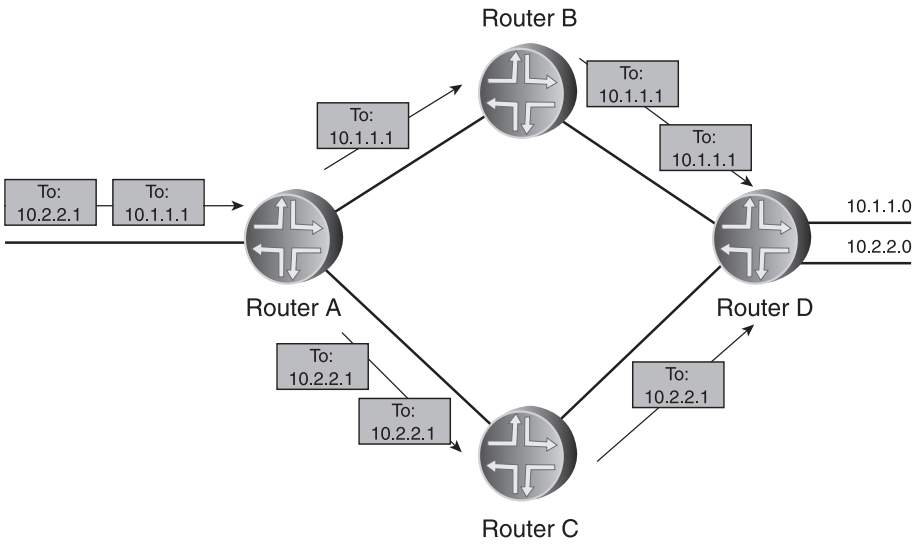

Figure 8.13 Per-destination load balancing distributes next hops or outgoing interfaces among multiple equal-cost routes.

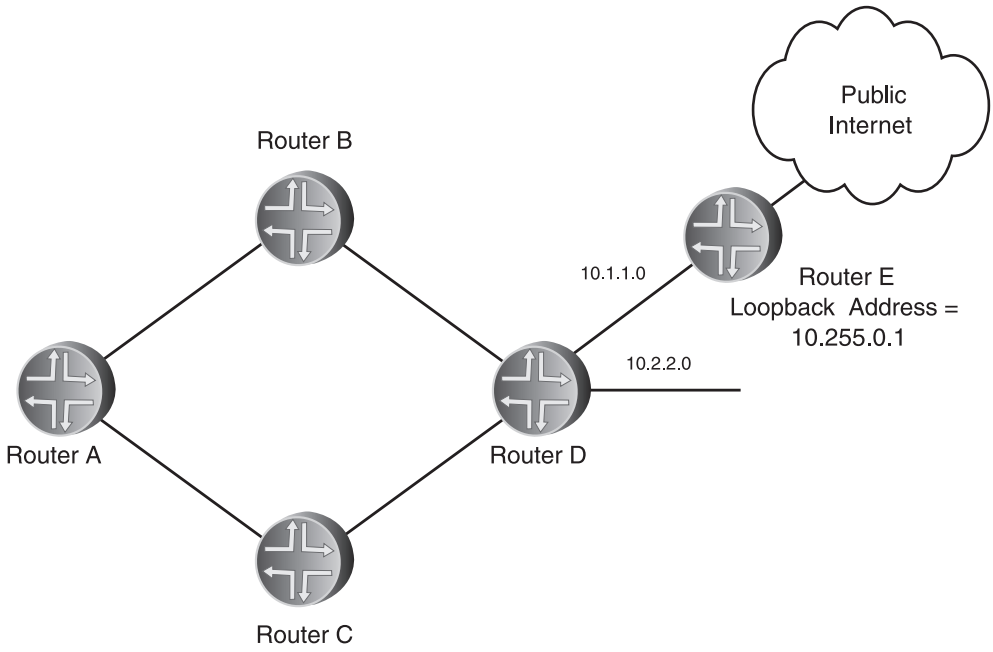

Figure 8.14 Per-destination load balancing might be inefficient in this topology.

Per-flow load balancing improves the traffic distribution by identifying individual traffic flows and forwarding all packets belonging to the flow to the same next hop or out the same interface. At a minimum, a flow might be defined as all packets with the same source and destination addresses. However, flows can be made more granular by defining a flow not only by source and destination address but also by protocol number, source and destination port,[2] and perhaps ToS or DSCP (differentiated services code point) values. When a packet is to be forwarded, the values of these fields become the input to a hashing algorithm. (Router vendors define their own hashing algorithms.) The packet is then associated with a flow based on the resulting hash value, and all packets belonging to the same flow are forwarded to the same next hop.

8.1.2 Pseudonodes and ECMP

If some but not all of the links in an ECMP group are broadcast links, a subtle problem can arise from simplistic SPF implementations. Another example SPF run will help you understand the problem.

The physical topology in Figure 8.15 shows two routers, R1 and R2, interconnected with both a point-to-point link and an Ethernet link. The outgoing cost on both of R1's interfaces is 2, so the two links comprise equal cost paths, and load balancing can be performed between R1 and R2. The logical topology shows that the Ethernet link is represented as a pseudonode, which is labeled in this example as P3. Recall from the discussion of pseudonode basics in Section 4.4 that the cost from a pseudonode to any of its attached routers is 0, so that the pseudonode does not affect the transit cost of the broadcast link. Figure 8.16 shows the start of the SPF calculation at R1 for this small network, in which R1 installs itself in the tree database as the root and then adds all links to its neighbors to the candidate database.

The two links added to the candidate database are of equal cost (2) to the root, so as in previous examples, one is chosen at random and added to the tree database (Figure 8.17). In this case, the chosen link is R1-R2. Although not yet apparent, this random choice will cause a problem with ECMP.

[2] When source and destination ports are included in the flow identification, the flow is called a *microflow*. Implementations that load balance by microflow are more challenging to create because they require the router to look further into the packet than just the IP header.

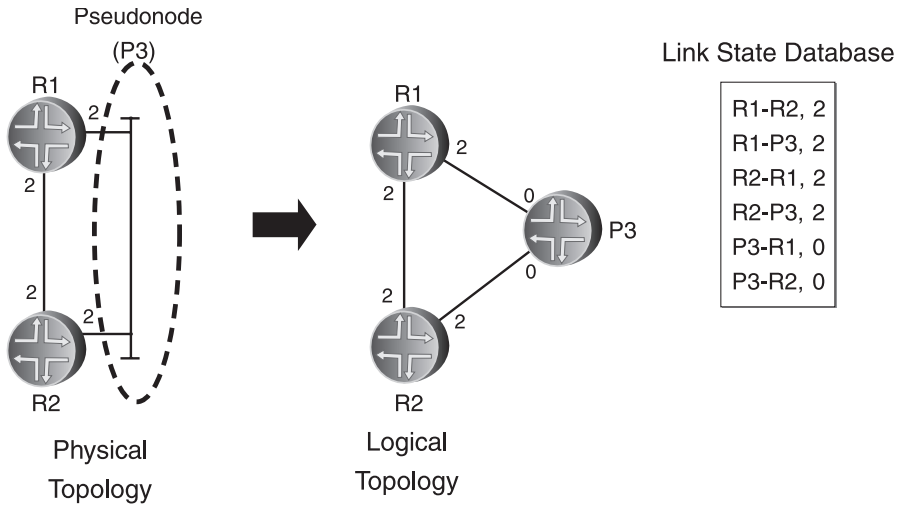

Figure 8.15 There are two equal-cost paths between R1 and R2, one of which is point-to-point and one of which is broadcast.

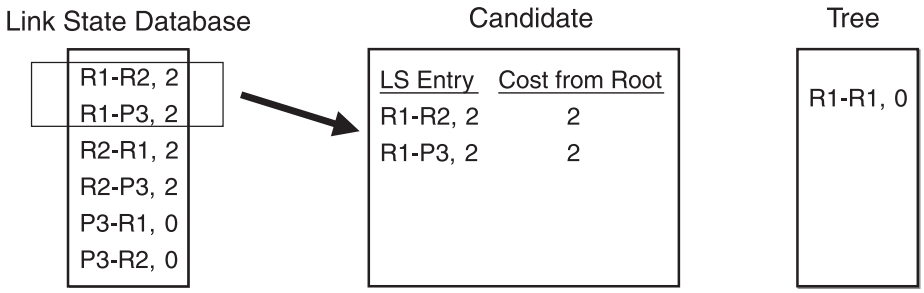

Figure 8.16 R1 installs itself as root and adds the links to its neighbors to the candidate database.

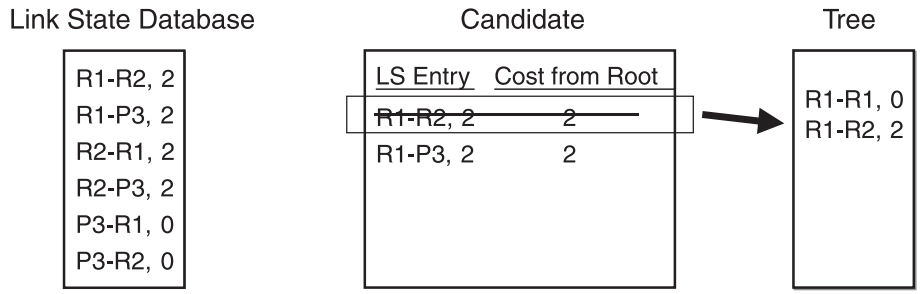

Figure 8.17 The R1-R2 and R1-P3 links are of equal cost from the root, so one is randomly chosen to move to the tree database.

Because a link to R2 was added to the tree, the links to R2's neighbors are examined in the link state database (Figure 8.18). There are two, R2-R1 and R2-P3. R1 is already in the tree database, so only the R2-P3 link is added to the candidate database.

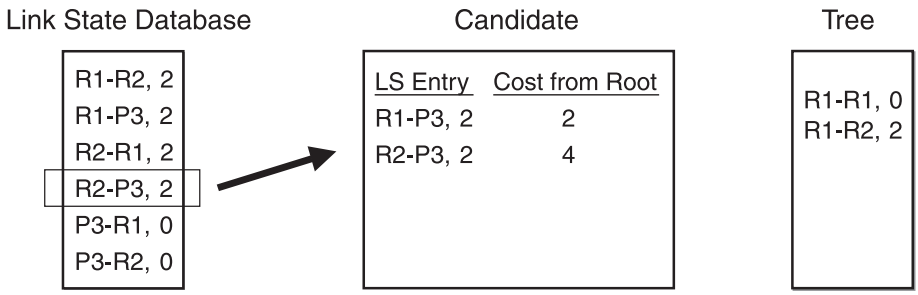

Figure 8.18 A link to R2 was added to the tree database in the last step, so the R2-P3 link is added to the candidate database.

In Figure 8.19, R1-P3 has the lowest cost from the root and so is moved to the tree database. Because there is now a path to P3 in the tree database, the higher-cost R2-P3 link is deleted from the candidate database.

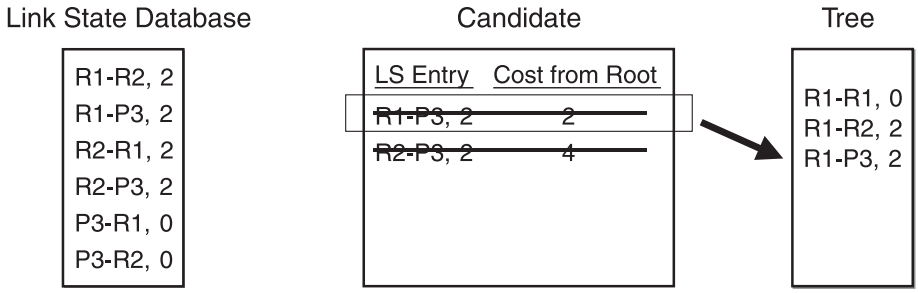

Figure 8.19 The lower-cost R1-P3 link is moved to the tree database and the higher-cost R2-P3 link is deleted from the candidate database.

With P3 added to the tree, its links to its neighbors are examined in the link state database. There are two, P3-R1 and P3-R2. But both R1 and R2 are already on the tree, so neither of these links is moved to the candidate database. At this point, the candidate database is empty, so the SPF calculation stops. And now, in Figure 8.20, we can look at the resulting tree and see the ECMP problem. R1 has branches to R2 and to P3, each at cost of 2, but the tree does not include a branch from P3 to R2. Therefore, R1 forwards all traffic to R2 across the point-to-point link and none across the Ethernet link. No load balancing can take place.

As mentioned, the problem actually arose with the random selection in Figure 8.17 of the R1-R2 link from the candidate database. So let's back up to that step and select the other link instead. Figure 8.21 shows the candidate database in the same state as in Figure 8.17, but now R1-P3 is selected and moved to the tree database instead of R1-R2.

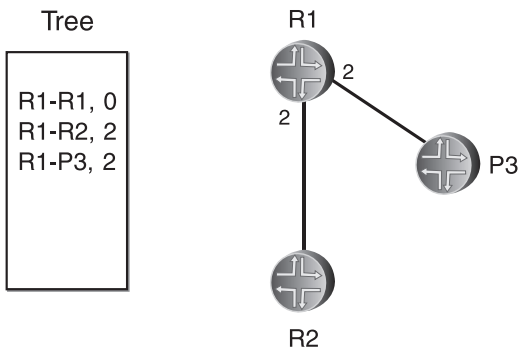

Figure 8.20 The resulting tree does not allow load balancing across the two equal-cost paths because it does not include a branch from P3 to R2.

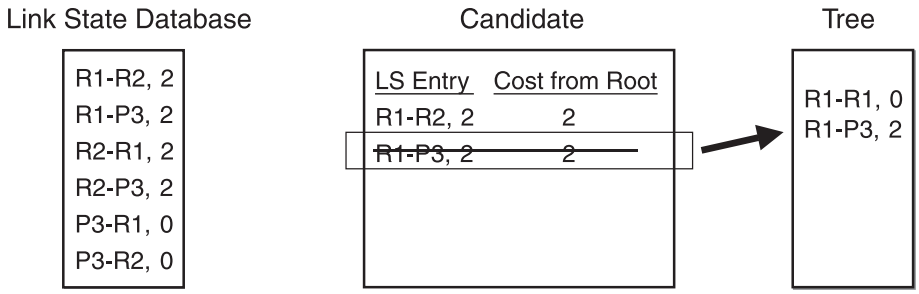

Figure 8.21 Returning to the point in the SPF calculation where either R1-R2 or R1-P3 is randomly selected for the tree database, R1-P3 is now chosen instead of R1-R2 as in Figure 8.17.

Continuing on with the SPF calculation, because a link to P3 was added to the tree, the links to P3's neighbors are examined in the link state database. R1 is already on the tree, so only P3-R2 is added to the candidate database (Figure 8.22).

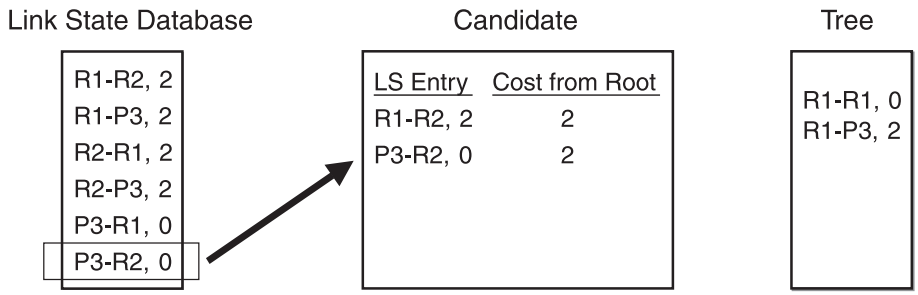

Figure 8.22 The P3-R2 link is added to the candidate database.

Now, in Figure 8.23, the candidate database contains two equal-cost entries for links to R2. Using the modification to the SPF algorithm described in Section 8.1.1 to accommodate ECMP, rather than randomly selecting one and discarding the other, both are moved to the tree database. No links remain in the link state database to nodes that are not already on the tree, and the candidate database is now empty, so SPF stops.

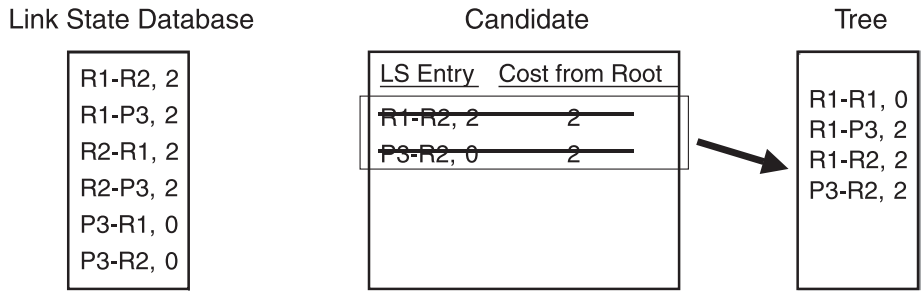

Figure 8.23 R1-R2 and P3-R2 are equal-cost links to the same node, so they are both moved to the tree database.

Figure 8.24 shows the resulting tree. Both equal-cost paths are now represented, and R1 can load balance traffic to R2. The significant step that prevented disruption of ECMP is that in Figure 8.21 the link to the pseudonode was selected for the tree database *before* the link to R2. Therefore, to ensure that ECMP works correctly when the ECMP group consists of a mixture of point-to-point and broadcast links, the following simple rule is added to the SPF procedure:

> If there are multiple entries in the candidate database with equally low cost, and if at least one link is to a pseudonode and at least one link is to a router, always select the link to the pseudonode first rather than randomly selecting among the links.

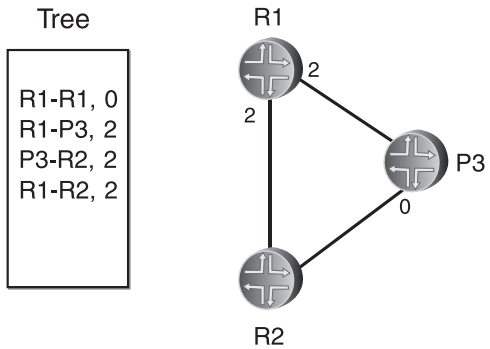

Figure 8.24 The resulting tree correctly accounts for the two equal-cost paths, and so allows load balancing.

Specific OSPF or IS-IS implementations might or might not include this modification. So if load balancing is a part of your network design, be sure to verify with your vendor that their SPF routine includes this rule.

8.1.3 Incremental SPF Calculations

It is not difficult to find network engineers and operators who think distance vector protocols are preferable over link state for all but the largest networks. Ask them why, and their answer is "because the SPF calculations are complex and CPU intensive, and they will hurt my routers' performance." These folks are stuck in the nineties. Concern over the price in router resources needed to pay for SPF was partially justified a decade or so ago, but there is no longer reason to fear SPF (in relation to distance vector protocols, at least). Several factors contribute to this change in attitude:

- Increased router performance and memory capacity
- Increased sophistication of physical and logical router architectures
- Increased base of experience designing and operating large, complex networks
- Increased base of vendor experience designing sophisticated SPF processes

One of the improvements several router vendors have incorporated into their SPF procedures is Incremental SPF (iSPF). To understand iSPF, consider the network shown in Figure 8.25. Using the link costs given in the illustration, you can determine with little trouble that R1 will calculate the shortest-path tree shown in Figure 8.26.

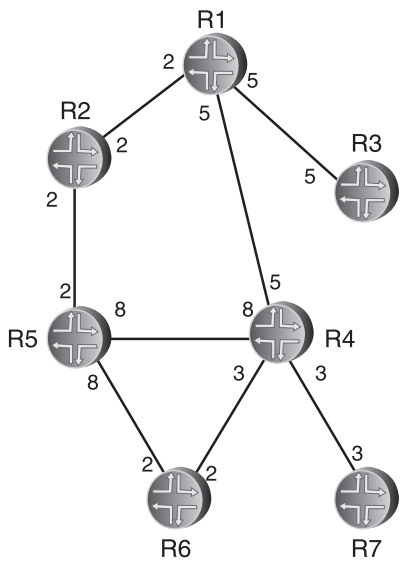

Figure 8.25 An example network and link costs.

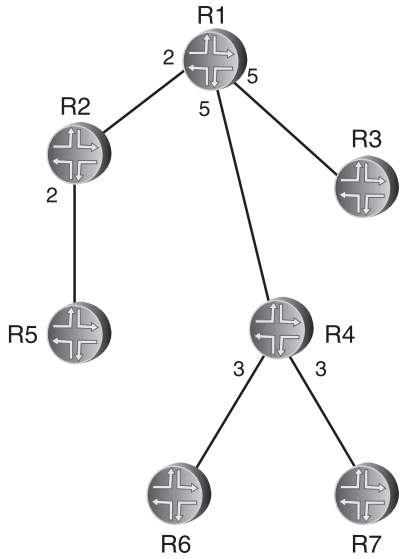

Figure 8.26 The shortest-path tree calculated by R1.

Now consider a change to the network topology. Figure 8.27 shows the addition of a new node, R8. The change to the tree to account for the new node is shown in Figure 8.28. With the SPF procedures as you have seen them so far, the addition of the new node requires the reflooding of LSAs or LSPs and a recalculation of SPF by all routers in the area. But a cursory look at the new topology shows that most of the tree has not changed; the addition of R8 involves the simple addition of a branch from R5.

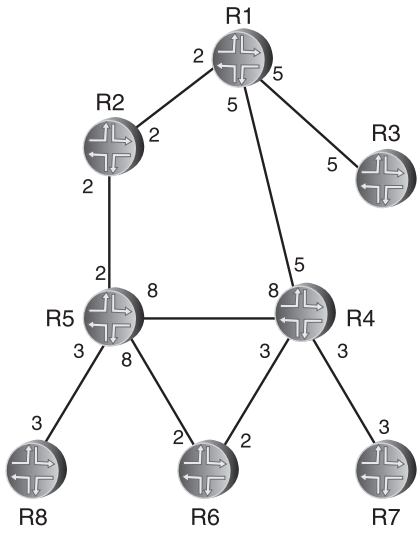

Figure 8.27 Node R8 is added to the topology of Figure 8.25.

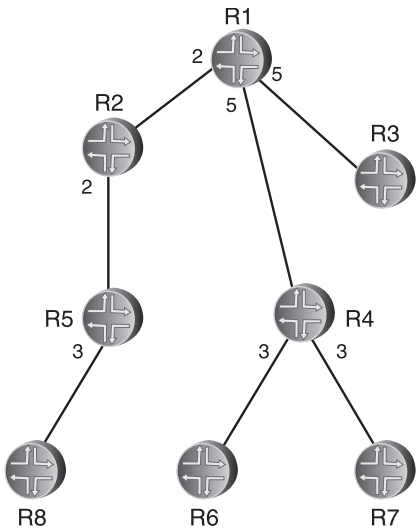

Figure 8.28 The new shortest-path tree from R1 shows the added router R8.

Consider another topological change, shown in Figure 8.29. Here, the link between R4 and R5 has failed or been disabled. Again, our understanding of SPF so far requires that this information be flooded throughout the area and that all nodes rerun SPF. But this link is not a part of the shortest-path tree shown in Figure 8.26, so rerunning SPF results in a tree that looks exactly the same as it did before the link failure.

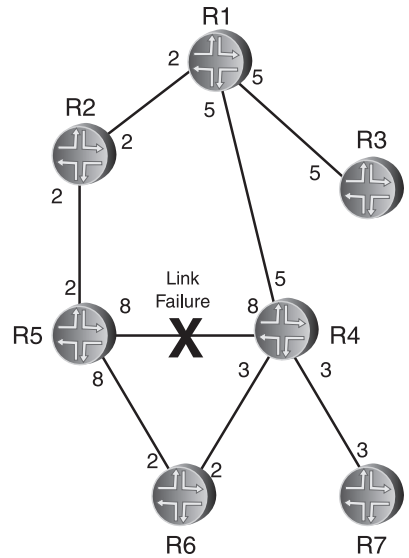

Figure 8.29 Failure of the R4-R5 link does not change the shortest-path tree, because it was not a branch of the tree to begin with.

These two topological changes are the basis for iSPF. In the first example, in which a router with a single link to the network was added (that is, a *stub router*), there are no alternative links to consider and so no need to attempt to calculate a shortest path. There is only one path to the new node. Therefore, a full SPF run is unnecessary for the routers that were already on the tree; they only need to add the new branch from R5 to R8.

In the second example, the failure of the link has no affect whatsoever on the shortest-path tree and so again there is no need for the nodes in the area to rerun SPF. Only a reduction in the metrics associated with the R4-R5 link would necessitate a full SPF run.

iSPF takes into consideration topological changes, such as the two shown in this section, and performs SPF only to the extent necessary to allow for the changes. In the first case, a simple distance vector–like addition to the tree is required; in the second case, no action at all is required. Experience has shown that iSPF provides the greatest benefit in cases of the first example, when stub routers are added to or removed from the topology.

8.1.4 Partial Route Calculations

The topology changes described in the previous section are by no means unusual in an IGP-routed topology, so it is easy to understand how the changes to the basic SPF procedure incorporated by iSPF can improve efficiency. Another network change, as common as those of the previous section, is the addition, deletion, or metric change of IP addresses. Reviewing the basic SPF procedure, you can see that the only addresses used in the computation of the shortest-path tree are the RIDs. Although recording the distance and direction of IP prefixes from each router is the ultimate goal of any routing protocol, the prefixes themselves have no bearing on an SPF calculation; after the location and cost of each node on the shortest-path tree are determined, it should be a simple matter to record what prefixes are attached to or advertised by what node.

This is the basis of *Partial Route Calculations* (PRC). When a node advertises the addition, deletion, or metric change of an IP prefix, rather than unnecessarily rerunning the SPF calculation the other nodes on the tree simply record the change. Although a full SPF run in even large networks is very fast—a high-performance router should be able to process 500 to 1000 nodes in 50 to 100ms—PRC is substantially faster, ranging from 0.5 to 10ms depending on the number of prefixes to be scanned.

PRC's increased efficiency is more substantial in IS-IS than in OSPFv2 because of the way each protocol advertises IP prefixes. IS-IS advertises all IP prefixes in IP Reachability TLVs, whereas node information necessary for SPF calculations is advertised in IS Neighbors or IS Reachability TLVs. This clear separation of prefix information from topology information makes PRC easily applicable to any IP address change.

OSPFv2, on the other hand, incorporates IP address semantics into type 1 and type 2 LSAs. That is, routers or pseudonodes advertise both topological information and their attached IP addresses in the same LSAs. So for example, if an IP address attached to an OSPFv2 router changes, the router must originate a type 1 LSA to advertise the change. However, because node information is also carried in type 1 LSAs, flooding the LSA triggers a full SPF calculation in all routers in an area—even though the address change is irrelevant

to the node topology. As a result, only type 3, 4, 5, and 7 LSAs—whose only purpose is to advertise prefix information—trigger a PRC in OSPFv2.

The applicability of PRC to all IP prefixes in an IS-IS domain, not just prefixes external to an area as in OSPFv2, is a significant contributing factor to the better scaling properties of IS-IS in a single area. You will see in Chapter 12 that OSPFv3 improves its scaling by removing addressing semantics from its Router and Network LSAs and using a new type of LSA to advertise attached prefixes.

8.1.5 SPF Delay

Although the kinds of network events that trigger iSPF and PRC are common, they are not necessarily frequent. In large networks, the kinds of address changes that trigger PRC happen only a few times a day, and the kinds of topological changes that trigger iSPF might happen only a few times a week. Moreover, a full SPF run in even a very large network takes only tens of milliseconds. Given these facts, the value of iSPF and PRC becomes evident mainly when abnormal things happen. But in a large area, LSAs or LSPs will flood regularly because of the random expiration of refresh timers around the network, requiring regular SPF runs. And when bad things happen, routers can become inundated with changing LSA/LSPs. If each LSA/LSP triggers a full or incremental SPF run, and if they are arriving fast and furious, SPF can begin eating up the majority of CPU cycles. Scheduler slips result, important tasks become delayed or completely missed, and the router can become seriously destabilized.

The challenge in large-scale networks, then, is to quickly react to network changes while at the same time not allowing SPF calculations to dominate the route processors. This is the goal of *SPF delay*, also called *SPF holddown* or *SPF throttling*. Rather than kick off an SPF calculation every time a new LSA/LSP arrives, SPF delay forces the router to wait a bit between SPF runs. If a large number of LSA/LSPs are being flooded, a delay between SPF runs means that more LSA/LSPs are added to the link state database during the holddown period. Efficiency is then increased because when the holddown period expires and SPF is run, more network changes are included in a single calculation.

SPF delay arguably adds another small efficiency. When an SPF calculation is being run, it is necessary to "freeze" the link state database so that new LSA/LSPs are not added.[3] The reason for this is obvious: Changing the link state database in the middle of an SPF calculation could corrupt the results. So, all LSA/LSPs that arrive during a running SPF calculation must be buffered until the calculation ends, and only then added to the link state database (likely triggering another SPF run). SPF delay, by reducing the overall frequency of SPF runs, reduces the frequency that LSA/LSPs must be buffered.

Of course, the price you pay for delaying an SPF calculation is an increased network convergence time. So, the challenge when determining a delay interval is to make it long enough that a heavy surge of LSA/LSPs does not destabilize the router while still keeping it short enough that convergence is not hurt significantly. Even better would be to have normal "fast" SPF calculations when conditions in the network are normal, and delayed SPF calculations when the network is unstable. Several vendors do offer such adaptive SPF timers.

[3] A more elegant approach than freezing the database is to lock just the LSPs that are being processed.

Juniper Networks uses a delay scheme in which the normal period between SPF runs is short. This period is 200ms by default, meaning an SPF run cannot take place any sooner than 200ms after the last run. The period is configurable with the **spf-delay** command to between 50 and 1000ms; the command can be used with both OSPF and IS-IS, and applies to both full and partial SPF calculations. If three SPF runs are triggered in quick succession, indicating instability in the network, the delay period is automatically changed to 5 seconds. This "slow mode" period is not configurable. The routers remain in this "slow mode" until 20 seconds have passed since the last SPF run—indicating that the network has stabilized—and then switches back to "fast mode."

Cisco Systems originally had a similar linear fast/slow algorithm (configured for OSPF with the command **timers spf**). But more recently, Cisco has taken a different approach to adaptive SPF delays by using an *exponential backoff* algorithm. *Initial delay*, *delay increment*, and *maximum delay* periods[4] are configured. The router waits the initial delay period before first running SPF. After the first run, the delay is increased by doubling the delay increment every time SPF runs. So for example, if the initial delay is 100ms and the delay increment is 1000ms, the router delays the first SPF run by 100ms, the second by 1000ms, the third by 2000ms, the fourth by 4000ms, and so on. The maximum delay value specifies in seconds the largest value to which the delay can be incremented—an obvious necessity to prevent an unstable network from causing the SPF delay to increase so much that SPF does not run at all. When SPF has not run for twice the time specified by the maximum delay period, the router switches back to "fast" mode in which the initial delay period is used.

These three timers are specified for OSPF with the IOS command **timers throttle spf** and for full IS-IS SPF with the command **spf-interval**. Partial SPF for IS-IS is configured separately with the IOS command **prc-interval.**

8.2 Flooding Enhancements

Section 8.1.5 refers several times to network instabilities causing a flood of LSA/LSPs, but the focus is on enhancing SPF so that the router's processor is not overwhelmed during unusually busy periods. In other words, the focus is on the router protecting itself. But as in any community, self-protection becomes less of an issue if all the members of the community behave themselves and watch out for their neighbors. In the case of link state protocols, enhancing the flooding mechanism to reduce the chance that a router will overwhelm a neighbor or an area with LSA/LSPs makes a router a better neighbor.

As with SPF enhancements, most flooding enhancements are not a part of the open protocol specifications. (IS-IS mesh groups are the exception.) They have been developed by vendors in an effort to make their OSPF and IS-IS implementations scalable to large networks. What enhancements are developed by a given vendor likely depend on the specific kinds of networks the vendor's routers must accommodate and the features requested by the vendor's customers.

[4] The actual IOS command terms for these three values varies depending on whether you are configuring exponential backoff for OSPF or IS-IS, but their function is the same.

8.2.1 Transmit Pacing

Increasing the interval between subsequent LSA/LSP transmissions is variously called delay, pacing, or throttling. Whatever you want to call it, delaying the transmission of LSA/LSPs prevents a router from dominating a link or overwhelming a neighbor. There are two aspects to delaying LSA/LSP transmission: delaying self-originated LSA/LSPs, and delaying LSA/LSPs forwarded through the router during the flooding process.

To understand how delaying can help make flooding more efficient, consider first a naïve implementation that refreshes at some set interval (every 30 minutes for OSPF, every 20 minutes for IS-IS). Whenever the refresh timer expires, the link state database is scanned and all of its self-originated LSAs or LSPs are flooded. This single-timer refresh interval results in a periodic, heavy flooding interspersed by equal periods of complete quiet, as illustrated in Figure 8.30(a). Implementing individual refresh timers for each LSA/LSP spreads the refresh load out randomly, as shown in Figure 8.30(b), so that neighbors are not hit with a large number of LSA/LSPs all at once.

However, individualizing the refresh timers is not as efficient as it could be, especially for OSPF where Update packets might carry a single or a very few LSAs. If, instead of immediately transmitting an LSA when its refresh timer expires, the transmission is delayed for some period, additional refresh timers are more likely to expire during the waiting period and more LSAs can be carried in a single Update. The "grouping" effect of this delay is shown in Figure 8.30(c).

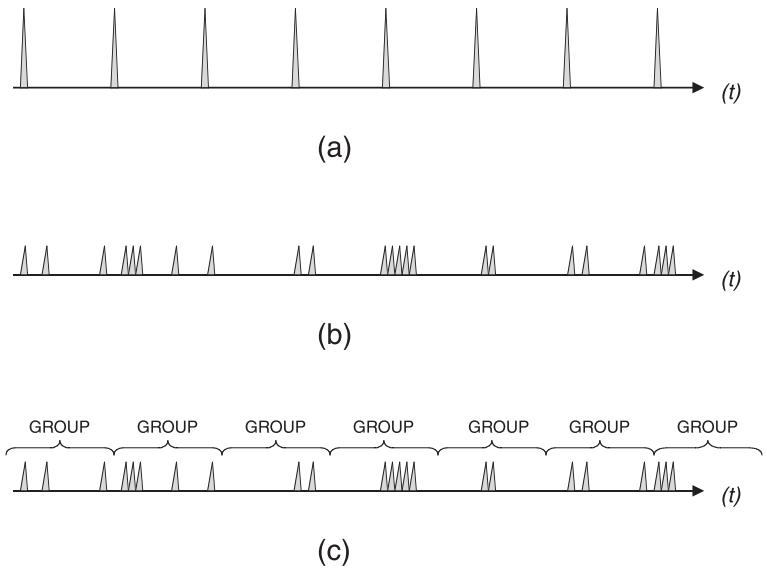

Figure 8.30 A single refresh timer for the entire link state database results in possibly heavy floods every refresh interval (a), whereas separate refresh timers for each database entity results in smaller, more random floods (b). Adding a delay to the individual refresh timers improves efficiency by grouping several entities into one flood (c).

Cisco's IOS uses a default LSA pacing timer of 4 minutes; the timer can be changed to between 10 seconds and 1800 seconds (30 minutes) with the command **timers pacing lsa-group**[5] This pacing applies not only to LSA refreshing but also to aging and checksumming. If the link state database is very large, on the order of thousands of LSAs, the default pacing delay might cause spikes in the flooding similar to those in Figure 8.30(a); reducing the pacing delay can smooth out the flooding pattern.

Delaying the transmission of self-originated LSA/LSPs has its greatest benefit when there is instability local to the router—most usually the flapping of a connected link. Rather than flooding a new LSA/LSP every time the link changes state, a delay period might span several state changes, thereby dampening the impact the flapping has on flooding activity. Both OSPF and IS-IS standards do specify delays that help in this regard. OSPF specifies two architectural constants: New instances of a given LSA cannot be generated more frequently than 5 seconds (MinLSInterval), and new instances of a given LSA cannot be received more frequently than 1 second (MinLSArrival). IS-IS specifies similar delays, although ISO 10589 suggests values for the delays rather than making them constants: 30 seconds between the generation of new LSPs (minimumLSPGenerationInterval), and 5 seconds between transmissions of LSPs from the same originator (minimumLSPTransmissionInterval). Vendor implementations often enhance these basic delays to scale to different network sizes.

Cisco's IOS uses the same exponential backoff mechanism described in Section 8.1.5 to throttle the transmission of self-originated LSPs: Using the command **lsp-gen-interval**, you can specify initial delay, delay increment, and maximum delay periods. As described, the initial delay specifies the time to wait, in milliseconds, after the first generation of a new LSP before transmitting it. The delay increment is the multiplier used to exponentially increase the delay between subsequent transmissions, in milliseconds. The interval between the first and second transmission is the delay increment value, and then the interval between subsequent transmissions is twice the interval of the previous transmission: IncrementValue, 2*IncrementValue, 4*IncrementValue, 8*IncrementValue, and so on, until the maximum delay value, specified in seconds, is reached. The delay between subsequent transmissions then remains at the maximum delay. If no new LSPs are generated for twice the maximum delay value, at that point the exponential backoff mechanism is reset.

Juniper's JUNOS does not use an exponential backoff, nor does it provide configurable options for the LSP delay. Instead, it uses a "fast mode" and "slow mode" scheme similar to that described for its SPF delay in Section 8.1.5. The normal "fast mode" transmission delay of self-originated LSPs is 20ms. If three LSPs are generated in quick succession, IS-IS switches to "slow mode" and delays each transmission by 10 seconds until the network stabilizes.

The other aspect of transmission pacing is the control of flooding of LSA/LSPs originated by other routers. In times of instability, hundreds or thousands of LSA/LSPs can be flooded within an area; a router must be able to pace the transmission of OSPF Updates or IS-IS LSPs to limit the rate its neighbors receive these messages.

IOS uses the command **timers pacing flood** to configure the minimum interval, in milliseconds, between transmitted OSPF Update packets. The default interval is 33ms and can

[5] Older versions of IOS use the command **timers lsa-group-pacing**.

be changed in the range of 5 to 100ms. JUNOS uses a hard-coded delay interval that cannot be changed.

For IS-IS, IOS uses the command **isis lsp-interval** to specify the pacing of LSP transmissions. The default is again 33ms. JUNOS uses the very similar command, **lsp-interval**, to change its default interval of 100ms.

Simple arithmetic shows that an interval of 100ms, for example, means that Updates or LSPs cannot be transmitted any faster than one every .1 seconds, or a maximum transmission rate of 10 packets per second; 50ms means a maximum rate of 20 packets per second, and so on.

In all cases, these commands are configured per interface so that the change from the default is applied to specific neighbors. It must be noted that in the great majority of cases, the default (or hard-coded) transmission interval is sufficient. There are better ways to protect a low-powered neighbor from the impact of large-scale flooding, such as good OSPF area design and well-designed packet queues.

8.2.2 Retransmit Pacing

Yet another aspect of controlling flooding is the pacing of LSA/LSP retransmits. Recall from the discussion in Chapter 5 that flooding must be reliable, and so LSA/LSPs that are not acknowledged either implicitly or explicitly within a specified time are retransmitted. OSPF does this by placing a copy of a transmitted LSA on a Retransmit List and setting a retransmit timer (normally 5 seconds). If the LSA is acknowledged, it is removed from the Retransmit List. If the retransmit timer expires, a copy of the LSA is retransmitted and the retransmit timer is restarted.

The IS-IS retransmission process for point-to-point and broadcast links is different. On point-to-point links, the Send Routing Message (SRM) flag of a transmitted LSP is not cleared until the LSP is explicitly acknowledged with a PSNP. So if the SRM for that link is still set the next time the LS database is scanned (which happens every 5 seconds or minimumLSPTransmissionInterval), the LSP is retransmitted. On broadcast links, transmitted LSPs are always implicitly acknowledged by CSNPs, transmitted by the DIS every 10 seconds. If a router does not see the instance of an LSP that it transmitted in the next received CSNP, it retransmits the LSP.

The problem here is that if flooding is heavy, a low-powered router might be so busy processing received LSA/LSPs that it does not acknowledge their receipt promptly, causing its neighbors to retransmit. If the router is already busy, the retransmissions can just make matters worse. You can use the IOS command **ip ospf retransmit-interval** or the JUNOS command **retransmit-interval** to change the default OSPF retransmission interval from 5 seconds to an interval in the range of 1 to 65,535 seconds; both commands are applied per interface.

IOS can also change the default IS-IS retransmission interval of 5 seconds on point-to-point links using the command **isis retransmit-interval**. The interval you can set with this command ranges from 0 to 65,535 seconds. Although this command changes the interval between scans of the database for any SRM flags set for the interface, there is another

command—**isis retransmit-throttle-interval**—that actually controls the rate, in milliseconds, at which retransmitted LSPs are sent. JUNOS does not allow the default IS-IS retransmission interval to be changed.

8.2.3 Mesh Groups

Flooding load is a particular problem in heavily meshed networks—such as those built on ATM or Frame Relay infrastructures. Recall from the basics of flooding that when a router receives a flooded LSA/LSP, it forwards it to all neighbors except the one from which it received the data unit. This is simple split-horizon forwarding. With heavily meshed networks, however, each router has many paths to other routers; in a fully meshed network such as the one in Figure 8.31, each router has a connection to *every* other router. This meshing means that there are numerous ways for an LSA/LSP to be replicated so that one router is likely to receive many copies of the same LSA/LSP. In Figure 8.32, for example, a router originates and floods an LSA or LSP. The direct connections mean that the information is communicated to all other routers with this initial flood. But the other routers have no way of knowing that all of their neighbors have received the information, so they flood the LSA/LSPs to all neighbors except the one they received the information from as shown in Figure 8.33. This second phase of flooding is entirely unnecessary.

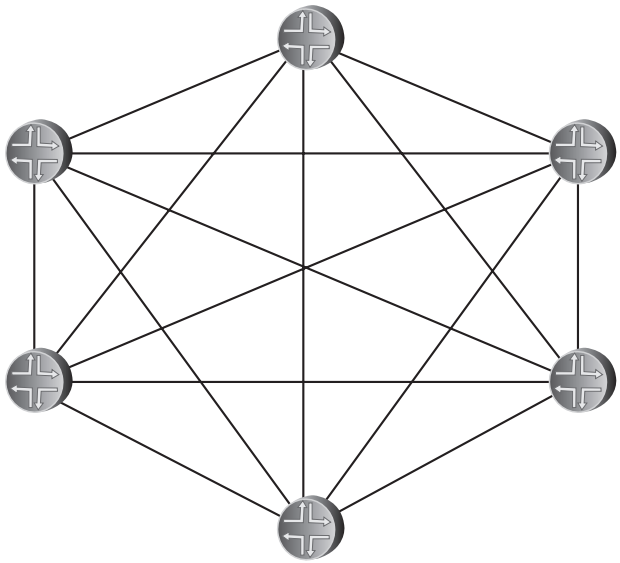

Figure 8.31 In a fully meshed network, every router has a connection to every other router.

In a fully meshed network, every router except the originator will flood $(n - 2)$ unnecessary LSA/LSPs, where n is the number of routers in the network. This works out to $(n - 1)(n - 2)$ or $(n^2 - 3n + 2)$ unnecessary LSA/LSPs. The example network shown here

is small enough that the extra flooding load—20 extra LSA/LSPs—does not significantly impact network resources. As the network grows larger, however, the waste of resources also becomes larger. One flooded LSA/LSP in a fully meshed network of 50 routers, for example, results in 2352 unnecessary replications; in a network of 100 routers, 9702 unnecessary LSA/LSPs are flooded.

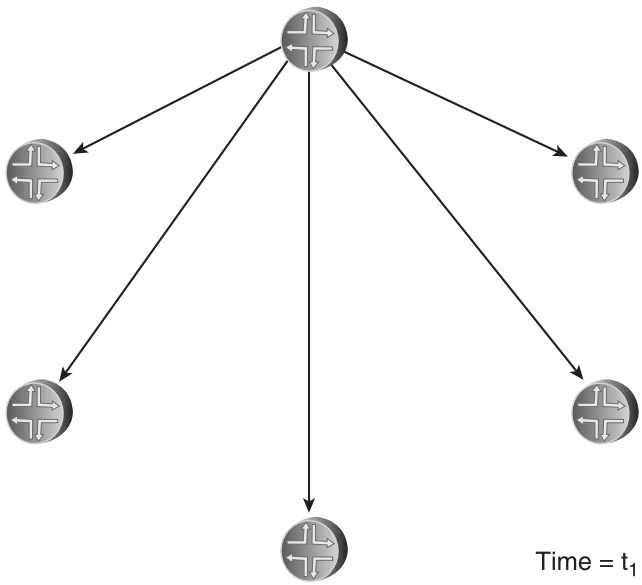

Figure 8.32 When a router floods an LSA or LSP in a fully meshed network, the information is immediately received by all other routers.

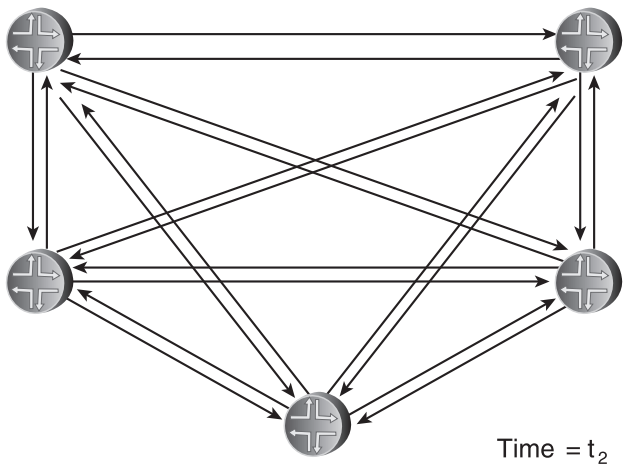

Figure 8.33 Because the other routers in the meshed network have no way to know that their neighbors have received the flooded information, they unnecessarily flood to their neighbors.

IS-IS provides a technique for limiting the scope of unnecessary flooding called *mesh groups*.[6] Mesh groups apply to point-to-point interfaces, and when mesh groups is enabled an interface can be in one of three modes:

- Inactive
- Blocked
- Set

Inactive mode means the mesh group is inactive for that interface, and LSPs are flooded normally. In blocked mode, no LSPs are flooded out that interface. Figure 8.34 shows how blocked mode might be applied to the network in Figure 8.31. Here, all links depicted with a dashed line are in blocked mode and do not flood LSPs. Each router has two unblocked links, so flooding can still take place if any one link fails. However, some redundancy is exchanged for scalability. If both of the unblocked links to one of the routers fail, the router cannot flood LSPs even though it still has three perfectly good—but blocked—links to the rest of the network.

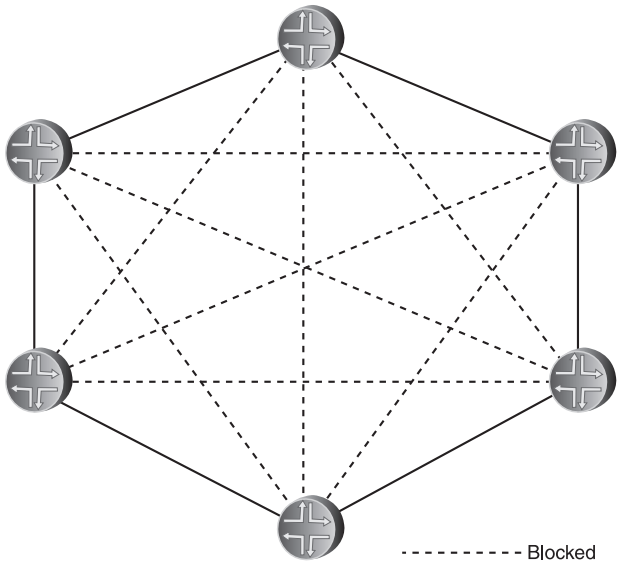

Figure 8.34 Interfaces in blocked mode do not flood any LSPs.

[6] Rajesh Balay, Dave Katz, and Jeff Parker, "IS-IS Mesh Groups," RFC 2973, October 2000.

Some convergence time might also be sacrificed. In Figure 8.35, for example, an LSP flooded from one router must pass through a few other routers before the LSP reaches all routers, even though the originator has direct links to every router.

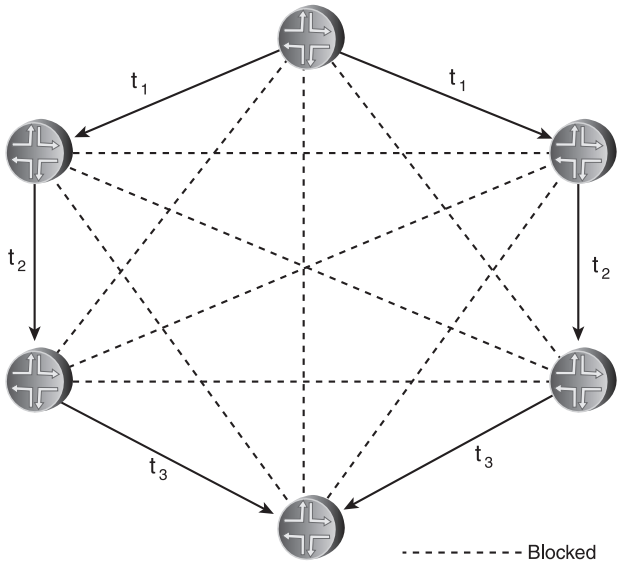

Figure 8.35 Blocking LSP flooding on some interfaces can slightly increase overall network convergence time.

Set mode offers a compromise between the sharply reduced flooding load but reduced redundancy and increased convergence time of blocked mode. Rather than grouping all interfaces into either blocked or unblocked, set mode groups interfaces into numbered groups. For example, in Figure 8.36 all interfaces belong to either group 1 or group 2. The rule for set mode is then very simple: A received LSP is not flooded out any interface belonging to the same group as the interface on which it was received.

Suppose a router in the network of Figure 8.36 originates an LSP, as shown in Figure 8.37. As the originator, it floods the LSP to all neighbors. Comparing this illustration with the numbered mesh groups in Figure 8.36, you can see that some neighbors receive the LSP on an interface belonging to mesh group 1 and some neighbors receive the LSP on interfaces belonging to mesh group 2.

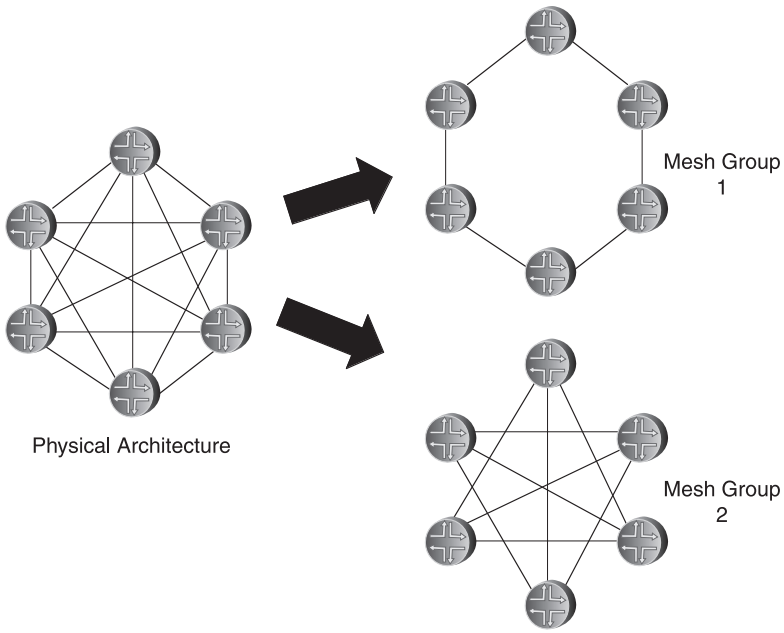

Figure 8.36 Set mode assigns interfaces to a numbered group.

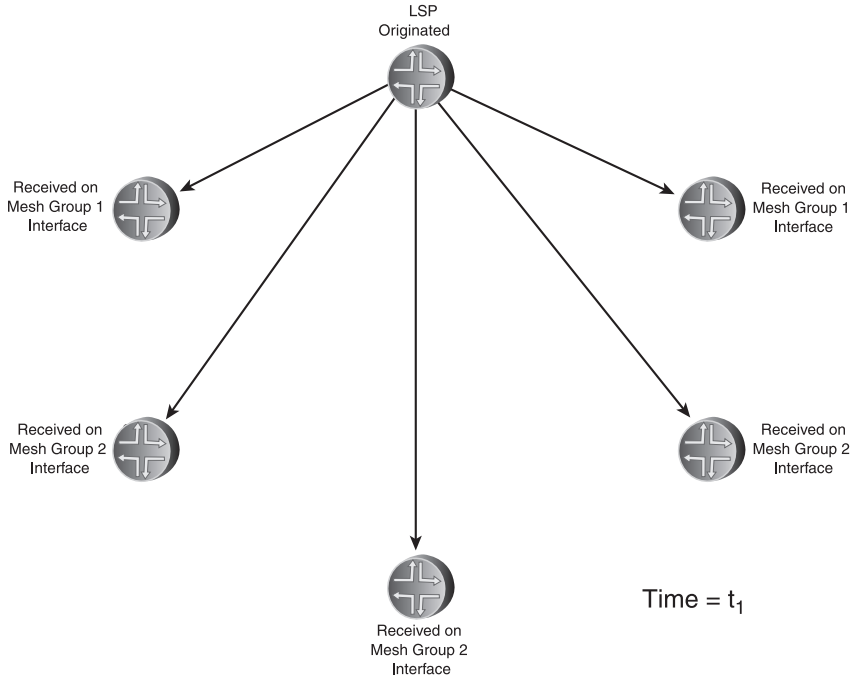

Figure 8.37 When an LSP is initially flooded, it is received by some neighbors on group 1 interfaces and by some neighbors on group 2 interfaces.

In Figure 8.38, the neighbors flood the LSP. Comparing the illustration to the groups in Figure 8.36, you can see that if the LSP was received on a group 1 interface, it is not flooded on any group 1 interface; and if the LSP was received on a group 2 interface, it is not flooded on any group 2 interface. You can also see that although there is less unnecessary flooding than in the fully meshed network in Figure 8.33, there is more than with the blocked mode in Figure 8.35. However, the reduced convergence seen with the blocked mode architecture is eliminated in this set mode architecture.

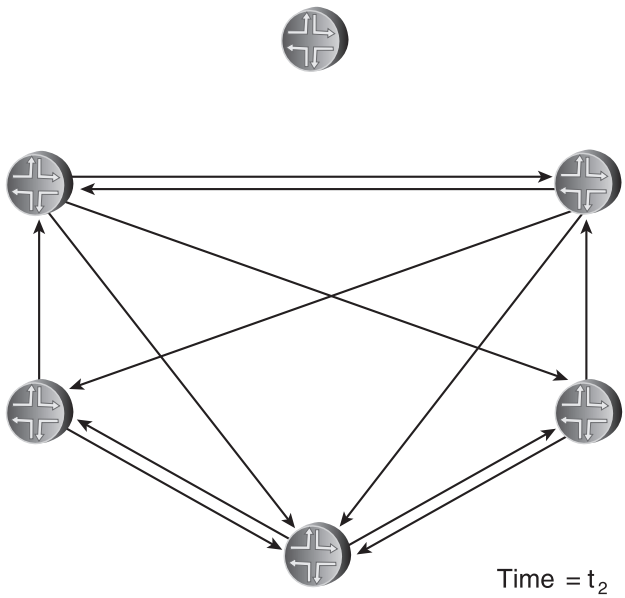

Figure 8.38 Neighbors flood the received LSP only on interfaces belonging to groups other than the receiving interface's group.

In more complex topologies than shown here, you can use a combination of inactive, blocked, and set mode interfaces to manage the flooding patterns in the network. However, any time you use mesh groups, you trade some redundancy or convergence time or both for improved scalability, so you should consider carefully whether mesh groups are right for your network and, if so, design them carefully for the best balance between reduced flooding and reduced reliability.

OSPF does not have a comparable feature to IS-IS mesh groups. But some implementations do provide an LSA filtering function that enables you to create an effect similar to mesh groups. The Cisco IOS **database-filter** command, for instance, can be applied to block the flooding of LSAs on a given interface. However, OSPF networks typically consist of multiple areas that help scale flooding, whereas IS-IS is often used in very large, single-area networks where flooding load is more of a problem. Therefore, mesh groups are not as important for OSPF as they can be for IS-IS.

8.2.4 Demand Circuits and Flood Reduction

Although IS-IS is usually found only in relatively large IP networks, OSPF is found in networks of all sizes. And in small networks, you are more likely to find *demand circuits*—links that should be used only when there is a demand for them, and should otherwise be silent. The most common modern examples of demand circuits are links that treat data exchanges as a "call," such as dial-up and low-bandwidth ISDN. You do not want such connections to stay up permanently. Demand circuits also include any circuit for which you are billed based on the amount of packets traversing it.

Running OSPF over demand circuits is problematic because the Hellos will either keep the circuit up permanently or cause the circuit to connect and disconnect every 10 seconds just to transport the Hello packets. Additionally, periodically refreshing LSAs across a demand circuit when nothing has changed can cause unwanted connections or billing. An extension to OSPF makes the following modifications to accommodate demand circuits:[7]

- Hellos are sent only to bring up the circuit for the initial database synchronization of the neighbors on each side of the link. After synchronization, no Hellos are sent.
- LSAs are flooded across the demand circuit during synchronization, but are not periodically refreshed; LSAs are sent across the link only if there is a change in the LSA warranting a new instance.

If an LSA is not going to be periodically refreshed across a demand circuit, it must not "age out" of the link state databases in which it resides. That is, the age should not reach MaxAge. To accomplish this, the highest-order bit in the 16-bit Age field of the LSA header is designated as the *DoNotAge* bit. When this bit is set in an LSA, the age is incremented as usual during flooding, but is not incremented after the LSA has been installed in the link state database.

Of course, for this scheme to work all routers must understand and support the DoNotAge bit. If one router in the area does not, and increments the age to MaxAge, it will delete the LSA and the databases in the area will no longer be identical. Therefore, for OSPF over demand circuits to be reliable, all routers in an area must indicate their support for the extension by setting the Demand Circuit (DC) bit in the Options field of all LSAs it originates (Figure 8.39). If any LSA appears in any link state database in the area with the DC bit cleared, the router flushes all DoNotAge LSAs from its database.[8] The originators of these LSAs must then flood new instances, with the DoNotAge bit cleared. The DC bit is also set in the Options field of Hello and Database Description packets sent across demand circuits during synchronization, to negotiate an agreement to stop sending Hellos after the neighbors are synchronized.

[7] John Moy, "Extending OSPF to Support Demand Circuits," RFC 1793, April 1995.

[8] Note that this is an exception to the rule that no router can flush an LSA from its database that it did not originate.

Figure 8.39 The DC bit in the Options field indicates support for DoNotAge LSAs.

Because all LSAs in an area in which OSPF is running over a demand circuit must have their DC bits set, it is best to put demand circuits in stub, totally stubby, or NSSA areas. Doing so eliminates the necessity of an ABR or ASBR having to set the DC bits in all type 3, 4, and 5 LSAs.

Obviously, if LSAs are not being refreshed periodically, some of the robustness inherent to OSPF is lost. This should be a factor when considering whether to run OSPF over a demand circuit.

Another consideration is that with no Hellos exchanged across a demand circuit, there is no keepalive function. If a router on one end of the circuit becomes unreachable, the neighbor on the other end will not detect it. A solution to the detection of a failed neighbor, called neighbor probing, is proposed in RFC 3883.[9] With neighbor probing, any time the link is connected for the transmission of application packets OSPF can send Updates and look for Acknowledgments. However, probing only takes place when the link is up for packet transmission; the link is not brought up just for probing.

There is also a potential situation in which the neighbor is available but the link is not. Again, the lack of Hellos means this condition cannot be detected. So, there must be a *presumption of reachability*, meaning that the circuit is presumed to be available when needed. If for some reason a connection cannot be established, OSPF does not report the link as down. Instead, the link is considered oversubscribed and packets destined to transit the link are dropped.

Yet another consideration has to do with network management software. The routers at each end of a demand circuit still refresh their LSAs out all other interfaces; periodic refreshes are suppressed only across the demand circuit. This means that the sequence number of the same LSA might not match in databases on each side of the circuit, which can lead some network management applications to falsely conclude that the databases in an area are not synchronized.

All in all, running OSPF over demand circuits in a modern network is probably a bad idea. The extension was developed in the mid-1990s, when such links were more common than they are today. But when a dial-up or low-bandwidth ISDN link is used in current networks, presumably it connects a stub router to the network rather than serving as a transit link in the middle of an area. Therefore, a better and simpler solution is likely to be static routes at each end of the link.

Although OSPF over demand circuits might not be a good idea, the demand circuit extensions can be exploited for limiting overall flooding.[10] A router performing this *flood*

[9] Sira Panduranga Rao, Alex Zinin, and Abhay Roy, "Detecting Inactive Neighbors over OSPF Demand Circuits (DC)," RFC 3883, October 2004.

[10] Padma Pillay-Esnault, "OSPF Refresh and Flooding Reduction in Stable Technologies," draft-pillay-esnault-ospf-flooding-07.txt, June 2003.

reduction continues to send Hellos to its neighbors but sets the DoNotAge bit in its LSAs as they are flooded so that they are not aged in other databases. The Cisco IOS command **ip ospf flood-reduction** is an example of a command enabling flood reduction. As with the demand circuit extensions, existing LSAs are then reflooded only when a change occurs warranting a new instance. Specifically, a new instance of an LSA is flooded only if

- The LSA's Options field changes.
- A new instance of an LSA is received which has an age of MaxAge or DoNotAge+MaxAge.
- The Length field in the LSA header changes.
- The contents of the LSA have changed, excluding the 20-octet header (because the sequence number and checksum are expected to change and do not indicate a topology change).

As with OSPF over demand circuits, the price you pay for this OSPF flood reduction is diminished robustness of the link state database maintenance. Therefore, you should use this option only in topologies that are normally stable and reliable.

8.3 Fragmentation

Sections 7.3.2 and 7.4.2 discuss the issue of large LSAs and LSPs and the effect they have on the scalability of an area in terms of bandwidth usage during flooding and memory usage as they are stored in the link state databases. Another issue with the scaling of LSAs and LSPs concerns the ability of link MTUs to accommodate them. That issue is the question of how to handle an LSA or LSP that is larger than the MTU of a link it is supposed to traverse. When there is a low-MTU link along the flooding path, an OSPF or IS-IS implementation can do one of three things:

- It can limit the information units it generates to a very small size to ensure that they never exceed the lowest possible MTU of any link. This is not a very practical approach.
- It can perform path MTU discovery and adjust the transmitted unit sizes accordingly. This adds a layer of complexity to the flooding mechanism.
- It can use fragmentation as necessary.

Fragmentation is a common and well-understood part of IP networks, so using fragmentation makes sense in situations where an OSPF or IS-IS protocol data unit is larger than the MTU of a link it must traverse.

Fragmentation is less of an issue for OSPF than for IS-IS for two reasons. First, because a single originator can generate many different LSAs of several different types to convey information, no one LSA is likely to grow extremely large. Types 3, 4, 5, and 7 LSAs carry only a single prefix, so they are always small. Type 2 LSAs might grow large, but it is rare to find a pseudonode with a great number of neighbors. Of all the LSA types, only type 1 is likely to grow large and even then only if the router has a large number of neighbors or stub links (an access router is a good example of a router with a very large number of stub links). The maximum size of an LSA is 64KB. Given 24 bytes of fixed fields and 12 bytes to represent each link, a type 1 LSA can advertise a maximum of 5331 links. In any reasonable network design, this LSA capacity should more than suffice.

The second reason that fragmentation is less of an issue for OSPF is that the LSAs, whatever their size, are encapsulated in Update messages for transmission to neighbors. The Update message is in turn encapsulated in an IP packet, and IP packet headers are formatted to accommodate fragmentation. In other words, the standard IP fragmentation procedures serve OSPF just fine.

IS-IS is a little more problematic: Unlike OSPF's many LSAs, IS-IS generates one LSP per level, per router. As a result, this one LSP can become quite large. And because the LSP is not an IP packet, it cannot take advantage of IP's fragmentation mechanism. Therefore, IS-IS must have its own fragmentation mechanism.

Prerequisite to IS-IS fragmentation, IS-IS must be able to assume that every link on which it floods LSPs can handle PDUs up to at least a certain size. That is, there must be a minimum guaranteed MTU such that IS-IS knows it can send PDUs up to that size without them exceeding the link MTU. Recall from the discussion of IS-IS Hello protocol basics in Section 4.2.2 that when Hellos are initially exchanged between IS-IS neighbors the Padding TLV is used to pad the Hello up to the ReceiveLSPBufferSize of 1492 bytes.[11] If a neighboring interface has an MTU lower than 1492 bytes, it cannot receive the padded Hellos and drops them, so that an adjacency is not formed. Therefore, if an adjacency exists with a neighbor, an IS-IS router knows that so long as its PDUs do not exceed 1492 bytes they will not exceed the MTU of any of its links.

If the number of TLVs a router originates would result in an LSP larger than 1492 bytes IS-IS breaks the LSP into fragments, none of which are larger than 1492 bytes (including the header). The 8-bit LSP Number field in the LSP header (Figure 8.40) is used to track these "LSP fragments." The first LSP, whether it is fragmented or not, has an LSP number of 0x00. Subsequent fragments are numbered 0x01, 0x02, and so on. Figure 8.41 shows a database display with 17 fragments of the same LSP.

[11] The 1492-byte limitation in the standards is to accommodate SNAP encapsulation. But in practice, vendors use LLC encapsulation, which increases the maximum LSP size to 1497 bytes. Therefore, you will commonly find IS-IS Hellos padded to 1497 bytes.

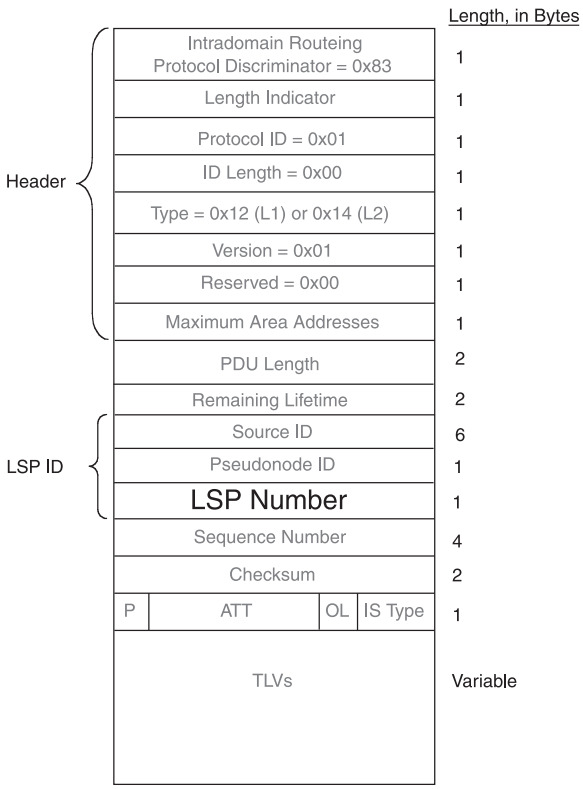

Figure 8.40 The LSP Number in the LSP header is used for tracking a fragmented LSP.

```
jeff@Juniper2> show isis database RTR1-SFO
IS-IS level 1 link-state database:
LSP ID                 Sequence Checksum Lifetime Attributes
RTR1-SFO.00-00          0x4298   0x8144    47193 L1 L2 Attached
RTR1-SFO.00-01          0x2c84   0x73c2    41078 L1 L2
RTR1-SFO.00-02          0x2919   0x72f4    41078 L1 L2
RTR1-SFO.00-03          0x21b2   0x1974    65420 L1 L2
RTR1-SFO.00-04          0x2213   0xdbc2    46671 L1 L2
RTR1-SFO.00-05          0x1e07   0x5036    65429 L1 L2
RTR1-SFO.00-06          0x1b63   0xe8e2    41078 L1 L2
RTR1-SFO.00-07          0x1624   0x7676    41078 L1 L2
RTR1-SFO.00-08          0x1598   0x4b2d    41078 L1 L2
RTR1-SFO.00-09          0x18c6   0xc7d7    41078 L1 L2
RTR1-SFO.00-0a          0x19a4   0x595d    65429 L1 L2
RTR1-SFO.00-0b          0x246b   0x3f98    65429 L1 L2
RTR1-SFO.00-0c           0xfe6   0x1a34    65429 L1 L2
RTR1-SFO.00-0d          0x1369   0x35e5    51739 L1 L2
RTR1-SFO.00-0e           0x81e    0xb82    41078 L1 L2
RTR1-SFO.00-0f          0x1211   0x8e7c    57266 L1 L2
RTR1-SFO.00-10          0x165a   0xca63    49398 L1 L2
  17 LSPs
```

Figure 8.41 The last octet of the LSP ID is the LSP Number, used for relating fragments of the same LSP.

A note about semantics is in order here. If you deal mostly with IETF terminology, using the term *fragments* when discussing a multipart LSP such as the one shown in Figure 8.41 might make the most sense to you. However, you might also say that the router RTR1-SFO has originated 17 LSPs. That is, instead of saying a router fragments its LSPs when they are large, you can say that a router produces multiple LSPs when necessary, and that the LSP number is used to differentiate the multiple LSPs from the same source. In fact, the language of ISO 10589 is oriented to multiple LSPs rather than fragments. Looking at the entries in Figure 8.41, you can see that each of the fragments have their own sequence numbers, checksums, and remaining lifetimes. And they are stored in the database as separate LSPs. But although they are flooded and stored separately, during the SPF calculations all the fragments from a single originating router are considered as a single LSP. So, you can consider these entities as fragments of the same LSP or as multiple LSPs originated from the same router. Either term is acceptable; use the one that makes more sense to you.

Also note that IS-IS does not do any sort of summation or other checking of the LSP numbers to ensure that all fragments are in the database; if there is a gap in the LSP number sequence, the SPF calculation still takes place. The one exception to this is LSP number 0. If that first fragment is missing, the other fragments are discarded. The reason for this is that only LSP number 0 contains information that is vital to the correct inclusion of the originating node in the SPF calculation, such as the ATT bit and the Overload bit (discussed in the next section).

Yet another scaling issue with IS-IS fragmentation has to do with the size of the LSP Number field. Because it is 8 bits, a single originator can generate a maximum of 256 fragments. Considering that every LSP can carry up to 1470 bytes of TLV space (after the 8 byte IS-IS header and a 19 byte LSP header, as discussed in Section 7.4.2), 256 fragments constitutes 256 * 1470 = 376,320 bytes of payload space for TLVs. If every TLV is 12 bytes long (a conservative assumption—many TLVs are much shorter), 256 LSP fragments can carry over 31,000 TLVs. This would seem to be more than sufficient for an intelligent network design. Nevertheless, the 256-fragment maximum has been the cause for some concern that IS-IS might not scale into the future, given the evolution of large-scale networks. Among the contributing factors to larger and larger LSPs are:

- The ongoing extension of the protocol, requiring new TLVs, for support of things like traffic engineering and IPv6
- The injection of more and more prefixes into the domain for increased routing precision
- The advent of multi-chassis routing platforms for very large core networks, which can easily support thousands or tens of thousands of links and adjacencies

Whether the 256-fragment limitation turns out to be imagined or real, an extension now permits IS-IS to exceed that limit.[12] Looking once more at the fragments in Figure 8.41, you can see that the identical LSP ID identifies the fragments as belonging to the same

[12] Amir Hermelin, Stefano Previdi, and Mike Shand, "Extending the Number of Intermediate System to Intermediate System (IS-IS) Link State PDU (LSP) Fragments Beyond the 256 Limit," RFC 3786, May 2004.

originator; the LSP number differentiates the fragments from one another. And of course, the LSP ID is based on the system ID assigned to the originator. So, overcoming the 256-fragment limitation becomes a simple matter of assigning more than one SysID to the same router. This additional SysID is then viewed as identifying a *virtual system* attached to the originating router. For every additional SysID given to it, an originator can generate an additional 256 LSP fragments.

The SPF calculation ordinarily would view the virtual system as a separate node from the originating system. To circumvent this potential problem, the virtual system is instead viewed as leaf node from the originator and is calculated as a cost of 0 from the originator. In this, the concept is similar to that of a pseudonode.

The additional SysIDs for the virtual systems are advertised in type 24 TLVs (Figure 8.42), called the IS Alias ID TLV. This TLV must always be included, if it exists, in fragment 0 of the originator's LSP, and notifies other routers that the specified additional SysIDs are a part of the originator's LSPs.

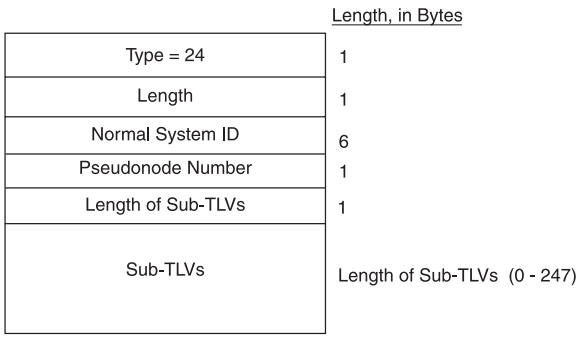

Figure 8.42 Type 24 TLVs enable the use of additional SysIDs to overcome the IS-IS 256-fragment limitation.

- **Normal System ID** is the SysID of the originating router.
- **Pseudonode Number** acts to relate the additional SysIDs with the normal SysID.
- **Sub-TLVs** include the additional SysIDs (also called IS-Alias-IDs).

8.4 Overloading

The previous section dealt with the issue of adjusting the size of LSAs and LSPs to existing link bandwidth—the assumption being that in large networks OSPF Updates and IS-IS can become very large. There is a "flip side" to this issue: What happens when a router's memory is not large enough to store all of the LSA/LSPs being flooded in an area? We know that all link state databases in an area must be identical. If the memory allocated to storing the link

state database becomes full so that not all information can be recorded, the SPF calculation at that router will likely be incorrect. The router then cannot be trusted to route correctly on the shortest-path tree and should not be included as a transit node.

IS-IS has a facility for coping with this situation. If the database memory fills up—that is, if the memory becomes overloaded—the router notifies the other routers in the area by setting the *Overload* (OL) bit in its LSAs (Figure 8.43). The other routers in the area treat an overloaded router as a leaf router on the shortest-path tree: It is included for reachability of its directly connected links, but not as a transit path to other routers.

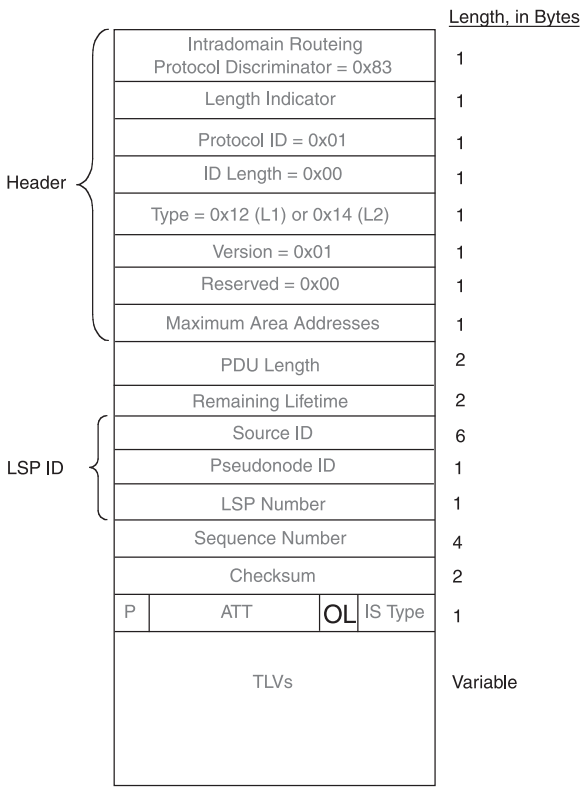

Figure 8.43 The Overload (OL) bit, when set, signals to other routers in the area that the originator's database memory is overloaded.

Like several other features of OSPF and IS-IS, both of which were created in the days when routers were sharply limited in CPU power, memory capacity, and throughput time, the original intention of the overload mechanism is now mostly irrelevant. Modern core routers have enormous memory capacity that will not be overloaded by IS-IS in any intelligently designed network.[13]

[13] A not-so-bright design is one in which a low-powered, low-memory router is exposed to a large core network. Such routers, if they are used at all, must be protected behind static routes or in totally stubby areas.

However, unlike other now-obsolete features of OSPF and IS-IS, the OL bit is tremendously useful in modern networks for a reason that has nothing to do with memory depletion: helping to prevent unintentional blackholing of packets in BGP transit networks. To understand this function, you must understand some basics of BGP. Figure 8.44 shows a very simple BGP transit network, AS 65502. BGP is used for routing through the AS between AS 65501 and AS 65503. BGP is a point-to-point protocol, running over TCP sessions. The sessions running between the autonomous systems are External BGP (EBGP), and the sessions running internal to an autonomous system are Internal BGP (IBGP). Whereas EBGP sessions are usually[14] between directly connected neighbors, IBGP sessions often pass through several routers internal to the AS. In Figure 8.44, the IBGP session between RTR A and RTR E passes through three internal routers. When RTR A and RTR E exchange routes learned from their EBGP neighbors, they show their RIDs (normally loopback addresses) as the next-hop address of the routes.[15] IBGP depends on the AS's IGP for two things:

- Finding the route through the AS to its IBGP peers for establishing its point-to-point TCP sessions
- Finding the routes to the next-hop addresses of the external routes advertised by its IBGP peers

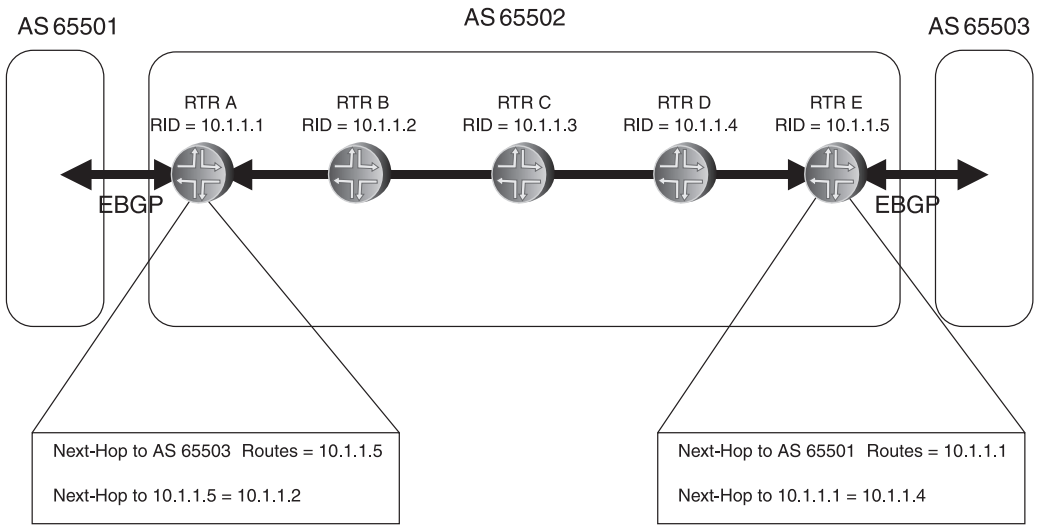

Figure 8.44 IBGP relies on the autonomous system's IGP to find its IBGP peers and the next-hop addresses of the external routes advertised by its IGP peers.

[14] EBGP sessions are occasionally multihop for some specialized applications, but that is irrelevant to this discussion.

[15] The default BGP behavior is to use the interface address of the external peer as the next-hop address of IBGP routes, but current best practice is to use a policy to change that next hop to the loopback address of the edge router advertising the prefixes to its IBGP peers.

If RTR A in Figure 8.44 needs to route a packet to a destination in AS 65503, it performs a route lookup and finds that the next hop for the route is RTR E, at 10.1.1.5. It then performs a second lookup for 10.1.1.5 and finds that the next hop for that destination is RTR B. The packet is then forwarded to RTR B. If the IBGP session shown in Figure 8.44 were the only IBGP session in the AS, there would now be a problem. Because RTR B has no IBGP session with RTR E, it cannot learn the external routes RTR E is advertising. So when RTR B receives the packet, it has no route entry for the destination in AS 65503 and drops the packet.

Therefore, in a transit AS, it is necessary to create a full mesh of IBGP sessions as shown in Figure 8.45.[16] Each internal router learns the external routes via IBGP. Now, when the packet in our example is forwarded from RTR A to RTR B, RTR B and subsequent routers along the path to RTR E can perform the correct lookups to get the packet to RTR E.

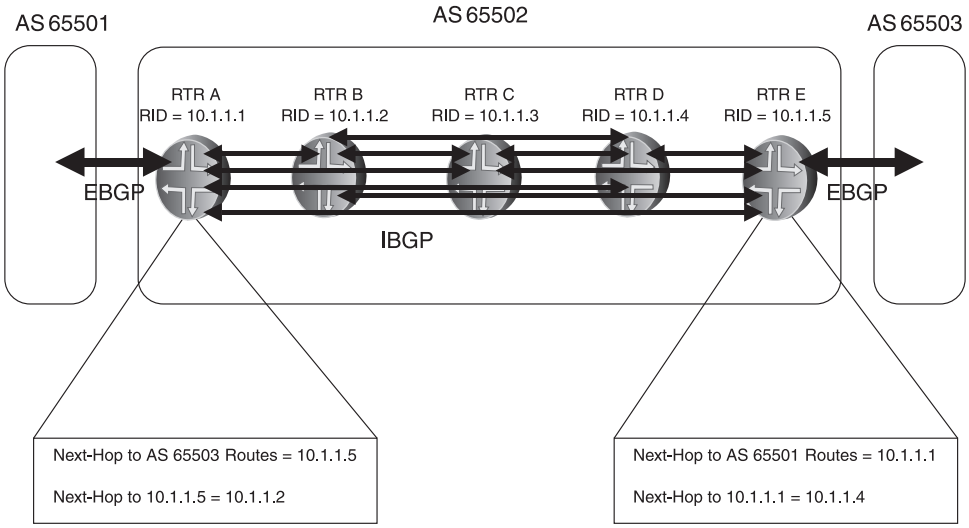

Figure 8.45 A full mesh of IBGP sessions is necessary for the non-EBGP routers to know how to forward packets transiting the AS.

But a potential problem lurks in the network, stemming from the nature of IGPs and BGP: OSPF and IS-IS converge very quickly, whereas BGP, which must first establish TCP sessions and then exchange tens or hundreds of thousands of routes, converges quite slowly. A BGP session carrying a full Internet routing table can take several minutes to converge. Suppose RTR C in Figure 8.45 is restarted intentionally or otherwise. The IGP routes will become known quickly, so that RTR A and RTR B again learn the route to RTR E through RTR C. If a packet to an AS 65503 destination is forwarded on the route after the IGP has converged but before IBGP has finished converging at RTR C, RTR C will not recognize the destination and will drop the packet.

[16] There are techniques—route reflectors and confederations—for avoiding a full IBGP mesh, which are again irrelevant to this discussion.

This is where the OL bit comes in handy. When a new IBGP router is added to the network, or a router is restarted, the IS-IS OL bit can be manually set. Because directly connected addresses on an overloaded router—including the loopback address that serves as an endpoint for IBGP sessions—are considered reachable by the other routers in the AS, IBGP can be brought up and can begin exchanging routes. However, the other routers will not include the overloaded router for transit traffic, and so will route packets transiting the AS on an alternate path. (In a correctly designed transit AS, unlike the simple example AS of Figure 8.45, there is always at least one alternate path.) When BGP has converged, the OL bit can be cleared and the router can then begin forwarding AS-transit packets.

In some special cases, you might want to leave the OL bit permanently set, so that the router has full network knowledge and other routers know its stub links, but the router is never used for transit traffic. Examples include routers that are connected for analysis purposes but should not be considered part of the production network, such as lab or network management routers, and IBGP route reflectors that you want to perform only as BGP route servers and not as transit nodes.

The OL bit is manually set in IOS with the command **set-overload-bit** and in JUNOS with the command **overload**. Removing the command from the configuration clears the OL bit. But remembering to set and then clear the OL bit whenever you add or restart a router is a hassle, and unplanned restarts give you no chance to use the OL bit. Both IOS and JUNOS have a very useful option—**set-overload-bit on-startup** in IOS and **overload timeout** in JUNOS—that allows you to specify a number of seconds that the OL bit is set automatically during startup. When the specified seconds expire—200 to 400 seconds is usually a reasonable period for allowing BGP to converge—the OL bit is automatically cleared. The option is highly recommended in any IBGP network.

OSPF has no comparable feature to the OL bit in its LSAs. However, some vendor implementations include a mechanism by which you can create the same kind of behavior in an OSPF area by setting the metric of all transit links on an "overloaded" router to 0xFFFF in its type 1 LSAs. This metric indicates that the links are unreachable, so that the router is not included as a transit node on the SPF tree. Stub links connected to the router are advertised with their normal metrics, so that they are still reachable when the router is in overload. OSPF overloading is supported in JUNOS, for example, using the same **overload** command, and allowing the same optional automatic overload timeout at startup, as with IS-IS.

Review Questions

1. Describe the modification to the simplistic SPF algorithm of Chapter 2 that allows ECMP.
2. Why is per-destination load balancing better than per-packet load balancing?
3. Why is per-flow load balancing better than per-destination load balancing?

4. What modification to the basic SPF algorithm is required, when there are equal costs involving both point-to-point links and broadcast links, to avoid disruption of ECMP?
5. What is Incremental SPF, and in what two general cases can it improve the efficiency of SPF in an area?
6. What is partial route calculation, and how does it improve the efficiency of SPF calculations?
7. How does SPF delay improve the efficiency of SPF calculations?
8. How can an excessive SPF delay reduce the performance of a network?
9. What is adaptive SPF delay, and what is its advantage over a set delay period?
10. Why are individual refresh timers for each self-originated LSA or LSP in a database better than a single monolithic timer?
11. How does adding a delay period to the refresh timers of self-originated LSAs or LSPs reduce problems associated with heavy flooding?
12. Under what circumstances can increasing the time a router waits for an acknowledgement of a transmitted LSA or LSP before retransmitting improve flooding efficiency?
13. What is a mesh group?
14. What are the advantages and disadvantages of using blocked mode in a mesh group?
15. What are the advantages and disadvantages of using set mode in a mesh group?
16. Can the three mesh group modes (inactive, blocked, and set) be used in the same network?
17. What is a demand circuit?
18. What is the OSPF DoNotAge bit?
19. What two changes are made to OSPF procedures in the extensions for support of demand circuits?
20. What is the purpose of the DC bit in the Options field of LSAs?
21. What is the purpose of the DC bit in the Options field of OSPF Hello and Database Description packets?
22. What does the presumption of reachability mean in regard to OSPF over demand circuits?
23. In what way can OSPF over demand circuits effect the consistency of LSA sequence numbers, and in what way can this be a problem?
24. What is OSPF flood reduction?
25. Why is fragmentation less of an issue with OSPF than with IS-IS?
26. What is the IS-IS ReceiveLSPBufferSize, and why are Hellos padded to this size during when attempting to form an adjacency?

27. What is the purpose of the LSP Number in the LSP header?
28. What happens if LSP Number 0 of a fragmented LSP is missing in a link state database?
29. What is the purpose of the IS-Alias-ID (type 24) TLV?
30. What does it mean when an IS-IS router is overloaded?
31. How is the OL bit useful in BGP transit networks?
32. What is the advantage of enabling an overload timeout option in BGP transit networks?
33. How can "overloading" behavior be created in an OSPF area?