Hacer Ppt agregándolo al que ya está

CHAPTER

11

Unicast Routing Protocols (RIP, OSPF, and BGP)

As we have discussed in some previous chapters, unicast communication means communication between one sender and one receiver, a one-to-one communication. In this chapter, we discuss how the routers create their routing tables to support unicast communication. We show how the Internet is divided into administrative areas known as autonomous systems to efficiently handle the exchange of routing information. We then explain two dominant routing protocols used inside an autonomous system and one routing protocol used for exchange of routing information between autonomous systems.

OBJECTIVES

The chapter has several objectives:

- To introduce the idea of autonomous systems (ASs) that divide the Internet into smaller administrative regions for the purpose of exchanging routing information.
- To discuss the idea of distance vector routing as the first intra-AS routing method and how it uses the Bellman-Ford algorithm to update routing tables.
- To discuss how the Routing Information Protocol (RIP) is used to implement the idea of distance vector routing in the Internet.
- To discuss the idea of link state routing as the second intra-AS routing method and how it uses Dijkstra algorithm to update the routing tables.
- To discuss how Open Shortest Path First (OSPF) is used to implement the idea of link state routing in the Internet.
- To discuss the idea of path vector routing as the dominant inter-AS routing method and explain the concept of policy routing.
- To discuss how Border Gateway Protocol (BGP) is used to implement the idea of path vector routing in the Internet.

11.1 INTRODUCTION

An internet is a combination of networks connected by routers. When a datagram goes from a source to a destination, it will probably pass through many routers until it reaches the router attached to the destination network.

Cost or Metric

A router receives a packet from a network and passes it to another network. A router is usually attached to several networks. When it receives a packet, to which network should it pass the packet? The decision is based on optimization: Which of the available pathways is the optimum pathway? What is the definition of the term *optimum*?

One approach is to assign a **cost** for passing through a network. We call this cost a **metric.** High cost can be thought of as something *bad*; low cost can be thought of something *good*. For example, if we want to maximize the throughput in a network, the high throughput means low cost and the low throughput means high cost. As another example, if we want to minimize the delay, low delay is low cost and high delay is high cost.

Static versus Dynamic Routing Tables

A routing table can be either static or dynamic. A *static table* is one with manual entries. A *dynamic table,* on the other hand, is one that is updated automatically when there is a change somewhere in the internet. Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a link is down, and they need to be updated whenever a better route has been found.

Routing Protocol

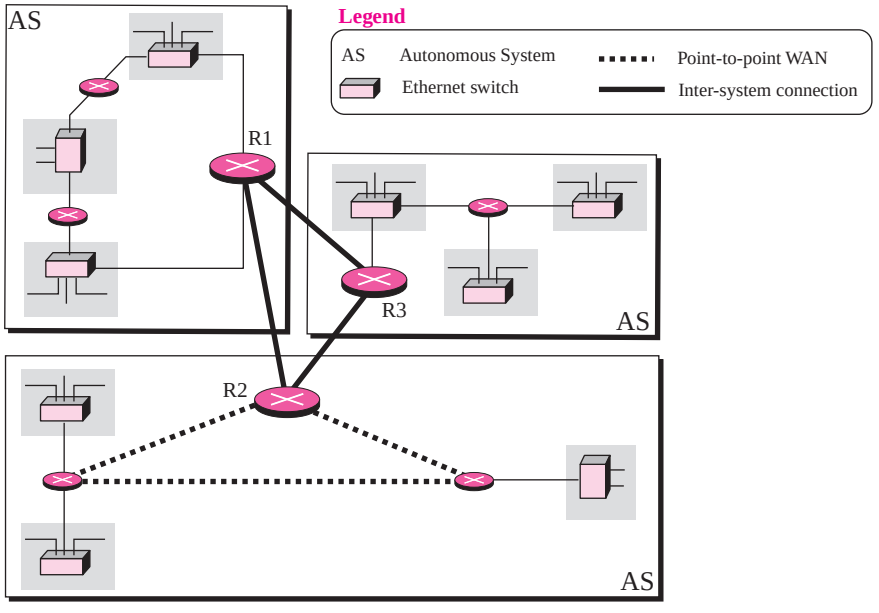
Routing protocols have been created in response to the demand for dynamic routing tables. A routing protocol is a combination of rules and procedures that lets routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhood. The sharing of information allows a router in San Francisco to know about the failure of a network in Texas. The routing protocols also include procedures for combining information received from other routers.

Routing protocols can be either an *interior protocol* or an *exterior protocol*. An interior protocol handles *intradomain routing*; an exterior protocol handles *interdomain routing*. We start the next section with defining these terms.

11.2 INTRA- AND INTER-DOMAIN ROUTING

Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems. An **autonomous system (AS)** is a group of networks and routers under the authority of a single administration. Routing inside an autonomous system is referred to as *intra-domain routing*. Routing between autonomous systems is referred to as *inter-domain routing*. Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system. However, only one interdomain routing protocol handles routing between autonomous systems. See Figure 11.1.

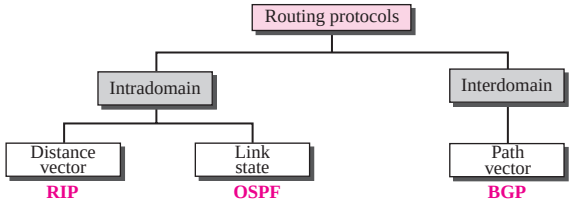
Figure 11.1 *Autonomous systems*



Several intra-domain and inter-domain routing protocols are in use. In this chapter, we cover only the most popular ones. We discuss two intra-domain routing protocols: distance vector and link state. We also introduce one inter-domain routing protocol: path vector (see Figure 11.2).

Routing Information Protocol (RIP) is the implementation of the distance vector protocol. Open Shortest Path First (OSPF) is the implementation of the link state protocol. Border Gateway Protocol (BGP) is the implementation of the path vector protocol. RIP and OSPF are interior routing protocols; BGP is an exterior routing protocol.

Figure 11.2 *Popular routing protocols*



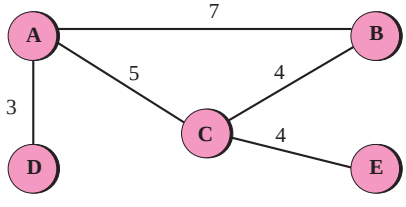
11.3 DISTANCE VECTOR ROUTING

We first discuss **distance vector routing.** This method sees an AS, with all routers and networks, as a *graph*, a set of nodes and lines (edges) connecting the nodes. A router can normally be represented by a node and a network by a link connecting two nodes, although other representations are also possible. The graph theory used an algorithm called Bellman-Ford (also called Ford-Fulkerson) for a while to find the shortest path between nodes in a graph given the distance between nodes. We first discuss this algorithm before we see how it can be modified to be used for updating routing tables in a distance vector routing.

Bellman-Ford Algorithm

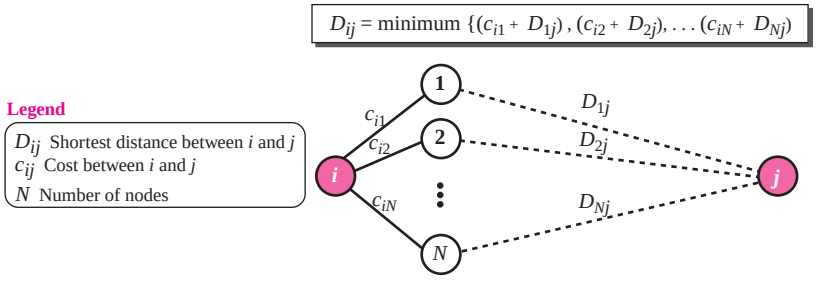
Let us briefly discuss the **Bellman-Ford algorithm.** The algorithm can be used in many applications in graph theory. If we know the cost between each pair of nodes, we can use the algorithm to find the least cost (shortest path) between any two nodes. Figure 11.3 shows a map with nodes and lines. The cost of each line is given over the line; the algorithm can find the least cost between any two nodes. For example, if the nodes represent cities and the lines represent roads connecting them, the graph can find the shortest distance between any two cities.

Figure 11.3 *A graph for the Bellman-Ford algorithm*



The algorithm is based on the fact that if all neighbors of node *i* know the shortest distance to node *j*, then the shortest distance between node *i* and *j* can be found by adding the distance between node *i* and each neighbor to the neighbor's shortest distance to node *j* and then select the minimum, as shown in Figure 11.4.

Figure 11.4 *The fact behind Bellman-Ford algorithm*



Although the principle of the Bellman-Ford algorithm is very simple and intuitive, the algorithm itself looks circular. How have the neighbors calculated the shortest path to the destination? To solve the problem, we use iteration. We create a shortest distance table (vector) for each node using the following steps:

1. The shortest distance and the cost between a node and itself is initialized to 0.
2. The shortest distance between a node and any other node is set to infinity. The cost between a node and any other node should be given (can be infinity if the nodes are not connected).
3. The algorithm repeat as shown in Figure 11.4 until there is no more change in the shortest distance vector.

Table 11.1 shows the algorithm in pseudocode.

Table 11.1 *Bellman-Ford Algorithm*

```
1  Bellman_Ford ( )
2  {
3     // Initialization
4     for (i = 1 to N; for j = 1 to N)
5     {
6        if(i == j)  Dij = 0   cij = 0
7        else        Dij = ∞   cij = cost between i and j
8     }
9     // Updating
10    repeat
11    {
12       for (i = 1 to N; for j = 1 to N)
13       {
14          Dij  ← minimum [(ci1 + D1j)  ... (ciN + DNj)]
15       } // end for
16    } until (there was no change in previous iteration)
17 } // end Bellman-Ford
```

Distance Vector Routing Algorithm

The Bellman-Ford algorithm can be very well applied to a map of roads between cities because we can have all of the initial information about each node at the same place. We can enter this information into the computer and let the computer hold the intermediate results and create the final vectors for each node to be printed. In other words, the algorithm is designed to create the result *synchronously*. If we want to use the algorithm for creating the routing table for routers in an AS, we need to change the algorithm:

1. In distance vector routing, the cost is normally hop counts (how many networks are passed before reaching the destination). So the cost between any two neighbors is set to 1.
2. Each router needs to update its routing table *asynchronously*, whenever it has received some information from its neighbors. In other words, each router executes part of the whole algorithm in the Bellman-Ford algorithm. Processing is *distributive*.
3. After a router has updated its routing table, it should send the result to its neighbors so that they can also update their routing table.
4. Each router should keep at least three pieces of information for each route: destination network, the cost, and the next hop. We refer to the whole routing table as Table, to the row i in the table as Table_i, to the three columns in row i as Table_i.dest, Table_i.cost, and Table_i.next.
5. We refer to information about each route received from a neighbor as R (record), which has only two pieces of information: R.dest and R.cost. The next hop is not included in the received record because it is the source address of the sender.

Table 11.2 shows the algorithm in pseudocode.

Table 11.2 *Distance Vector Algorithm Used by Each Router*

```
1  Distance_Vector_Algorithm ( )
2  {
3      // At startup
4     for (i = 1 to N)          // N is number of ports
5     {
6         Table_i.dest = address of the attached network
7         Table_i.cost = 1
8         Table_i.next = —       // Means at home
9         Send a record R about each row to each neighbor
10    } // end for loop
11
12      // Updating
13    repeat (forever)
14    {
15        Wait for arrival of a record R from a neighbor
16        Update (R, T)          // Call update module
17        for (i = 1 to N)       // N is the current table size
```

Table 11.2 *Distance Vector Algorithm Used by Each Router (continued)*

```
18        {
19              Send a record R about each row to each neighbor
20        }
21       }  // end repeat
22
23  }  // end Distance_Vector
24  Update (R, T)          // Update module
25  {
26      Search T for a destination matching the one in R
27      if (destination is found in row i)
28      {
29          if (R.cost + 1 < T_i.cost    or    R.next == T_i.next)
30          {
31                T_i.cost = R.cost + 1
32                T_i.next = Address of sending router
33          }
34       else  discard  the  record     // No change is needed
35       }
36       else
37        // Insert the new router
38       {
39           T_N+1.dest = R.dest
40           T_N+1.cost = R.cost + 1
41           T_N+1.next = Address of sending router
42           Sort the table according to destination address
43       }
44  }  // end of Update module
```

Lines 4 to 10 show the initialization at the start-up. The router creates a preliminary routing table that can only be used to route packets to the networks directly attached to its interfaces. For each entry in the routing table, it sends a record with only two fields: destination address and the cost to each neighbor.

The router updates itself whenever it receives a record from a neighbor. After each update, the route sends a record for each entry in the routing table to its neighbors to let them also update themselves.

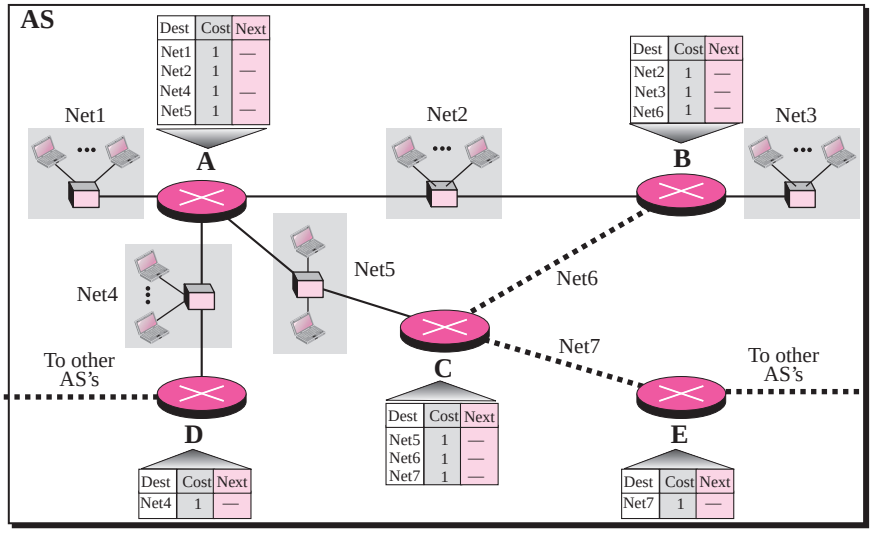
Lines 23 to 47 give the details of updating process. When a record arrives, the router searches for the destination address in the routing table.

1. If the corresponding entry is found, two cases need to be checked and the route information should be changed.
   a. If the record cost plus 1 is smaller than the corresponding cost in the table, it means the neighbors have found a better route to that destination.
   b. If the next hop is the same, it means some change has happened in some part of the network. For example, suppose a neighbor has previously advertised a route to a destination with cost 3, but now there is no path between this neighbor and that destination. The neighbor advertises this destination with cost value infinity. The receiving router must not ignore this value even though its old entry is smaller. The old route does not exist any more.
2. If the entry is not in the table, the router adds it to the table and sorts the table according to the destination address.

Example 11.1

Figure 11.5 shows the initial routing table for an AS. Note that the figure does not mean that all routing tables have been created at the same time; each router creates its own routing table when it is booted.

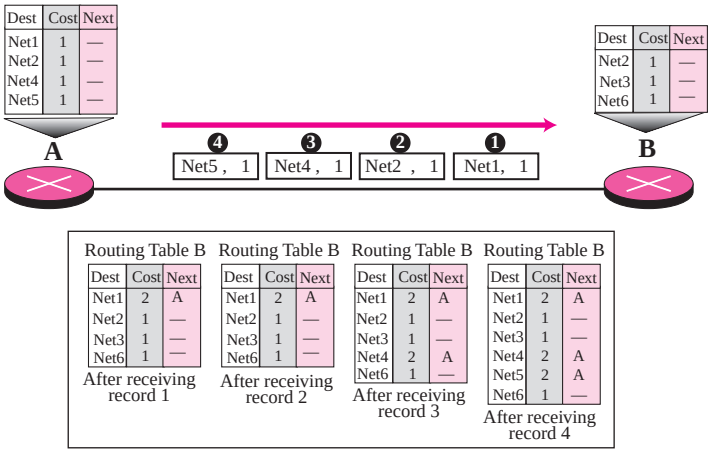
Figure 11.5 *Example 11.1*



Example 11.2

Now assume router A sends four records to its neighbors, routers B, D, and C. Figure 11.6 shows the changes in B's routing table when it receives these records. We leave the changes in the routing tables of other neighbors as exercise.

Figure 11.6 *Example 11.2*



a. When router B receives record 1, it searches its routing table for the route to net1, and since it is not found there, it adds one hop to the cost (distance between B and A) and adds it to the table with the next hop to be A.

b. When router B receives record 2, it searches its routing table and finds the destination net2 there. However, since the announced cost plus 1 is larger than the cost in the table, the record is discarded.

c. When router B receives record 3, it searches its router, and since Net4 is not found, it is added to the table.

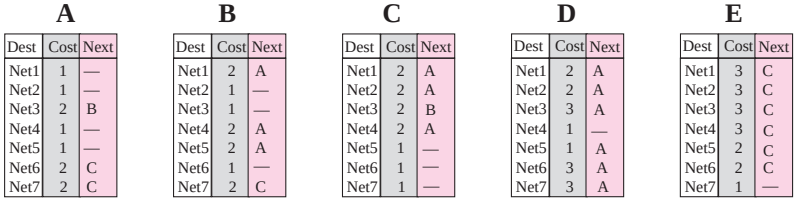
d. When router B receives record 4, it searches its router, and since Net5 is not found, it is added to the table.

Now router B has more information, but it is not complete. Router B does not even know that net7 exists. More updating is required.

Example 11.3

Figure 11.7 shows the final routing tables for routers in Figure 11.5.

Figure 11.7 *Example 11.3*



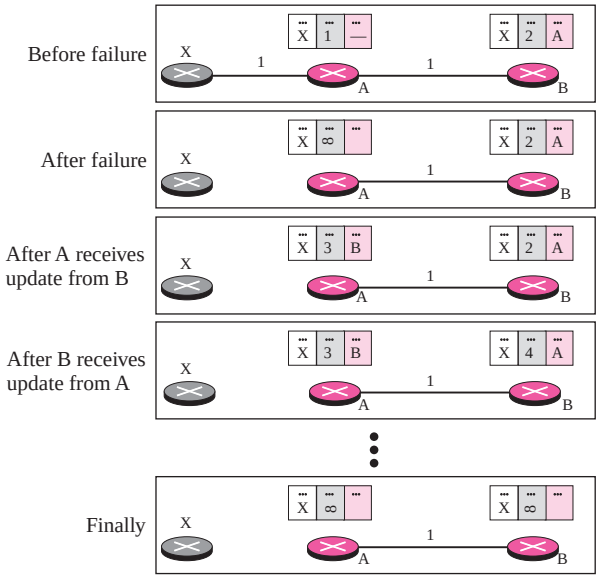
Count to Infinity

A problem with distance vector routing is that any decrease in cost (good news) propagates quickly, but any increase in cost (bad news) propagates slowly. For a routing protocol to work properly, if a link is broken (cost becomes infinity), every other router should be aware of it immediately, but in distance vector routing, this takes some time. The problem is referred to as **count to infinity.** It takes several updates before the cost for a broken link is recorded as infinity by all routers.

Two-Node Loop

One example of count to infinity is the two-node loop problem. To understand the problem, let us look at the scenario depicted in Figure 11.8.

Figure 11.8 *Two-node instability*



The figure shows a system with three nodes. We have shown only the portions of the routing table needed for our discussion. At the beginning, both nodes A and B know how to reach node X. But suddenly, the link between A and X fails. Node A changes its table. If A can send its table to B immediately, everything is fine. However, the system becomes unstable if B sends its routing table to A before receiving A's routing table. Node A receives the update and, assuming that B has found a way to reach X, immediately updates its routing table. Now A sends its new update to B. Now B thinks that something has been changed around A and updates its routing table. The cost of reaching X increases gradually until it reaches infinity. At this moment, both A and B know that X cannot be reached. However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A

receives a packet destined for X, it goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem. A few solutions have been proposed for instability of this kind.

Defining Infinity The first obvious solution is to redefine infinity to a smaller number, such as 16. For our previous scenario, the system will be stable in fewer updates. As a matter of fact, most implementations of the Distance Vector Protocol define 16 as infinity. However, this means that distance vector cannot be used in large systems. The size of the network, in each direction, can not exceed 15 hops.

Split Horizon Another solution is called **split horizon.** In this strategy, instead of flooding the table through each interface, each node sends only part of its table through each interface. If, according to its table, node B thinks that the optimum route to reach X is via A, it does not need to advertise this piece of information to A; the information has come from A (A already knows). Taking information from node A, modifying it, and sending it back to node A is what creates the confusion. In our scenario, node B eliminates the last line of its routing table before it sends it to A. In this case, node A keeps the value of infinity as the distance to X. Later, when node A sends its routing table to B, node B also corrects its routing table. The system becomes stable after the first update: both node A and B know that X is not reachable.

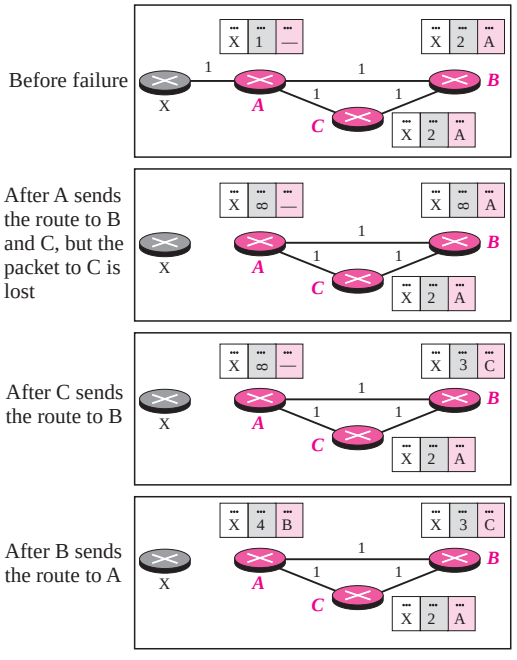
Split Horizon and Poison Reverse Using the split horizon strategy has one drawback. Normally, the Distance Vector Protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess that this is due to the split horizon strategy (the source of information was A) or because B has not received any news about X recently. The split horizon strategy can be combined with the **poison reverse** strategy. Node B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

Three-Node Instability

The two-node instability can be avoided using split horizon combined with poison reverse. However, if the instability is between three nodes, stability cannot be guaranteed. Figure 11.9 shows the scenario.

Suppose, after finding that X is not reachable, node A sends a packet to B and C to inform them of the situation. Node B immediately updates its table, but the packet to C is lost in the network and never reaches C. Node C remains in the dark and still thinks that there is a route to X via A with a distance of 2. After a while, node C sends its routing table to B, which includes the route to X. Node B is totally fooled here. It receives information on the route to X from C, and according to the algorithm, it updates its table showing the route to X via C with a cost of 3. This information has come from C, not from A, so after awhile node B may advertise this route to A. Now A is fooled and updates its table to show that A can reach X via B with a cost of 4. Of course, the loop continues; now A advertises the route to X to C, with increased cost, but not to B. C then advertises the route to B with an increased cost. B does the same to A. And so on. The loop stops when the cost in each node reaches infinity.

Figure 11.9 *Three-node instability*



11.4 RIP

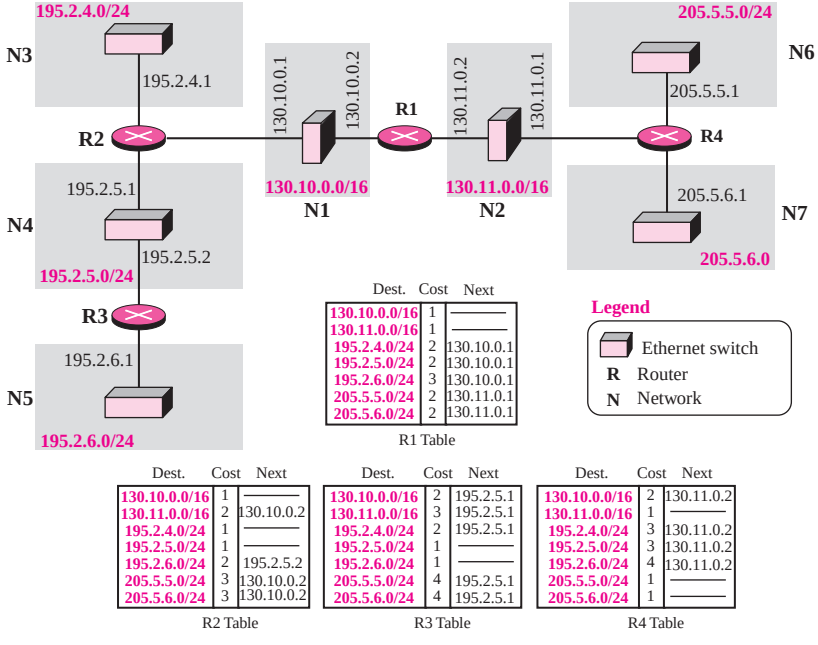
The **Routing Information Protocol (RIP)** is an intradomain (interior) routing protocol used inside an autonomous system. It is a very simple protocol based on distance vector routing. RIP implements distance vector routing directly with some considerations:

1. In an autonomous system, we are dealing with routers and networks (links), what was described as a node.
2. The destination in a routing table is a network, which means the first column defines a network address.
3. The metric used by RIP is very simple; the distance is defined as the number of links (networks) that have to be used to reach the destination. For this reason, the metric in RIP is called a **hop count.**
4. Infinity is defined as 16, which means that any route in an autonomous system using RIP cannot have more than 15 hops.
5. The next node column defines the address of the router to which the packet is to be sent to reach its destination.

Figure 11.10 shows an autonomous system with seven networks and four routers. The table of each router is also shown. Let us look at the routing table for R1. The table has seven entries to show how to reach each network in the autonomous system. Router R1 is

directly connected to networks 130.10.0.0 and 130.11.0.0, which means that there are no next hop entries for these two networks. To send a packet to one of the three networks at the far left, router R1 needs to deliver the packet to R2. The next node entry for these three networks is the interface of router R2 with IP address 130.10.0.1. To send a packet to the two networks at the far right, router R1 needs to send the packet to the interface of router R4 with IP address 130.11.0.1. The other tables can be explained similarly.

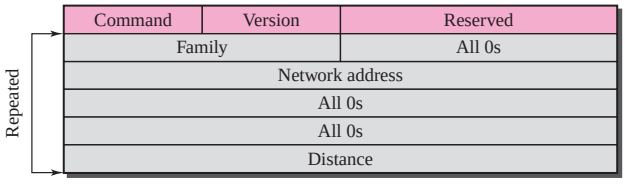
Figure 11.10 *Example of a domain using RIP*



RIP Message Format

The format of the RIP message is shown in Figure 11.11.

Figure 11.11 *RIP message format*



- **Command.** This 8-bit field specifies the type of message: request (1) or response (2).
- **Version.** This 8-bit field defines the version. In this book we use version 1, but at the end of this section, we give some new features of version 2.
- **Family.** This 16-bit field defines the family of the protocol used. For TCP/IP the value is 2.
- **Network address.** The address field defines the address of the destination network. RIP has allocated 14 bytes for this field to be applicable to any protocol. However, IP currently uses only 4 bytes. The rest of the address is filled with 0s.
- **Distance.** This 32-bit field defines the hop count (cost) from the advertising router to the destination network.

Note that part of the message is repeated for each destination network. We refer to this as an *entry.*

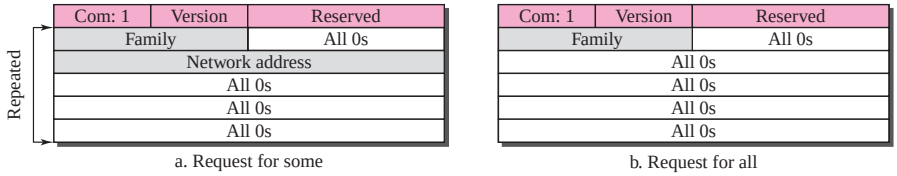
Requests and Responses

RIP has two types of messages: request and response.

Request

A request message is sent by a router that has just come up or by a router that has some time-out entries. A request can ask about specific entries or all entries (see Figure 11.12).

Figure 11.12 *Request messages*



Response

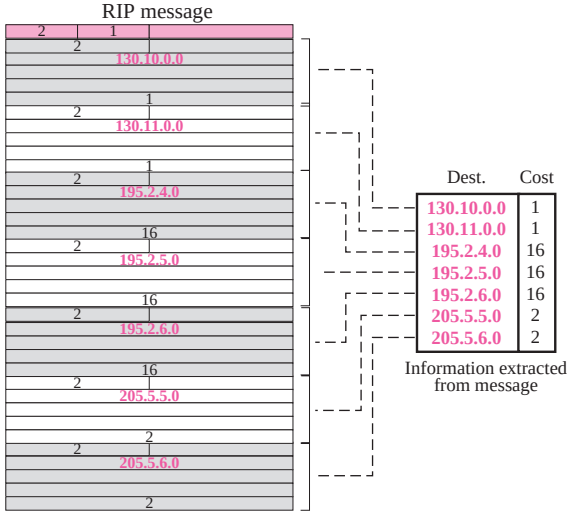
A response can be either solicited or unsolicited. A *solicited response* is sent only in answer to a request. It contains information about the destination specified in the corresponding request. An *unsolicited response,* on the other hand, is sent periodically, every 30 seconds or when there is a change in the routing table. The response is sometimes called an update packet. Figure 11.11 shows the response message format.

Example 11.4

Figure 11.13 shows the update message sent from router R1 to router R2 in Figure 11.10. The message is sent out of interface 130.10.0.2.

The message is prepared with the combination of split horizon and poison reverse strategy in mind. Router R1 has obtained information about networks 195.2.4.0, 195.2.5.0, and 195.2.6.0 from router R2. When R1 sends an update message to R2, it replaces the actual value of the hop counts for these three networks with 16 (infinity) to prevent any confusion for R2. The figure also shows the table extracted from the message. Router R2 uses the source address of the IP datagram

Figure 11.13 *Solution to Example 11.4*

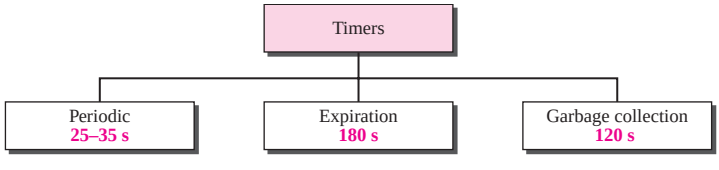


carrying the RIP message from R1 (130.10.02) as the next hop address. Router R2 also increments each hop count by 1 because the values in the message are relative to R1, not R2.

Timers in RIP

RIP uses three timers to support its operation (see Figure 11.14). The periodic timer controls the sending of messages, the expiration timer governs the validity of a route, and the garbage collection timer advertises the failure of a route.

Figure 11.14 *RIP timers*



Periodic Timer

The **periodic timer** controls the advertising of regular update messages. Although the protocol specifies that this timer must be set to 30 s, the working model uses a random number between 25 and 35 s. This is to prevent any possible synchronization and therefore overload on an internet if routers update simultaneously.

Each router has one periodic timer that is randomly set to a number between 25 and 35. It counts down; when zero is reached, the update message is sent, and the timer is randomly set once again.

Expiration Timer

The **expiration timer** governs the validity of a route. When a router receives update information for a route, the expiration timer is set to 180 s for that particular route. Every time a new update for the route is received, the timer is reset. In normal situations this occurs every 30 s. However, if there is a problem on an internet and no update is received within the allotted 180 s, the route is considered expired and the hop count of the route is set to 16, which means the destination is unreachable. Every route has its own expiration timer.

Garbage Collection Timer

When the information about a route becomes invalid, the router does not immediately purge that route from its table. Instead, it continues to advertise the route with a metric value of 16. At the same time, a timer called the **garbage collection timer** is set to 120 s for that route. When the count reaches zero, the route is purged from the table. This timer allows neighbors to become aware of the invalidity of a route prior to purging.

Example 11.5

A routing table has 20 entries. It does not receive information about five routes for 200 s. How many timers are running at this time?

Solution

The 21 timers are listed below:

Periodic timer: 1

Expiration timer: $20 - 5 = 15$

Garbage collection timer: 5

RIP Version 2

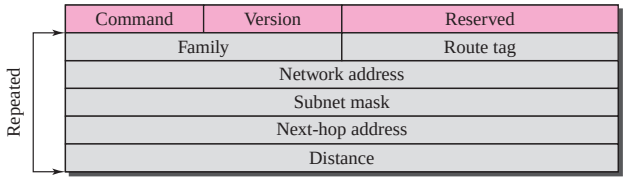
RIP version 2 was designed to overcome some of the shortcomings of version 1. The designers of version 2 have not augmented the length of the message for each entry. They have only replaced those fields in version 1 that were filled with 0s for the TCP/IP protocol with some new fields.

Message Format

Figure 11.15 shows the format of a RIP version 2 message. The new fields of this message are as follows:

- **Route tag.** This field carries information such as the autonomous system number. It can be used to enable RIP to receive information from an interdomain routing protocol.
- **Subnet mask.** This is a 4-byte field that carries the subnet mask (or prefix). This means that RIP2 supports classless addressing and CIDR.
- **Next-hop address.** This field shows the address of the next hop. This is particularly useful if two autonomous systems share a network (a backbone, for example). Then the message can define the router, in the same autonomous system or another autonomous system, to which the packet next goes.

Figure 11.15 *RIP version 2 format*



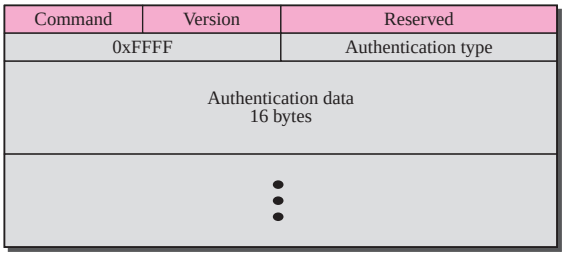
Classless Addressing

Probably the most important difference between the two versions of RIP is classful versus classless addressing. RIPv1 uses classful addressing. The only entry in the message format is the network address (with a default mask). RIPv2 adds one field for the subnet mask, which can be used to define a network prefix length. This means that in this version, we can use classless addressing. A group of networks can be combined into one prefix and advertised collectively, as we saw in Chapters 5 and 6.

Authentication

Authentication is added to protect the message against unauthorized advertisement. No new fields are added to the packet; instead, the first entry of the message is set aside for authentication information. To indicate that the entry is authentication information and not routing information, the value of $FFFF_{16}$ is entered in the family field (see Figure 11.16). The second field, the authentication type, defines the protocol used for authentication, and the third field contains the actual authentication data.

Figure 11.16 *Authentication*



Multicasting

Version 1 of RIP uses broadcasting to send RIP messages to every neighbor. In this way, all the routers on the network receive the packets, as well as the hosts. RIP version 2, on the other hand, uses the all-router multicast address to send the RIP messages only to RIP routers in the network.

Encapsulation

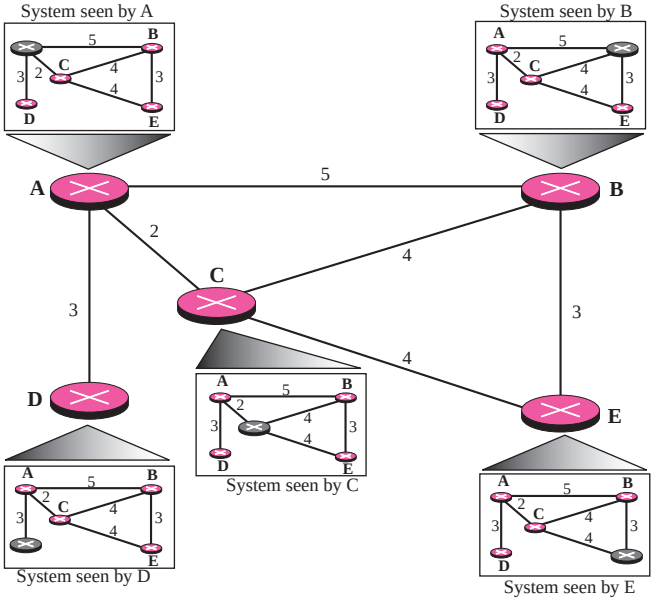
RIP messages are encapsulated in UDP user datagrams. A RIP message does not include a field that indicates the length of the message. This can be determined from the UDP packet. The well-known port assigned to RIP in UDP is port 520.

RIP uses the services of UDP on well-known port 520.

11.5 LINK STATE ROUTING

Link state routing has a different philosophy from that of distance vector routing. In link state routing, if each node in the domain has the entire topology of the domain—the list of nodes and links, how they are connected including the type, cost (metric), and the condition of the links (up or down)—the node can use the **Dijkstra algorithm** to build a routing table. Figure 11.17 shows the concept.

Figure 11.17 *Concept of link state routing*

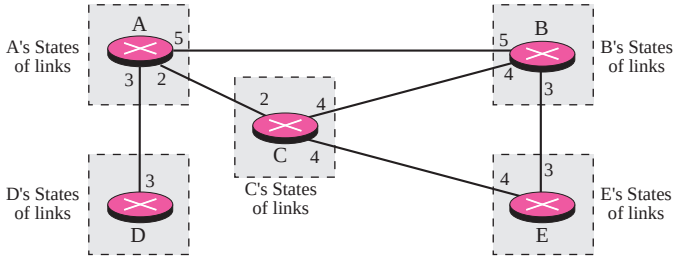


The figure shows a simple domain with five nodes. Each node uses the same topology to create a routing table, but the routing table for each node is unique because the calculations are based on different interpretations of the topology. This is analogous to a city map. Two persons in two different cities may have the same map, but each needs to take a different route to reach his destination.

The topology must be dynamic, representing the latest situation of each node and each link. If there are changes in any point in the network (a link is down, for example), the topology must be updated for each node.

How can a common topology be dynamic and stored in each node? No node can know the topology at the beginning or after a change somewhere in the network. Link state routing is based on the assumption that, although the global knowledge about the topology is not clear, each node has partial knowledge: it knows the state (type, condition, and cost) of its links. In other words, the whole topology can be compiled from the partial knowledge of each node. Figure 11.18 shows the same domain as in the previous figure, indicating the part of the knowledge belonging to each node.

Figure 11.18 *Link state knowledge*



Node A knows that it is connected to node B with metric 5, to node C with metric 2, and to node D with metric 3. Node C knows that it is connected to node A with metric 2, to node B with metric 4, and to node E with metric 4. Node D knows that it is connected only to node A with metric 3. And so on. Although there is an overlap in the knowledge, the overlap guarantees the creation of a common topology: a picture of the whole domain for each node.

Building Routing Tables

In **link state routing,** four sets of actions are required to ensure that each node has the routing table showing the least-cost node to every other node.

1. Creation of the states of the links by each node, called the link state packet or LSP.
2. Dissemination of LSPs to every other router, called **flooding,** in an efficient and reliable way.
3. Formation of a shortest path tree for each node.
4. Calculation of a routing table based on the shortest path tree.

Creation of Link State Packet (LSP)

A link state packet (LSP) can carry a large amount of information. For the moment, however, we assume that it carries a minimum amount of data: the node identity, the list of links, a sequence number, and age. The first two, node identity and the list of links, are needed to make the topology. The third, sequence number, facilitates flooding and

distinguishes new LSPs from old ones. The fourth, age, prevents old LSPs from remaining in the domain for a long time. LSPs are generated on two occasions:

1. *When there is a change in the topology of the domain.* Triggering of LSP dissemination is the main way of quickly informing any node in the domain to update its topology.
2. *On a periodic basis.* The period in this case is much longer compared to distance vector routing. As a matter of fact, there is no actual need for this type of LSP dissemination. It is done to ensure that old information is removed from the domain. The timer set for periodic dissemination is normally in the range of 60 minutes or 2 hours based on the implementation. A longer period ensures that flooding does not create too much traffic on the network.

Flooding of LSPs

After a node has prepared an LSP, it must be disseminated to all other nodes, not only to its neighbors. The process is called flooding and based on the following:

1. The creating node sends a copy of the LSP out of each interface.
2. A node that receives an LSP compares it with the copy it may already have. If the newly arrived LSP is older than the one it has (found by checking the sequence number), it discards the LSP. If it is newer, the node does the following:
   a. It discards the old LSP and keeps the new one.
   b. It sends a copy of it out of each interface except the one from which the packet arrived. This guarantees that flooding stops somewhere in the domain (where a node has only one interface).

Formation of Shortest Path Tree: Dijkstra Algorithm

After receiving all LSPs, each node will have a copy of the whole topology. However, the topology is not sufficient to find the shortest path to every other node; a **shortest path tree** is needed.

A tree is a graph of nodes and links; one node is called the root. All other nodes can be reached from the root through only one single route. A shortest path tree is a tree in which the path between the root and every other node is the shortest. What we need for each node is a shortest path tree with that node as the root. The **Dijkstra algorithm** is used to create a shortest path tree from a given graph. The algorithm uses the following steps:

1. **Initialization:** Select the node as the root of the tree and add it to the *path*. Set the shortest distances for all the root's neighbors to the cost between the root and those neighbors. Set the shortest distance of the root to zero.
2. **Iteration:** Repeat the following two steps until all nodes are added to the path:
   a. **Adding the next node to the path:** Search the nodes not in the *path*. Select the one with minimum shortest distance and add it to the *path*.
   b. **Updating:** Update the shortest distance for all remaining nodes using the shortest distance of the node just moved to the path in step 2.

$$D_j = \text{minimum } (D_j, D_i + c_{ij}) \quad \text{for all remaining nodes}$$

Table 11.3 shows the simple version of this algorithm.

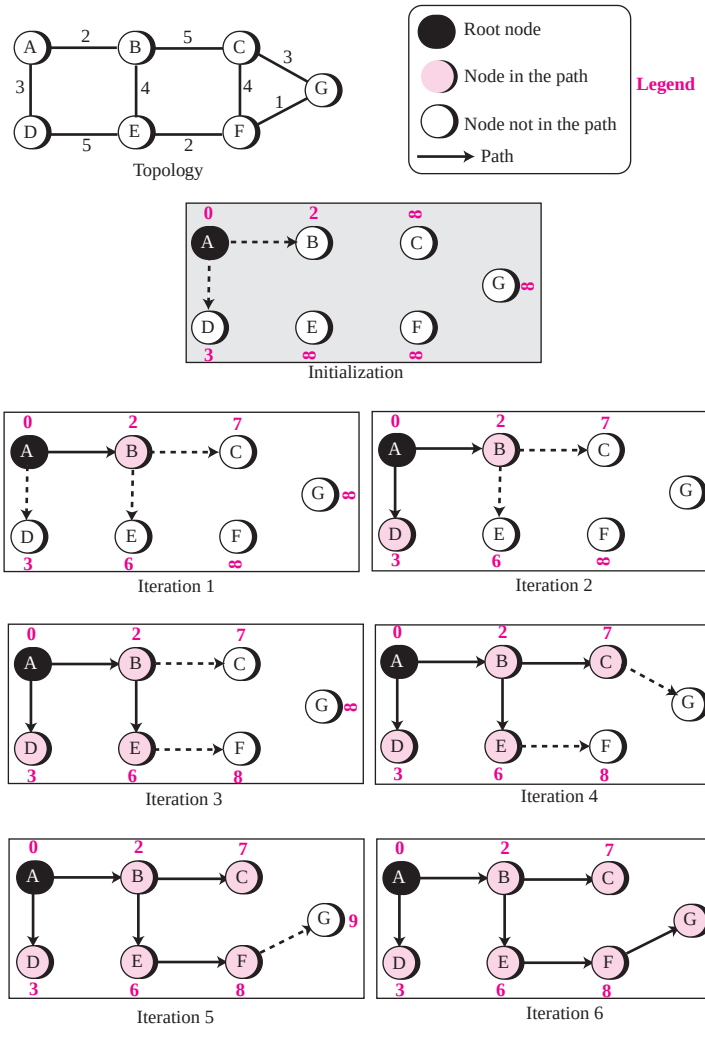
Table 11.3 *Dijkstra's Algorithm*

```
Dijkstra ( )
{
   // Initialization
   Path = {s}          // s means self
   for (i = 1 to N)
   {
      if(i is a neighbor of s and i ≠ s)   D_i = c_si
      if (i is not a neighbor of s)        D_i = ∞
   }
   D_s = 0

} // Dijkstra
   // Iteration
   Repeat
   {
       // Finding the next node to be added
    Path = Path ∪ i  if D_i is minimum among all remaining nodes

       // Update the shortest distance for the rest
    for (j = 1 to M)    // M number of remaining nodes
    {
         D_j = minimum (D_j ,  D_j  + c_ij)
     }
   } until (all nodes included in the path, M = 0)
```

Figure 11.19 shows the formation of the shortest path tree for the graph of seven nodes. All the nodes in the graph have the same topology, but each node creates a different shortest path tree with itself as the root of the tree.We show the tree created by node A. We need to go through an initialization step and six iterations to find the shortest tree.

In the initialization step, node A selects itself as the root. It then assigns shortest path distances to each node on the topology. The nodes that are not neighbors of A receive a shortest path distance value of infinity.

Figure 11.19 *Forming shortest path three for router A in a graph*

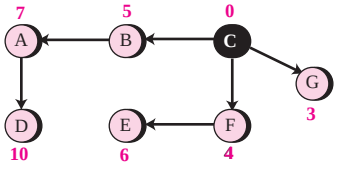


In each iteration, the next node with minimum distance is selected and added to the path. Then all shortest distances are updated with respect to the last node selected. For example, in the first iteration, node B is selected and added to the path and the shortest distances are updated with respect to node B (The shortest distances for C and E are changed, but for the others remain the same). After six iterations, the shortest path tree is found for node A. Note that in iteration 4, the shortest path to G is found via C, but in iteration 5, a new shortest route is discovered (via G); the previous path is erased and the new one is added.

Example 11.6

To show that the shortest path tree for each node is different, we found the shortest path tree as seen by node C (Figure 11.20). We leave the detail as an exercise.

Figure 11.20 *Example 11.6*



Calculation of Routing Table from Shortest Path Tree

Each node uses the shortest path tree found in the previous discussion to construct its routing table. The routing table shows the cost of reaching each node from the root. Table 11.4 shows the routing table for node A using the shortest path tree found in Figure 11.19.

Table 11.4 *Routing Table for Node A*

Destination	*Cost*	*Next Router*
A	0	—
B	2	—
C	7	B
D	3	—
E	6	B
F	8	B
G	9	B

11.6 OSPF

The **Open Shortest Path First (OSPF) protocol** is an intradomain routing protocol based on link state routing. Its domain is also an autonomous system.

Areas

To handle routing efficiently and in a timely manner, OSPF divides an autonomous system into areas. An **area** is a collection of networks, hosts, and routers all contained within an autonomous system. An autonomous system can be divided into many different areas. All networks inside an area must be connected.

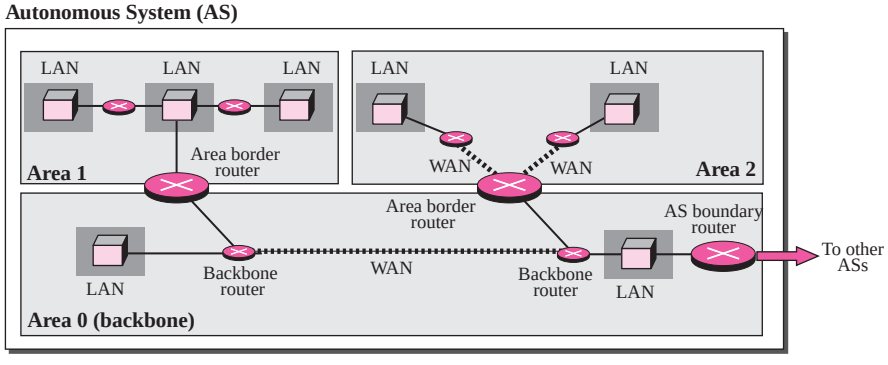
Routers inside an area flood the area with routing information. At the border of an area, special routers called **area border routers** summarize the information about the area and send it to other areas. Among the areas inside an autonomous system is a special area called the *backbone;* all of the areas inside an autonomous system must be

connected to the backbone. In other words, the backbone serves as a primary area and the other areas as secondary areas. This does not mean that the routers within areas cannot be connected to each other, however. The routers inside the backbone are called the ***backbone routers.*** Note that a backbone router can also be an area border router.

If, because of some problem, the connectivity between a backbone and an area is broken, a **virtual link** between routers must be created by the administration to allow continuity of the functions of the backbone as the primary area.

Each area has an area identification. The area identification of the backbone is zero. Figure 11.21 shows an autonomous system and its areas.

Figure 11.21 *Areas in an autonomous system*



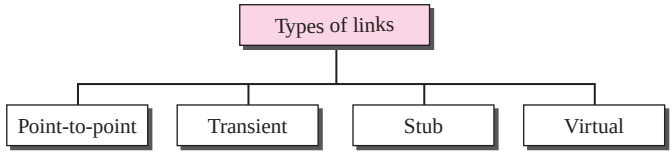
Metric

The OSPF protocol allows the administrator to assign a cost, called the **metric,** to each route. The metric can be based on a type of service (minimum delay, maximum throughput, and so on). As a matter of fact, a router can have multiple routing tables, each based on a different type of service.

Types of Links

In OSPF terminology, a connection is called a *link*. Four types of links have been defined: point-to-point, transient, stub, and virtual (see Figure 11.22).

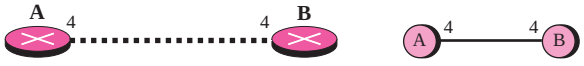
Figure 11.22 *Types of links*



Point-to-Point Link

A **point-to-point link** connects two routers without any other host or router in between. In other words, the purpose of the link (network) is just to connect the two routers. An example of this type of link is two routers connected by a telephone line or a T-line. There is no need to assign a network address to this type of link. Graphically, the routers are represented by nodes, and the link is represented by a bidirectional edge connecting the nodes. The metrics, which are usually the same, are shown at the two ends, one for each direction. In other words, each router has only one neighbor at the other side of the link (see Figure 11.23).

Figure 11.23 *Point-to-point link*

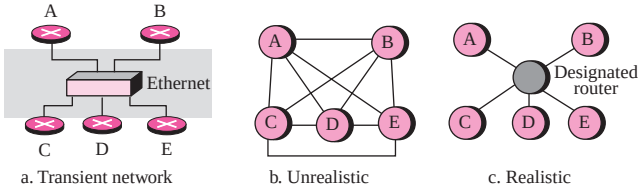


a. Point-to-point network b. Representation

Transient Link

A **transient link** is a network with several routers attached to it. The data can enter through any of the routers and leave through any router. All LANs and some WANs with two or more routers are of this type. In this case, each router has many neighbors. For example, consider the Ethernet in Figure 11.24a. Router A has routers B, C, D, and E as neighbors. Router B has routers A, C, D, and E as neighbors. If we want to show the neighborhood relationship in this situation, we have the graph shown in Figure 11.24b.

Figure 11.24 *Transient link*



a. Transient network b. Unrealistic c. Realistic

This is neither efficient nor realistic. It is not efficient because each router needs to advertise the neighborhood to four other routers, for a total of 20 advertisements. It is not realistic, because there is no single network (link) between each pair of routers; there is only one network that serves as a crossroad between all five routers.

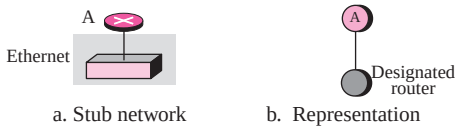
To show that each router is connected to every other router through one single network, the network itself is represented by a node. However, because a network is not a machine, it cannot function as a router. One of the routers in the network takes this responsibility. It is assigned a dual purpose; it is a true router and a designated router. We can use the topology shown in Figure 11.24c to show the connections of a transient network.

Now each router has only one neighbor, the designated router (network). On the other hand, the designated router (the network) has five neighbors. We see that the number of neighbor announcements is reduced from 20 to 10. Still, the link is represented as a bidirectional edge between the nodes. However, while there is a metric from each node to the designated router, there is no metric from the designated router to any other node. The reason is that the designated router represents the network. We can only assign a cost to a packet that is passing through the network. We cannot charge for this twice. When a packet enters a network, we assign a cost; when a packet leaves the network to go to the router, there is no charge.

Stub Link

A **stub link** is a network that is connected to only one router. The data packets enter the network through this single router and leave the network through this same router. This is a special case of the transient network. We can show this situation using the router as a node and using the designated router for the network. However, the link is only one-directional, from the router to the network (see Figure 11.25).

Figure 11.25 *Stub link*



a. Stub network b. Representation

Virtual Link

When the link between two routers is broken, the administration may create a **virtual link** between them using a longer path that probably goes through several routers.

Graphical Representation

Let us now examine how an AS can be represented graphically. Figure 11.26 shows a small AS with seven networks and six routers. Two of the networks are point-to-point networks. We use symbols such as N1 and N2 for transient and stub networks. There is no need to assign an identity to a point-to-point network. The figure also shows the graphical representation of the AS as seen by OSPF.

We have used color nodes for the routers and shaded nodes for the networks (represented by designated routers). However, OSPF sees both as nodes. Note that we have three stub networks.

OSPF Packets

OSPF uses five different types of packets: *hello, database description, link state request, link state update,* and *link state acknowledgment* (see Figure 11.27). The most important one is the link state update that itself has five different kinds.

Figure 11.26 *Example of an AS and its graphical representation in OSPF*

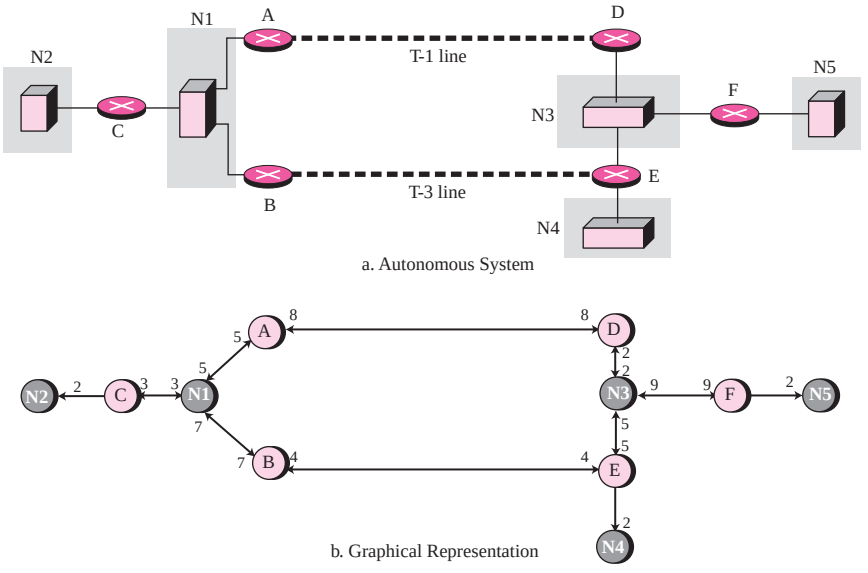
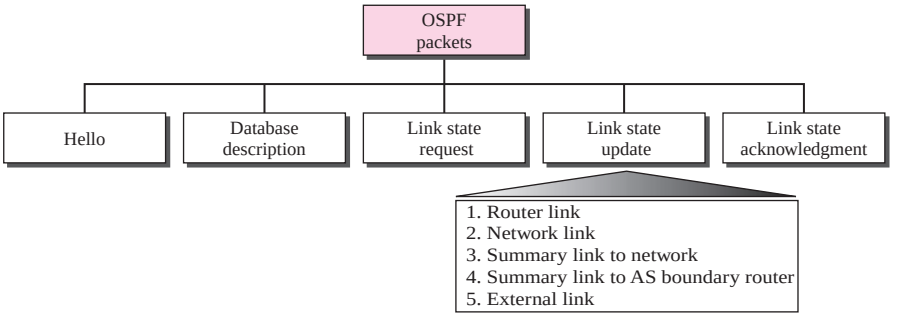


Figure 11.27 *Types of OSPF packets*

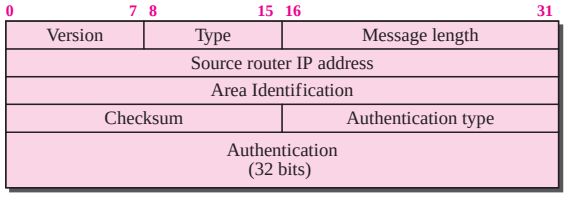


Common Header

All OSPF packets have the same common header (see Figure 11.28). Before studying the different types of packets, let us talk about this common header.

- **Version.** This 8-bit field defines the version of the OSPF protocol. It is currently version 2.
- **Type.** This 8-bit field defines the type of the packet. As we said before, we have five types, with values 1 to 5 defining the types.
- **Message length.** This 16-bit field defines the length of the total message including the header.

Figure 11.28 *OSPF common header*

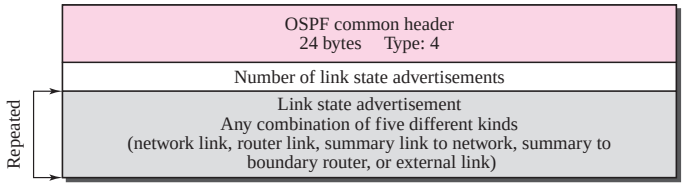


- **Source router IP address.** This 32-bit field defines the IP address of the router that sends the packet.
- **Area identification.** This 32-bit field defines the area within which the routing takes place.
- **Checksum.** This field is used for error detection on the entire packet excluding the authentication type and authentication data field.
- **Authentication type.** This 16-bit field defines the authentication protocol used in this area. At this time, two types of authentication are defined: 0 for none and 1 for password.
- **Authentication.** This 64-bit field is the actual value of the authentication data. In the future, when more authentication types are defined, this field will contain the result of the authentication calculation. For now, if the authentication type is 0, this field is filled with 0s. If the type is 1, this field carries an eight-character password.

Link State Update Packet

We first discuss the **link state update packet,** the heart of the OSPF operation. It is used by a router to advertise the states of its links. The general format of the link state update packet is shown in Figure 11.29.

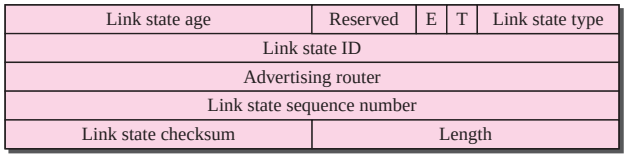
Figure 11.29 *Link state update packet*



Each update packet may contain several different LSAs. All five kinds have the same general header. This general header is shown in Figure 11.30 and described below:

- **Link state age.** This field indicates the number of seconds elapsed since this message was first generated. Recall that this type of message goes from router to router (flooding). When a router creates the message, the value of this field is 0. When

Figure 11.30 *LSA general header*



each successive router forwards this message, it estimates the transit time and adds it to the cumulative value of this field.

- **E flag.** If this 1-bit flag is set to 1, it means that the area is a stub area. A stub area is an area that is connected to the backbone area by only one path.
- **T flag.** If this 1-bit flag is set to 1, it means that the router can handle multiple types of service.
- **Link state type.** This field defines the LSA type. As we discussed before, there are five different advertisement types: router link (1), network link (2), summary link to network (3), summary link to AS boundary router (4), and external link (5).
- **Link state ID.** The value of this field depends on the type of link. For type 1 (router link), it is the IP address of the router. For type 2 (network link), it is the IP address of the designated router. For type 3 (summary link to network), it is the address of the network. For type 4 (summary link to AS boundary router), it is the IP address of the AS boundary router. For type 5 (external link), it is the address of the external network.
- **Advertising router.** This is the IP address of the router advertising this message.
- **Link state sequence number.** This is a sequence number assigned to each link state update message.
- **Link state checksum.** This is not the usual checksum. Instead, the value of this field is calculated using *Fletcher's checksum* (see Appendix C), which is based on the whole packet except for the age field.
- **Length.** This defines the length of the whole packet in bytes.

Router Link LSA

A router link defines the links of a true router. A true router uses this advertisement to announce information about all of its links and what is at the other side of the link (neighbors). See Figure 11.31 for a depiction of a router link.

The router link LSA advertises all of the links of a router (true router). The format of the router link packet is shown in Figure 11.32.

The fields of the router link LSA are as follows:

- **Link ID.** The value of this field depends on the type of link. Table 11.5 shows the different link identifications based on link type.
- **Link data.** This field gives additional information about the link. Again, the value depends on the type of the link (see Table 11.5).

Figure 11.31 *Router link*

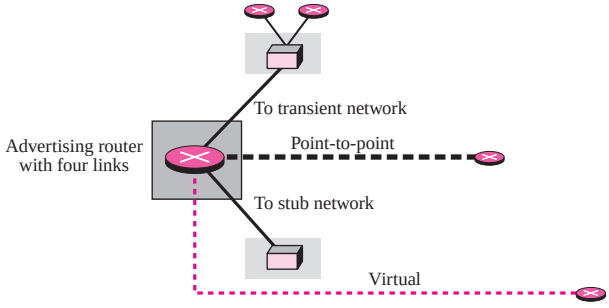


Figure 11.32 *Router link LSA*

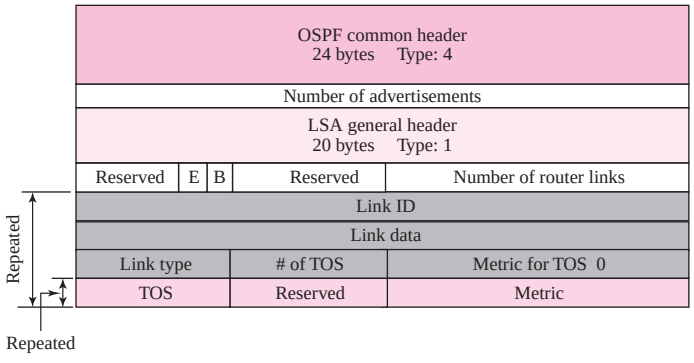


Table 11.5 *Link Types, Link Identification, and Link Data*

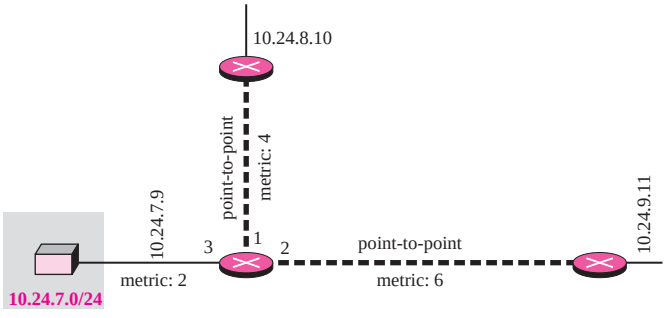
Link Type	*Link Identification*	*Link Data*
Type 1: Point-to-point	Address of neighbor router	Interface number
Type 2: Transient	Address of designated router	Router address
Type 3: Stub	Network address	Network mask
Type 4: Virtual	Address of neighbor router	Router address

- **Link type.** Four different types of links are defined based on the type of network to which the router is connected (see Table 11.5).
- **Number of types of service (TOS).** This field defines the number of types of services announced for each link.
- **Metric for TOS 0.** This field defines the metric for the default type of service (TOS 0).
- **TOS.** This field defines the type of service.
- **Metric.** This field defines the metric for the corresponding TOS.

Example 11.7

Give the router link LSA sent by router 10.24.7.9 in Figure 11.33.

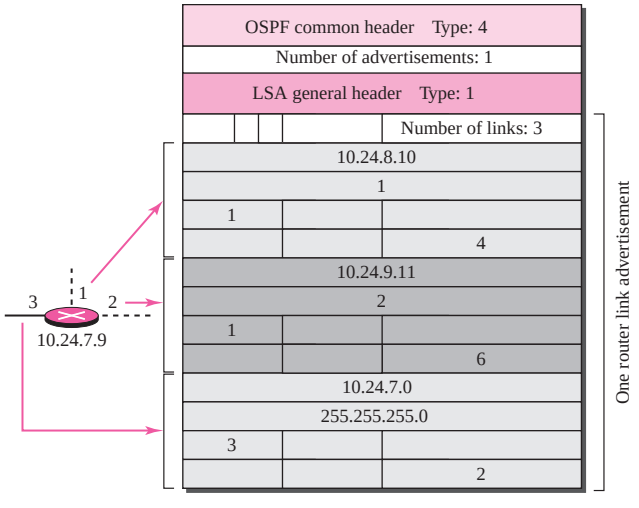
Figure 11.33 *Example 11.7*



Solution

This router has three links: two of type 1 (point-to-point) and one of type 3 (stub network). Figure 11.34 shows the router link LSA.

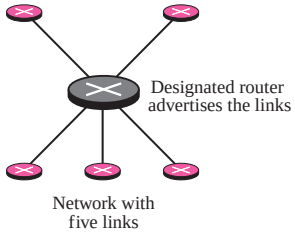
Figure 11.34 *Solution to Example 11.8*



Network Link LSA

A network link defines the links of a network. A designated router, on behalf of the transient network, distributes this type of LSP packet. The packet announces the existence of all of the routers connected to the network (see Figure 11.35). The format of the

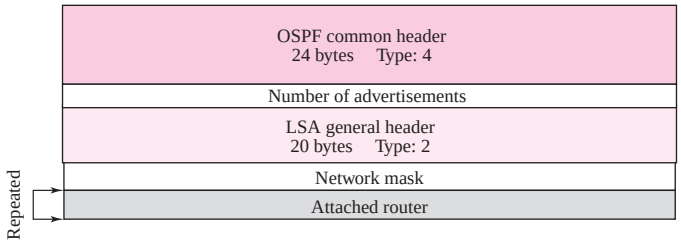
Figure 11.35 *Network link*



network link advertisement is shown in Figure 11.36. The fields of the network link LSA are as follows:

- **Network mask.** This field defines the network mask.
- **Attached router.** This repeated field defines the IP addresses of all attached routers.

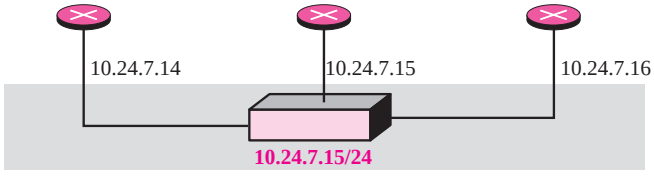
Figure 11.36 *Network link advertisement format*



Example 11.8

Give the network link LSA in Figure 11.37.

Figure 11.37 *Example 4*



Solution

The network for which the network link advertises has three routers attached. The LSA shows the mask and the router addresses. Figure 11.38 shows the network link LSA.

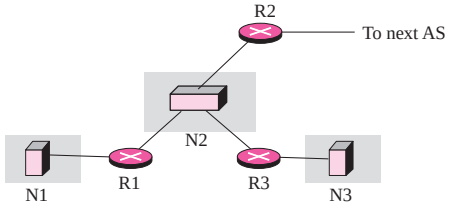
Figure 11.38 *Solution to Example 4*

OSPF common header Type: 4
Number of advertisements: 1
LSA general header Type: 2
255.255.255.0
10.24.7.14
10.24.7.15
10.24.7.16

Example 11.9

In Figure 11.39, which router(s) sends out router link LSAs?

Figure 11.39 *Example 11.9 and Example 11.10*



Solution

All routers advertise router link LSAs.

- **a.** R1 has two links, N1 and N2.
- **b.** R2 has one link, N1.
- **c.** R3 has two links, N2 and N3.

Example 11.10

In Figure 11.39, which router(s) sends out the network link LSAs?

Solution

All three networks must advertise network links:

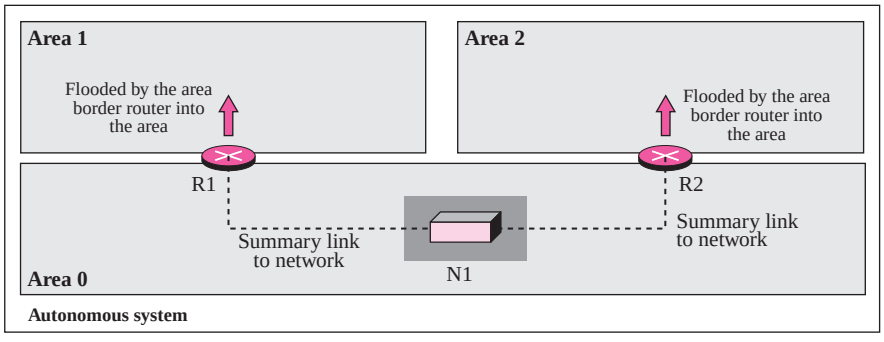
- **a.** Advertisement for N1 is done by R1 because it is the only attached router and therefore the designated router.
- **b.** Advertisement for N2 can be done by either R1, R2, or R3, depending on which one is chosen as the designated router.
- **c.** Advertisement for N3 is done by R3 because it is the only attached router and therefore the designated router.

Summary Link to Network LSA

Router link and network link advertisements flood the area with information about the router links and network links inside an area. But a router must also know about the networks outside its area; the area border routers can provide this information. An area border router is active in more than one area. It receives router link and network link

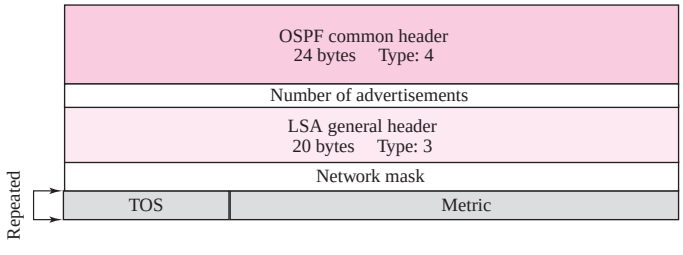
advertisements, and, as we will see, creates a routing table for each area. For example, in Figure 11.40, router R1 is an area border router. It has two routing tables, one for area 1 and one for area 0. R1 floods area 1 with information about how to reach a network located in area 0. In the same way, router R2 floods area 2 with information about how to reach the same network in area 0.

Figure 11.40 *Summary link to network*



The summary link to network LSA is used by the area border router to announce the existence of other networks outside the area. The summary link to network advertisement is very simple. It consists of the network mask and the metric for each type of service. Note that each advertisement announces only one single network. If there is more than one network, a separate advertisement must be issued for each. The reader may ask why only the mask of the network is advertised. What about the network address itself? The IP address of the advertising router is announced in the header of the link state advertisement. From this information and the mask, one can deduce the network address. The format of this advertisement is shown in Figure 11.41. The fields of the summary link to network LSA are as follows:

Figure 11.41 *Summary link to network LSA*



- **Network mask.** This field defines the network mask.
- **TOS.** This field defines the type of service.
- **Metric.** This field defines the metric for the type of service defined in the TOS field.

Summary Link to AS Boundary Router LSA

The previous advertisement lets every router know the cost to reach all of the networks inside the autonomous system. But what about a network outside the autonomous system? If a router inside an area wants to send a packet outside the autonomous system, it should first know the route to an autonomous boundary router; the summary link to AS boundary router provides this information. The area border routers flood their areas with this information (see Figure 11.42). This packet is used to announce the route to an AS boundary router. Its format is the same as the previous summary link. The packet just defines the network to which the AS boundary router is attached. If a message can reach the network, it can be picked up by the AS boundary router. The format of the packet is shown in Figure 11.43. The fields are the same as the fields in the summary link to network advertisement message.

Figure 11.42 *Summary link to AS boundary router*

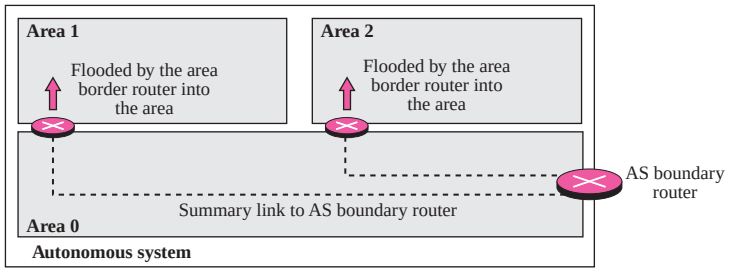
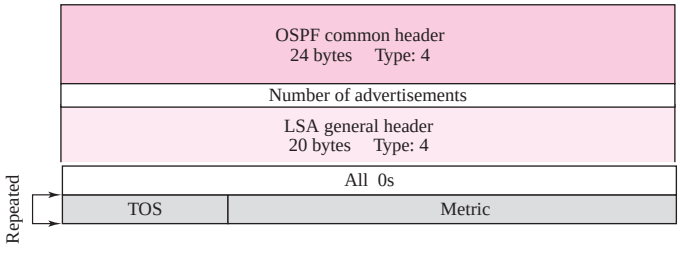


Figure 11.43 *Summary link to AS boundary router LSA*



External Link LSA

Although the previous advertisement lets each router know the route to an AS boundary router, this information is not enough. A router inside an autonomous system wants to know which networks are available outside the autonomous system; the external link advertisement provides this information. The AS boundary router floods the autonomous system with the cost of each network outside the autonomous system

using a routing table created by an interdomain routing protocol. Each advertisement announces one single network. If there is more than one network, separate announcements are made. Figure 11.44 depicts an external link. This is used to announce all the networks outside the AS. The format of the LSA is similar to the summary link to the AS boundary router LSA, with the addition of two fields. The AS boundary router may define a forwarding router that can provide a better route to the destination. The packet also can include an external route tag, used by other protocols, but not by OSPF. The format of the packet is shown in Figure 11.45.

Figure 11.44 *External link*

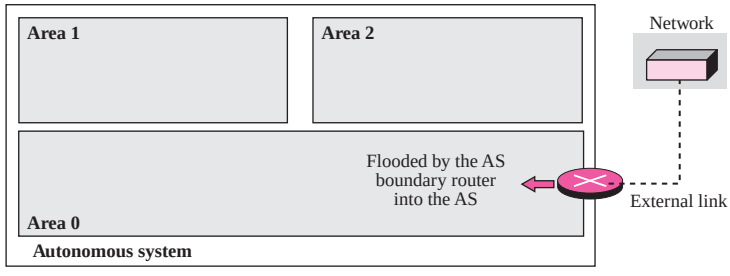
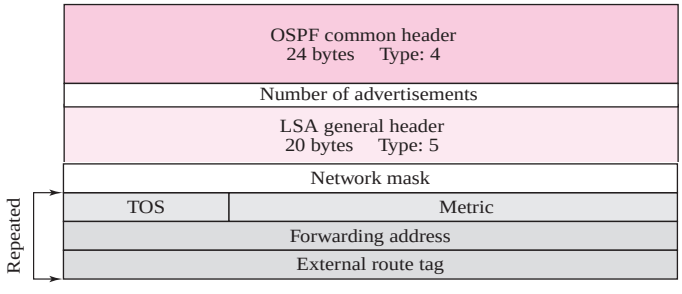


Figure 11.45 *External link LSA*



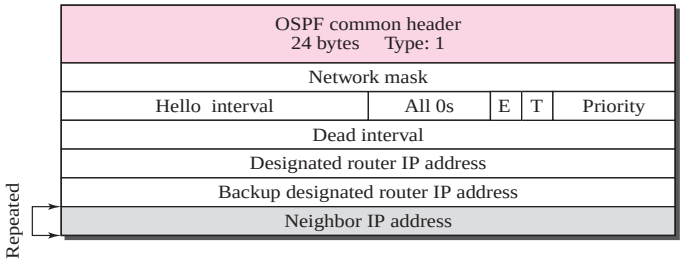
Other Packets

Now we discuss four other packet types (See Figure 11.27). They are not used as LSAs, but are essential to the operation of OSPF.

Hello Message

OSPF uses the **hello message** to create neighborhood relationships and to test the reachability of neighbors. This is the first step in link state routing. Before a router can flood all of the other routers with information about its neighbors, it must first greet its neighbors. It must know if they are alive, and it must know if they are reachable (see Figure 11.46).

Figure 11.46 *Hello packet*



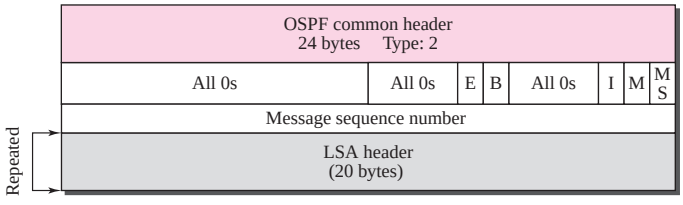
- **Network mask.** This 32-bit field defines the network mask of the network over which the hello message is sent.
- **Hello interval.** This 16-bit field defines the number of seconds between hello messages.
- **E flag.** This is a 1-bit flag. When it is set, it means that the area is a stub area.
- **T flag.** This is a 1-bit flag. When it is set, it means that the router supports multiple metrics.
- **Priority.** This field defines the priority of the router. The priority determines the selection of the designated router. After all neighbors declare their priorities, the router with the highest priority is chosen as the designated router. The one with the second highest priority is chosen as the backup designated router. If the value of this field is 0, it means that the router never wants to be a designated or a backup designated router.
- **Dead interval.** This 32-bit field defines the number of seconds that must pass before a router assumes that a neighbor is dead.
- **Designated router IP address.** This 32-bit field is the IP address of the designated router for the network over which the message is sent.
- **Backup designated router IP address.** This 32-bit field is the IP address of the backup designated router for the network over which the message is sent.
- **Neighbor IP address.** This is a repeated 32-bit field that defines the routers that have agreed to be the neighbors of the sending router. In other words, it is a current list of all the neighbors from which the sending router has received the hello message.

Database Description Message

When a router is connected to the system for the first time or after a failure, it needs the complete link state database immediately. It cannot wait for all link state update packets to come from every other router before making its own database and calculating its routing table. Therefore, after a router is connected to the system, it sends hello packets to greet its neighbors. If this is the first time that the neighbors hear from the router, they send a database description message. The database description packet does not

contain complete database information; it only gives an outline, the title of each line in the database. The newly connected router examines the outline and finds out which lines of information it does not have. It then sends one or more link state request packets to get full information about that particular link. When two routers want to exchange database description packets, one of them takes the role of master and the other the role of slave. Because the message can be very long, the contents of the database can be divided into several messages. The format of the database description packet is shown in Figure 11.47. The fields are as follows:

Figure 11.47 *Database description packet*

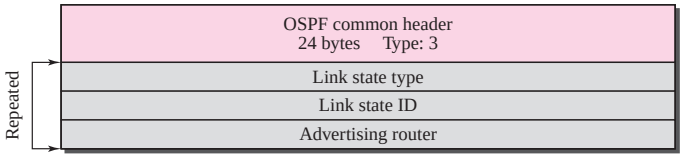


- **E flag.** This 1-bit flag is set to 1 if the advertising router is an autonomous boundary router (*E* stands for external).
- **B flag.** This 1-bit flag is set to 1 if the advertising router is an area border router.
- **I flag.** This 1-bit field, the *initialization* flag, is set to 1 if the message is the first message.
- **M flag.** This 1-bit field, the *more* flag, is set to 1 if this is not the last message.
- **M/S flag.** This 1-bit field, the *master/slave* bit, indicates the origin of the packet: master (M/S = 1) or slave (M/S = 0).
- **Message sequence number.** This 32-bit field contains the sequence number of the message. It is used to match a request with the response.
- **LSA header.** This 20-byte field is used in each LSA. The format of this header is discussed in the link state update message section. This header gives the outline of each link, without details. It is repeated for each link in the link state database.

Link State Request Packet

The format of the **link state request packet** is shown in Figure 11.48. This is a packet that is sent by a router that needs information about a specific route or routes. It is answered with a link state update packet. It can be used by a newly connected router to request more information about some routes after receiving the database description packet. The three fields here are part of the LSA header, which has already been discussed. Each set of the three fields is a request for one single LSA. The set is repeated if more than one advertisement is desired.

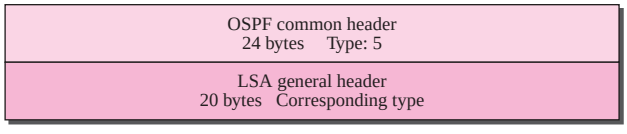
Figure 11.48 *Link state request packet*



Link State Acknowledgment Packet

OSPF makes routing more reliable by forcing every router to acknowledge the receipt of every link state update packet. The format of the **link state acknowledgment packet** is shown in Figure 11.49. It has the common OSPF header and the general LSA header. These two sections are sufficient to acknowledge a packet.

Figure 11.49 *Link state acknowledgment packet*



Encapsulation

OSPF packets are encapsulated in IP datagrams. They contain the acknowledgment mechanism for flow and error control. They do not need a transport layer protocol to provide these services.

OSPF packets are encapsulated in IP datagrams.

11.7 PATH VECTOR ROUTING

Distance vector and link state routing are both *interior* routing protocols. They can be used inside an autonomous system as intra-domain or intra-AS (as sometimes are called), but not between autonomous systems. Both of these routing protocols become intractable when the domain of operation becomes large. Distance vector routing is subject to instability if there is more than a few hops in the domain of operation. Link state routing needs a huge amount of resources to calculate routing tables. It also creates heavy traffic because of flooding. There is a need for a third routing protocol which we call **path vector routing.**

Path vector routing is exterior routing protocol proved to be useful for inter-domain or inter-AS routing as it is sometimes called. In distance vector routing, a

router has a list of networks that can be reached in the same AS with the corresponding cost (number of hops). In path vector routing, a router has a list of networks that can be reached with the path (list of ASs to pass) to reach each one. In other words, the domain of operation of the distance vector routing is a single AS; the domain of operation of the path vector routing is the whole Internet. The distance vector routing tells us the distance to each network; the path vector routing tells us the path.

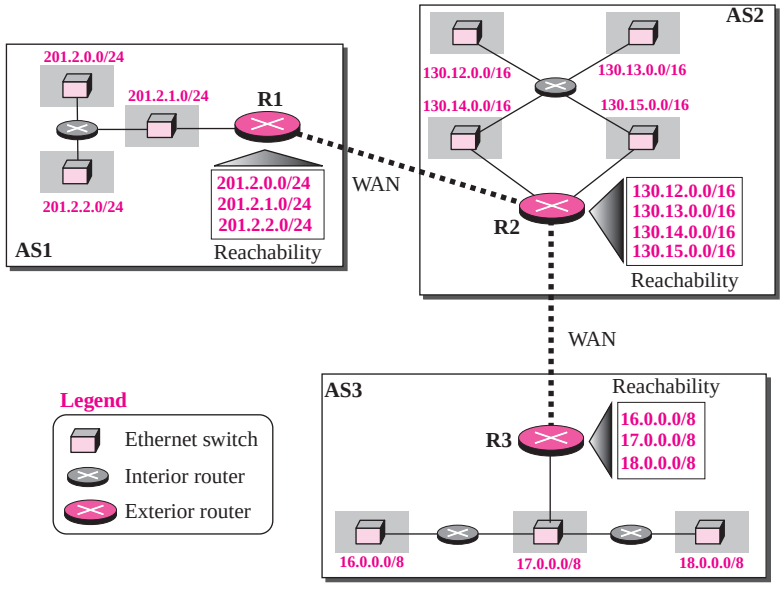
Example 11.11

The difference between the distance vector routing and path vector routing can be compared to the difference between a national map and an international map. A national map can tell us the road to each city and the distance to be travelled if we choose a particular route; an international map can tell us which cities exist in each country and which countries should be passed before reaching that city.

Reachability

To be able to provide information to other ASs, each AS must have at least one path vector routing that collects *reachability* information about each network in that AS. The information collected in this case only means which network, identified by its network address (CIDR prefix), exists (can be reached in this AS). In other words, the AS needs to have a list of existing networks in its territory. Figure 11.50 shows three ASs. Each distance vector (exterior router) has created a list which shows which network is reachable in that AS.

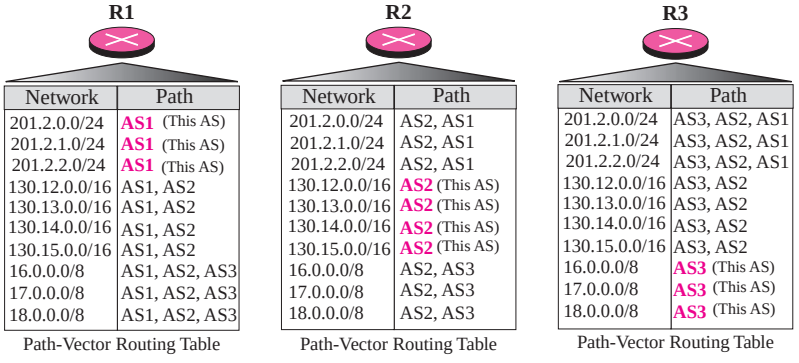
Figure 11.50 *Reachability*



Routing Tables

A path vector routing table for each router can be created if ASs share their reachability list with each other. In Figure 11.50, router R1 in AS1 can send its reachability list to router R2. Router R2, after combining its reachability list, can send the result to both R1 and R3. Router R3 can send its reachability list to R2, which in turn improves its routing table, and so on. Figure 11.51 shows the routing table for each router after all three routers have updated their routing table. Router R1 knows that if a packet arrives for the network 201.2.2.0/24, this network is in AS1 (at home), but if a packet arrives for the network 130.14.0.0/16, the packet should travel from AS1 to AS2 to reach its destination network. On the other hand, if router R2 receives a packet destined for the network 22.0.0.0.8, the router knows that it should travel from AS2 to AS3 to reach its destination. We can compare these routing tables with the distance vector routing table to see the differences.

Figure 11.51 *Stabilized tables for three autonomous systems*



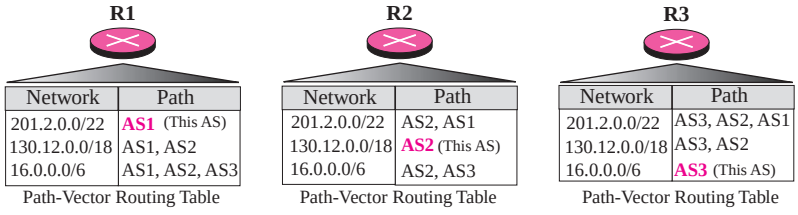
Loop Prevention

The instability of distance vector routing and the creation of loops can be avoided in path vector routing. When a router receives a reachability information, it checks to see if its autonomous system is in the path list to any destination. If it is, looping is involved and that network-path pair is discarded.

Aggregation

The path vector routing protocols normally support CIDR notation and the aggregation of addresses (if possible). This helps to make the path vector routing table simpler and exchange between routers faster. For example, the path vector routing table of Figure 11.51 can be aggregated to create shorter routing tables (Figure 11.52). Note that a range may also include a block that may not be in the corresponding AS. For example, the range 201.2.0.0/22 also includes the range 201.2.0.3/24, which is not the network address of any network in AS1. However, if this network exists in some other ASs, it eventually becomes part of the routing table. Based on the longest prefix principle we discussed in Chapter 6, this network address is above 201.2.0.0/22 and is searched first, which results in correct routing.

Figure 11.52 *Routing table after aggregation*



Policy Routing

Policy routing can be easily implemented through path vector routing. When a router receives a message, it can check the path. If one of the autonomous systems listed in the path is against its policy, it can ignore that path and that destination. It does not update its routing table with this path, and it does not send this message to its neighbors.

11.8 BGP

Border Gateway Protocol (BGP) is an interdomain routing protocol using path vector routing. It first appeared in 1989 and has gone through four versions.

Types of Autonomous Systems

As we said before, the Internet is divided into hierarchical domains called autonomous systems (ASs). For example, a large corporation that manages its own network and has full control over it is an autonomous system. A local ISP that provides services to local customers is an autonomous system. Note that a single organization may choose to have multiple ASs because of geographical spread, different providers (ISPs), or even some local obstacles.

We can divide autonomous systems into three categories: stub, multihomed, and transit.

Stub AS

A stub AS has only one connection to another AS. The interdomain data traffic in a stub AS can be either created or terminated in the AS. The hosts in the AS can send data traffic to other ASs. The hosts in the AS can receive data coming from hosts in other ASs. Data traffic, however, cannot pass through a stub AS. A stub AS is either a source or a sink. A good example of a stub AS is a small corporation or a small local ISP.

Multihomed AS

A multihomed AS has more than one connection to other ASs, but it is still only a source or sink for data traffic. It can receive data traffic from more than one AS. It can send data traffic to more than one AS, but there is no transient traffic. It does not allow data coming from one AS and going to another AS to pass through. A good example of a multihomed AS is a large corporation that is connected to more than one regional or national AS that does not allow transient traffic.

Transit AS

A transit AS is a multihomed AS that also allows transient traffic. Good examples of transit ASs are national and international ISPs (Internet backbones).

CIDR

BGP uses classless interdomain routing addresses. In other words, BGP uses a prefix, as discussed in Chapter 5, to define a destination address. The address and the number of bits (prefix length) are used in updating messages.

Path Attributes

In our previous example, we discussed a path for a destination network. The path was presented as a list of autonomous systems, but is, in fact, a list of attributes. Each attribute gives some information about the path. The list of attributes helps the receiving router make a better decision when applying its policy.

Attributes are divided into two broad categories: well-known and optional. A **well-known attribute** is one that every BGP router must recognize. An **optional attribute** is one that needs not be recognized by every router.

Well-known attributes are themselves divided into two categories: mandatory and discretionary. A *well-known mandatory attribute* is one that must appear in the description of a route. A *well-known discretionary attribute* is one that must be recognized by each router, but is not required to be included in every update message. One well-known mandatory attribute is ORIGIN. This defines the source of the routing information (RIP, OSPF, and so on). Another well-known mandatory attribute is AS_PATH. This defines the list of autonomous systems through which the destination can be reached. Still another well-known mandatory attribute is NEXT-HOP, which defines the next router to which the data packet should be sent.

The optional attributes can also be subdivided into two categories: transitive and nontransitive. An *optional transitive attribute* is one that must be passed to the next router by the router that has not implemented this attribute. An *optional nontransitive attribute* is one that must be discarded if the receiving router has not implemented it.

BGP Sessions

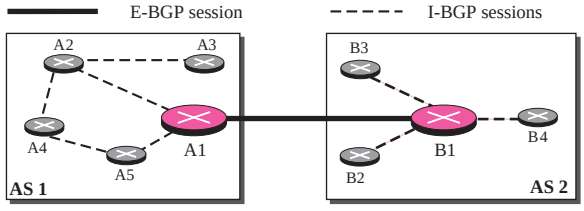
The exchange of routing information between two routers using BGP takes place in a session. A session is a connection that is established between two BGP routers only for the sake of exchanging routing information. To create a reliable environment, BGP uses the services of TCP. In other words, a session at the BGP level, as an application program, is a connection at the TCP level. However, there is a subtle difference between a connection in TCP made for BGP and other application programs. When a TCP connection is created for BGP, it can last for a long time, until something unusual happens. For this reason, BGP sessions are sometimes referred to as *semipermanent connections*.

External and Internal BGP

If we want to be precise, BGP can have two types of sessions: external BGP (E-BGP) and internal BGP (I-BGP) sessions. The E-BGP session is used to exchange information

between two speaker nodes belonging to two different autonomous systems. The I-BGP session, on the other hand, is used to exchange routing information between two routers inside an autonomous system. Figure 11.53 shows the idea.

Figure 11.53 *Internal and external BGP sessions*

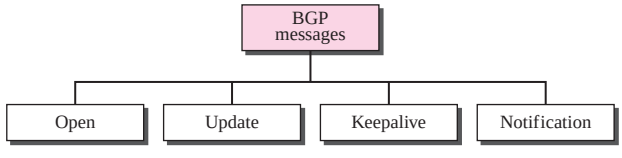


The session established between AS1 and AS2 is an E-BGP session. The two speaker routers exchange information they know about networks in the Internet. However, these two routers need to collect information from other routers in the autonomous systems. This is done using I-BGP sessions.

Types of Packets

BGP uses four different types of messages: **open, update, keepalive,** and **notification** (see Figure 11.54).

Figure 11.54 *Types of BGP messages*

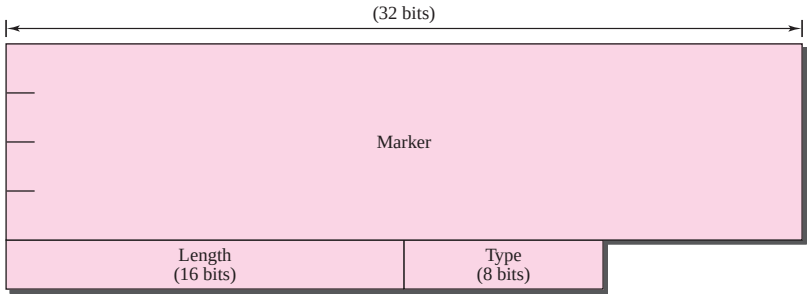


Packet Format

All BGP packets share the same common header. Before studying the different types of packets, let us talk about this common header (see Figure 11.55). The fields of this header are as follows:

- **Marker.** The 16-byte marker field is reserved for authentication.
- **Length.** This 2-byte field defines the length of the total message including the header.
- **Type.** This 1-byte field defines the type of the packet. As we said before, we have four types, and the values 1 to 4 define those types.

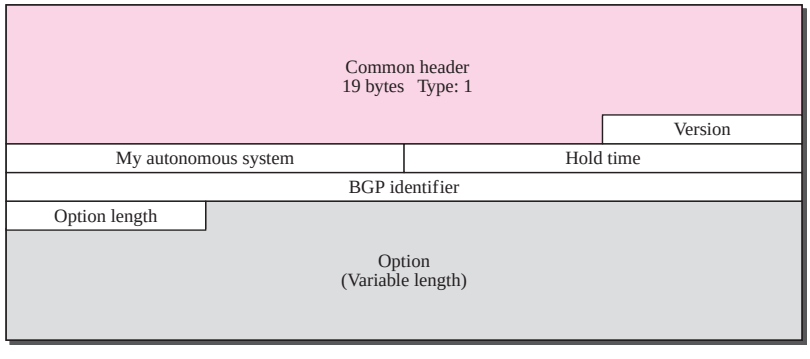
Figure 11.55 *BGP packet header*



Open Message

To create a neighborhood relationship, a router running BGP opens a TCP connection with a neighbor and sends an **open message.** If the neighbor accepts the neighborhood relationship, it responds with a **keepalive message,** which means that a relationship has been established between the two routers. See Figure 11.56 for a depiction of the open message format.

Figure 11.56 *Open message*



The fields of the open message are as follows:

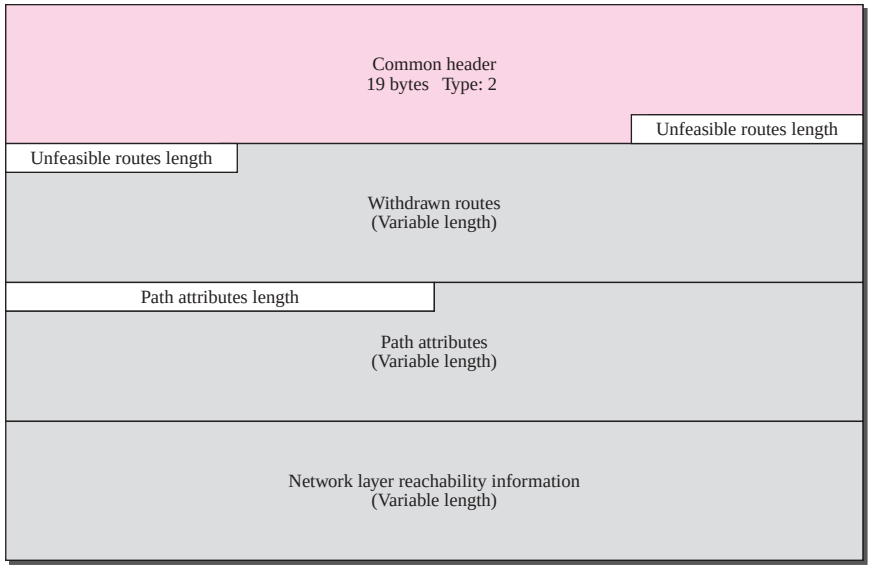
- **Version.** This 1-byte field defines the version of BGP. The current version is 4.
- **My autonomous system.** This 2-byte field defines the autonomous system number.
- **Hold time.** This 2-byte field defines the maximum number of seconds that can elapse until one of the parties receives a keepalive or update message from the other. If a router does not receive one of these messages during the hold time period, it considers the other party dead.

- **BGP identifier.** This 4-byte field defines the router that sends the open message. The router usually uses one of its IP addresses (because it is unique) for this purpose.
- **Option length.** The open message may contain some option parameters. In this case, this 1-byte field defines the length of the total option parameters. If there are no option parameters, the value of this field is zero.
- **Option parameters.** If the value of the option parameter length is not zero, it means that there are some option parameters. Each option parameter itself has two subfields: the length of the parameter and the parameter value. The only option parameter defined so far is authentication.

Update Message

The update message is the heart of the BGP protocol. It is used by a router to withdraw destinations that have been advertised previously, announce a route to a new destination, or both. Note that BGP can withdraw several destinations that were advertised before, but it can only advertise one new destination in a single update message. The format of the update message is shown in Figure 11.57.

Figure 11.57 *Update message*



The update message fields are listed below:

- **Unfeasible routes length.** This 2-byte field defines the length of the next field.
- **Withdrawn routes.** This field lists all the routes that must be deleted from the previously advertised list.

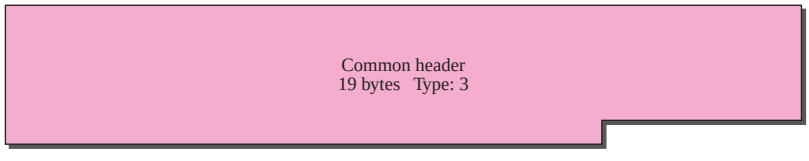
- **Path attributes length.** This 2-byte field defines the length of the next field.
- **Path attributes.** This field defines the attributes of the path (route) to the network whose reachability is being announced in this message.
- **Network layer reachability information (NLRI).** This field defines the network that is actually advertised by this message. It has a length field and an IP address prefix. The length defines the number of bits in the prefix. The prefix defines the common part of the network address. For example, if the network is 153.18.7.0/24, the length of the prefix is 24 and the prefix is 153.18.7. BGP4 supports classless addressing and CIDR.

BGP supports classless addressing and CIDR.

Keepalive Message

The routers (called *peers* in BGP parlance) running the BGP protocols exchange keepalive messages regularly (before their hold time expires) to tell each other that they are alive. The keepalive message consists of only the common header shown in Figure 11.58.

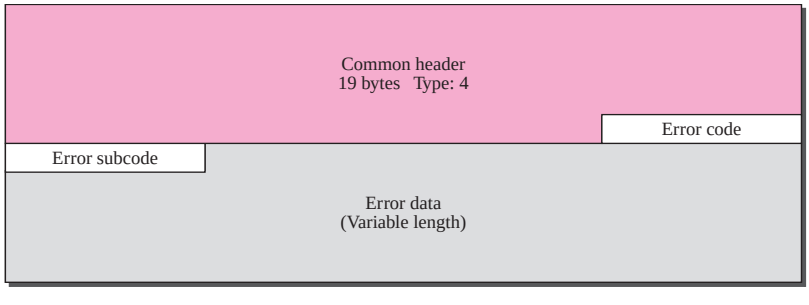
Figure 11.58 *Keepalive message*



Notification Message

A notification message is sent by a router whenever an error condition is detected or a router wants to close the connection. The format of the message is shown in Figure 11.59.The fields making up the notification message follow:

Figure 11.59 *Notification message*



- **Error code.** This 1-byte field defines the category of the error. See Table 11.6.
- **Error subcode.** This 1-byte field further defines the type of error in each category.
- **Error data.** This field can be used to give more diagnostic information about the error.

Table 11.6 *Error Codes*

Error Code	*Error Code Description*	*Error Subcode Description*
1	Message header error	Three different subcodes are defined for this type of error: synchronization problem (1), bad message length (2), and bad message type (3).
2	Open message error	Six different subcodes are defined for this type of error: unsupported version number (1), bad peer AS (2), bad BGP identifier (3), unsupported optional parameter (4), authentication failure (5), and unacceptable hold time (6).
3	Update message error	Eleven different subcodes are defined for this type of error: malformed attribute list (1), unrecognized well-known attribute (2), missing well-known attribute (3), attribute flag error (4), attribute length error (5), invalid origin attribute (6), AS routing loop (7), invalid next hop attribute (8), optional attribute error (9), invalid network field (10), malformed AS_PATH (11).
4	Hold timer expired	No subcode defined.
5	Finite state machine error	This defines the procedural error. No subcode defined.
6	Cease	No subcode defined.

Encapsulation

BGP messages are encapsulated in TCP segments using the well-known port 179. This means that there is no need for error control and flow control. When a TCP connection is opened, the exchange of update, keepalive, and notification messages is continued until a notification message of type cease is sent.

BGP uses the services of TCP on port 179.

11.9 FURTHER READING

For more details about subjects discussed in this chapter, we recommend the following books and RFCs. The items enclosed in brackets refer to the reference list at the end of the book.

Books

Several books give more information about unicast routing. In particular, we recommend [Com 06], [Pet & Dav 03], [Kur & Ros 08], [Per 00], [Gar & Vid 04], [Tan 03], [Sta 04], and [Moy 98].

RFCs

Several RFCs are related to the protocols we discussed in this chapter. RIP is discussed in RFC1058 and RFC 2453. OSPF is discussed in RFC 1583 and RFC 2328. BGP is discussed in RFC 1654, RFC 1771, RFC 1773, RFC 1997, RFC 2439, RFC 2918, and RFC 3392.

11.10 KEY TERMS

area
area border router
autonomous system (AS)
autonomous system boundary router
backbone router
Bellman-Ford algorithm
Border Gateway Protocol (BGP)
cost
count to infinity
Dijkstra algorithm
distance vector routing
expiration timer
flooding
garbage collection timer
hello interval
hello message
hop count
inter-domain routing
intra-domain routing
keepalive message
link state request packet
link state routing
link state update packet
metric
notification message
open message
Open Shortest Path First (OSPF)
optional attribute
path vector routing
periodic timer
point-to-point link
poison reverse
Routing Information Protocol (RIP)
split horizon
stub link
transient link
update message
virtual link
well-known attribute

11.11 SUMMARY

- A metric is the cost assigned for passage of a packet through a network. A router consults its routing table to determine the best path for a packet.
- An autonomous system (AS) is a group of networks and routers under the authority of a single administration. RIP and OSPF are popular intradomain or intra-AS routing protocols (also called interior routing protocols) used to update routing tables in an AS. RIP is based on distance vector routing, in which each router shares, at regular intervals, its knowledge about the entire AS with its neighbors. OSPF divides an AS into areas, defined as collections of networks, hosts, and routers. OSPF is based on link state routing, in which each router sends the state of its neighborhood to every other router in the area.
- BGP is an interdomain or inter-AS routing protocol (also called exterior routing protocol) used to update routing tables. BGP is based on a routing protocol called path vector routing. In this protocol, the ASs through which a packet must pass are

explicitly listed. Path vector routing does not have the instability nor looping problems of distance vector routing. There are four types of BGP messages: open, update, keepalive, and notification.

11.12 PRACTICE SET

Exercises

1. In RIP, why is the expiration timer value six times that of the periodic timer value?
2. How does the hop count limit alleviate RIP's problems?
3. Contrast and compare distance vector routing with link state routing.
4. Why do OSPF messages propagate faster than RIP messages?
5. What is the size of a RIP message that advertises only one network? What is the size of a RIP message that advertises *N* packets? Devise a formula that shows the relationship between the number of networks advertised and the size of a RIP message.
6. A router running RIP has a routing table with 20 entries. How many periodic timers are needed to handle this table? How many expiration timers are needed to handle this table? How many garbage collection timers are needed to handle this table if five routes are invalid?
7. A router using RIP has the routing table shown in Table 11.7.

Table 11.7 *Routing Table for Exercise 7*

Destination	*Cost*	*Next Router*
Net1	4	B
Net2	2	C
Net3	1	F
Net4	5	G

   Show the RIP response message sent by this router.
8. The router in Exercise 7 receives a RIP message with four records from router C as shown below:

```
(Net1, 2), (Net2, 1), (Net3, 3), (Net4, 7)
```

   Show the updated routing table.
9. How many bytes are empty in a RIP message that advertises *N* networks?
10. Using Figure 11.26, do the following:
    a. Show the link state update/router link advertisement for router A.
    b. Show the link state update/router link advertisement for router D.
    c. Show the link state update/router link advertisement for router E.
    d. Show the link state update/network link advertisement for network N2.
    e. Show the link state update/network link advertisement for network N4.
    f. Show the link state update/network link advertisement for network N5.

11. In Figure 11.26,

- **a.** Assume that the designated router for network N1 is router A. Show the link state update/network link advertisement for this network.
- **b.** Assume that the designated router for network N3 is router D. Show the link state update/network link advertisement for this network.

12. Assign IP addresses to networks and routers in Figure 11.26. Now do the following:

- **a.** Show the OSPF hello message sent by router C.
- **b.** Show the OSPF database description message sent by router C.
- **c.** Show the OSPF link state request message sent by router C.

13. Show the autonomous system with the following specifications:

- **a.** There are eight networks (N1 to N8)
- **b.** There are eight routers (R1 to R8)
- **c.** N1, N2, N3, N4, N5, and N6 are Ethernet LANs
- **d.** N7 and N8 are point-to-point WANs
- **e.** R1 connects N1 and N2
- **f.** R2 connects N1 and N7
- **g.** R3 connects N2 and N8
- **h.** R4 connects N7 and N6
- **i.** R5 connects N6 and N3
- **j.** R6 connects N6 and N4
- **k.** R7 connects N6 and N5
- **l.** R8 connects N8 and N5

Now draw the graphical representation of the autonomous system of Exercise 29 as seen by OSPF. Which of the networks is a transient network? Which is a stub network?

14. In Figure 11.50,

- **a.** Show the BGP open message for router R1.
- **b.** Show the BGP update message for router R1.
- **c.** Show the BGP keepalive message for router R1.
- **d.** Show the BGP notification message for router R1.

15. In Figure 11.5, assume that the link between router A and router B fails (breaks). Show the changes in the routing table for routers in Figure 11.7.

16. In Figure 11.5, assume that the link between router C and router D fails (breaks). Show the changes in the routing table for routers in Figure 11.7.

17. Use the Bellman-Ford algorithm (Table 11.1) to find the shortest distance for all nodes in the graph of Figure 11.60.

18. Use the Dijkstra algorithm (Table 11.3) to find the shortest paths for all nodes in the graph of Figure 11.61.

19. Find the shortest path tree for node B in Figure 11.19.

Figure 11.60 *Exercise 17*

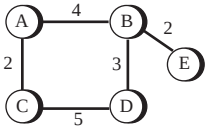
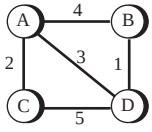


Figure 11.61 *Exercise 18*



20. Find the shortest path tree for node E in Figure 11.19.

21. Find the shortest path tree for node G in Figure 11.19.

Research Activities

22. Before BGP, there was a protocol called EGP. Find some information about this protocol. Find out why this protocol has not survived.

23. In UNIX, there are some programs under the general name *daemon*. Find the daemons that can handle routing protocols.

24. If you have access to a UNIX system, find some information about the *routed* program. How can this program help to trace the messages exchanged in RIP? Does *routed* support the other routing protocols we discussed in the chapter?

25. If you have access to a UNIX system, find some information about the *gated* program. Which routing protocols discussed in this chapter can be supported by *gated*?

26. There is a routing protocol called HELLO, which we did not discuss in this chapter. Find some information about this protocol.