UNIT III
ROUTING

Routing (RIP, OSPF, metrics) – Switch basics – Global Internet (Areas, BGP, IPv6), Multicast – addresses – multicast routing (DVMRP, PIM)

3.1. Routing

- A routing protocol is a combination of rules and procedures that let routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhood.
- Two tables are used here. They are *forwarding table* and *routing table*
  - ✓ The *forwarding table* is used when a packet is being forwarded and it means that a row in the forwarding table contains the mapping from a network prefix to an outgoing interface and some MAC information, such as the Ethernet address of the next hop.
  - ✓ The *routing table* is the table that is built up by the routing algorithms as a precursor to building the forwarding table. It generally contains mappings from network prefixes to next hops. It may also contain information about how this information was learned, so that the router will be able to decide when it should discard some information.
- For example, the forwarding table needs to be structured to optimize the process of looking up an address when forwarding a packet, while the routing table needs to be optimized for the purpose of calculating changes in topology.
- In many cases, the forwarding table may even be implemented in specialized hardware, whereas this is rarely if ever done for the routing table.
- The following table provides an example of a row from each sort of table.
  - ✓ In this case, the routing table tells us that network prefix 18/8 is to be reached by a next hop router with the IP address 171.69.245.10, while the forwarding table contains the information about exactly how to forward a packet to that next hop:
  - ✓ Send it out interface number 0 with a MAC address of 8:0:2b:e4:b:1:2. Note that the last piece of information is provided by the Address Resolution Protocol.
  - ✓ An *autonomous system (AS)* or routing domain is a group of networks and routers under the authority of a single administration. It shown in Figure 3.1.

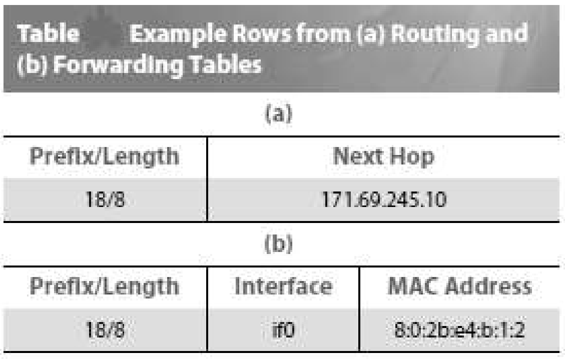


Table Example Rows from (a) Routing and (b) Forwarding Tables

(a)

Prefix/Length	Next Hop
18/8	171.69.245.10

(b)

Prefix/Length	Interface	MAC Address
18/8	if0	8:0:2b:e4:b:1:2

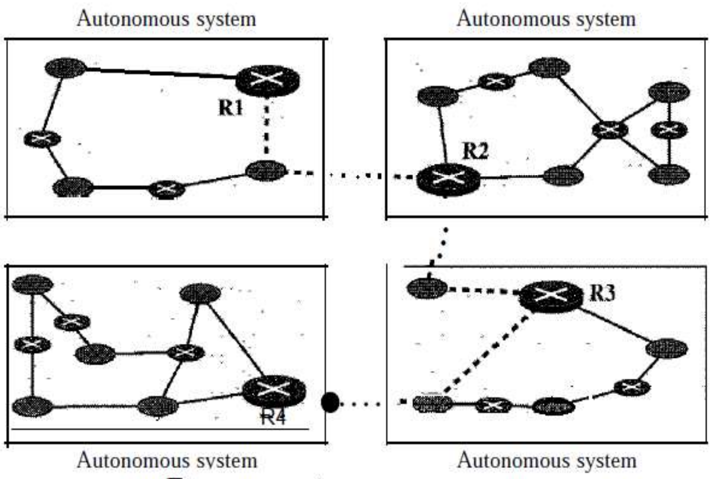


Figure 3.1 *Autonomous systems*

- Routing inside an autonomous system is referred to as *intradomain routing*. Distance vector and link state are intra domain routing.
- Routing between autonomous systems is referred to as *interdomain routing*. Each autonomous system can choose one or more intradomain routing protocols to handle routing inside the autonomous system.*intradomain* routing protocols, or *interior gateway protocols* (IGPs).

✓ A routing table can be either static or dynamic. A *static table* is one with manual entries. A *dynamic table,* is one that is updated automatically when there is a change somewhere in the internet.

✓ The routing protocols are shown in Figure 3.2.

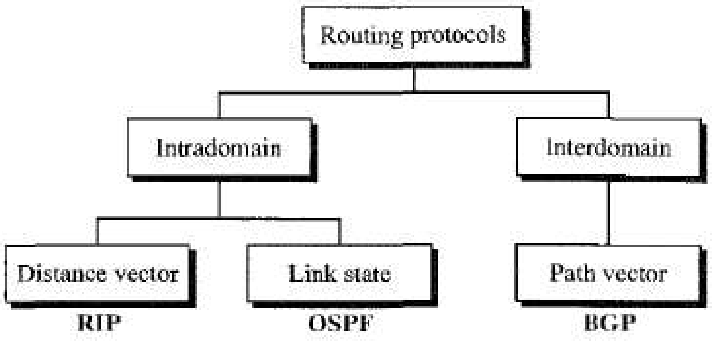


Figure 3.2 *Popular routing protocols*

Network as a Graph

- Routing is a problem of graph theory.
- Figure 3.3 shows a graph representing a network.
  - ✓ The nodes of the graph, labeled A through F, may be hosts, switches, routers, or networks.
  - ✓ The edges of the graph correspond to the network links.
  - ✓ Each edge has an associated *cost*, which gives some indication of the desirability of sending traffic over that link.
- ***Problem of routing***
  - ✓ To find the lowest-cost path between any two nodes, where the cost of a path equals the sum of the costs of all the edges that make up the path.
  - ✓ For a simple network like Figure 3.3, just calculating all the shortest paths and loading them into some nonvolatile storage on each node.
  - ✓ Such a static approach has several shortcomings:
    - o It does not deal with node or link failures.
    - o It does not consider the addition of new nodes or links.
    - o It implies that edge costs cannot change, even though we might reasonably wish to have link costs

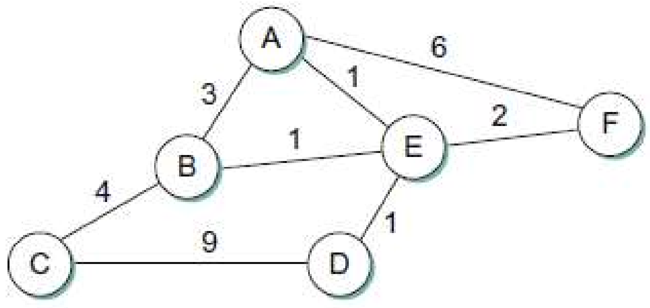


Figure 3.3. Network represented as a graph

change over time (e.g., assigning high cost to a link that is heavily loaded).

- These protocols provide a distributed, dynamic way to solve the problem of finding the lowest-cost path in the presence of link and node failures and changing edge costs.
- The two main classes of routing protocols: *distance vector* and *link state*.

3.1.1. Distance-Vector (RIP)

- Each node constructs a one-dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors
- The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.
- These costs may be provided when the router is configured by a network manager. A link that is down is assigned an infinite cost.

Operation

Consider an example Figure 3.4.

- In this example, the cost of each link is set to 1, so that a least-cost path is simply the one with the fewest hops.
- Each node's knowledge about the distances to all other nodes as a table like Table (Global View).

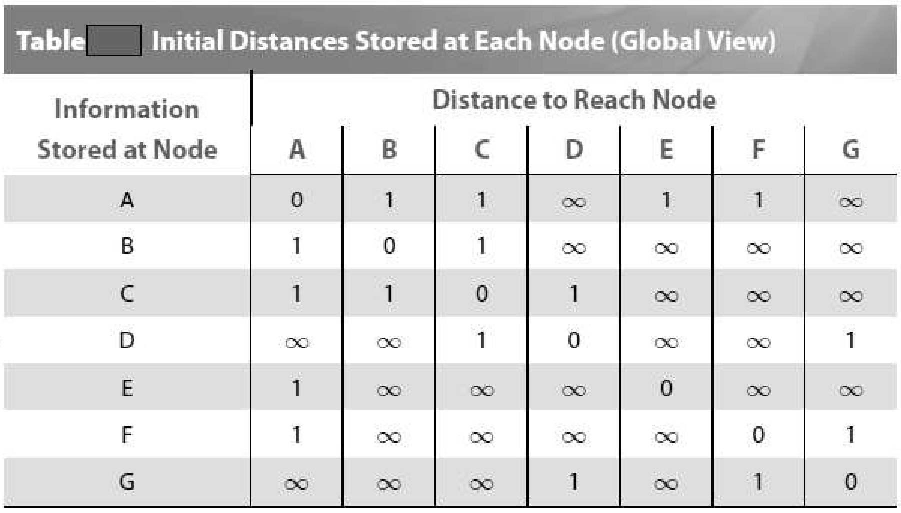


Table [redacted] Initial Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

- Note that each node knows only the information in one row of the table
- Each row in above Table as a list of distances from one node to all other nodes, representing the current beliefs of that node.
- Initially, each node sets a cost of 1 to its directly connected neighbors and ∞ to all other nodes.
- A initially believes that it can reach B in one hop and that D is unreachable. The routing table stored at A reflects this set of beliefs and includes the name of the next hop that A would use to reach any reachable node.
- Initially, then, A's routing table would look like Table (Node A)
- The next step in distance-vector routing is that every node sends

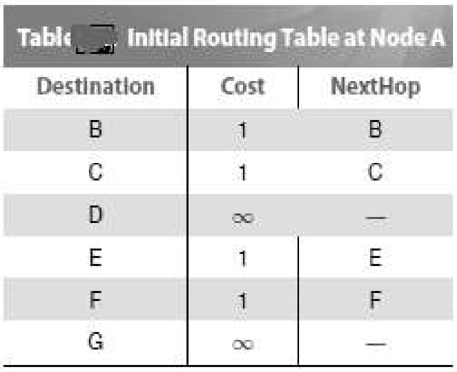


Table [redacted] Initial Routing Table at Node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

a message to its directly connected neighbors containing its personal list of distances.

- For example, node F tells node A that it can reach node G at a cost of 1; A also knows it can reach F at a cost of 1, so it adds these costs to get the cost of reaching G by means of F.
- This total cost of 2 is less than the current cost of infinity, so A records that it can reach G at a cost of 2 by going through F
- Similarly, A learns from C that D can be reached from C at a cost of 1; it adds this to the cost of reaching C (1) and decides that D can be reached via C at a cost of 2, which is better than the old cost of infinity.
- At the same time, A learns from C that B can be reached from C at a cost of 1, so it concludes that the cost of reaching B via C is 2.
- A can update its routing table with costs and next hops for all nodes in the network. The result is shown in Table (Final routing table)
- The process of getting consistent routing information to all the nodes is called *convergence*. Table shows the final set of costs from each node to all other nodes when routing has converged.
- There are two different circumstances under which a given node decides to send a routing update to its neighbors.
    - The *periodic* update. In this case, each node automatically sends an update message every so often, even if nothing has changed.
    - Second mechanism, sometimes called a *triggered* update, happens whenever a node notices a link failure or receives an update from one of its neighbors that causes it to change one of the routes in its routing table.
    - Whenever a node's routing table changes, it sends an update to its neighbors, which may lead to a change in their tables, causing them to send an update to their neighbors.

Table [redacted] Final Routing Table at Node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	2	C
E	1	E
F	1	F
G	2	F

Table -- Final Distances Stored at Each Node (Global View)

Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

- What happens when a link or node fails?
    - The nodes that notice first send new lists of distances to their neighbors, and normally the system settles down fairly quickly to a new state.
- How a node detects a failure

- In one approach, a node continually tests the link to another node by sending a control packet and seeing if it receives an acknowledgment.
- Another approach, a node determines that the is down if it does not receive the expected periodic routing update for the last few update cycles.

• Consider what happens when F detects that its link to G has failed.
  - First, F sets its new distance to G to infinity and passes that information along to A.
  - Since A knows that its 2-hop path to G is through F, A would also set its distance to G to infinity.
  - With the next update from C, A would learn that C has a 2-hop path to G. Thus, A would know that it could reach G in 3 hops through C, which is less than infinity, and so A would update its table accordingly.
  - When it advertises this to F, node F would learn that it can reach G at a cost of 4 through A, which is less than infinity, and the system would again become stable.
  - There are several partial solutions to this problem.
    - ✓ The first one is to use some relatively small number as an approximation of infinity. For example, we might decide that the maximum number of hops to get across a certain network is never going to be more than 16, and so we could pick 16 as the value that represents infinity.
    - ✓ This at least bounds the amount of time that it takes to count to infinity.
    - ✓ One technique to improve the time to stabilize routing is called *split horizon*.
      - The idea is that when a node sends a routing update to its neighbors, it does not send those routes it learned from each neighbor back to that neighbor.
      - For example, if B has the route (E, 2, A) in its table, then it knows it must have learned this route from A, and so whenever B sends a routing update to A, it does not include the route (E, 2) in that update. In a stronger variation of split horizon, called *split horizon with poison reverse*, B actually sends that route back to A, but it puts negative information in the route to ensure that A will not eventually use B to get to E.

```
#define MAX_ROUTES 128 /* maximum size of routing table */
#define MAX_TTL 120 /* time (in seconds) until route expires */
typedef struct
{
   NodeAddr Destination; /* address of destination */
   NodeAddr NextHop; /* address of next hop */
   int Cost; /* distance metric */
   u_short TTL; /* time to live */
} Route;
int numRoutes = 0;
Route routingTable[MAX_ROUTES];
```

    - ✓ The routine that updates the local node's routing table based on a new route is given by merge Route.
      - A timer function periodically scans the list of routes in the node's routing table, decrements the TTL (time to live) field of each route, and discards any routes that have a time to live of 0.

```
void mergeRoute (Route *new)
```

```
     {
        int i;
        for (i = 0; i < numRoutes; ++i)
        {
              if (new->Destination == routingTable[i].Destination)
              {
                    if (new->Cost + 1 < routingTable[i].Cost)
                    {
                          /* found a better route: */
                          break;
                    }
                    else if (new->NextHop == routingTable[i].NextHop)
                    {
                          /*  metric  for  current  next-hop  may  have
                          changed: */
                          break;
                    }
                    else
                    {
                          /* route is uninteresting---just ignore it */
                          return;
                    }
                    }
              }
              if (i == numRoutes)
              {
              /* this is a completely new route; is there room for it?
        */
              if (numRoutes < MAXROUTES)
              {
                    ++numRoutes;
              }
              else
              {
                    /* can't fit this route in table so give up */
                    return;
              }
  }
  routingTable[i] = *new;
  /* reset TTL */
```

```
        routingTable[i].TTL = MAX_TTL;
        /* account for hop to get to next node */
        ++routingTable[i].Cost;
}
```

- ✓ Finally, the procedure update Routing Table is the main routine that calls merge Route to incorporate all the routes contained in a routing update that is received from a neighboring node.

```
void updateRoutingTable (Route *newRoute, int numNewRoutes)
{
   int i;
   for (i=0; i < numNewRoutes; ++i)
   {
         mergeRoute(&newRoute[i]);
   }
}
```

Routing Information Protocol (RIP)

- The goal of the routers is to learn how to forward packets to various *networks*.
- For example, in Figure 3.4, router C would advertise to router A the fact that it can reach networks 2 and 3 (to which it is directly connected) at a cost of 0, networks 5 and 6 at cost 1, and network 4 at cost 2.
- Evidence of this in the RIP (version 2) packet format in Figure 3.5.
- For example, if router A learns from router B that network X can be reached at a lower cost via B than via the existing next hop in the routing table, A updates the cost and next hop information for the network number accordingly.

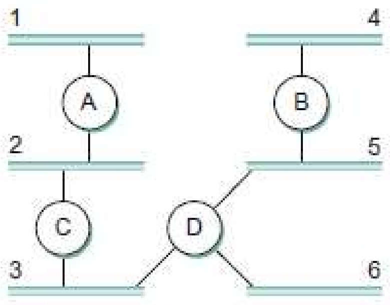


Figure 3.4 example network running RIP

- Routers running RIP send their advertisements every 30 seconds; a router also sends an update message whenever an update from another router causes it to change its routing table.
- RIP takes the simplest approach, with all link costs being equal to 1, just as in our example above. Thus, it always tries to find the minimum hop route.
- Valid distances are 1 through 15, with 16 representing infinity. This also limits RIP to running on fairly small networks—those with no paths longer than 15 hops.

3.1.2. Link State (OSPF)

- ***Goals***
  - A link-state protocol’s flooding mechanism is that the newest information must be flooded to all nodes as quickly as possible, while old information must be removed from the network and not allowed to circulate.
    - To minimize the total amount of routing traffic that is sent around the network;
    - To achieve these goals,

- ✓ To reduce overhead is to avoid generating LSPs unless absolutely necessary. This can be done by using very long timers for the periodic generation of LSPs. Given that the flooding protocol is truly reliable when topology changes, it is safe to assume that messages saying "nothing has changed" do not need to be sent very often.
  - ▪ LSPs carry sequence numbers. Each time a node generates a new LSP, it increments the sequence number by 1.
  - ▪ If a node goes down and then comes back up, it starts with a sequence number of 0. If the node was down for a long time, all the old LSPs for that node will have timed out (as described below); otherwise, this node will eventually receive a copy of its own LSP with a higher sequence number, which it can then increment and use as its own sequence number.

- Each node is assumed to be capable of finding out the state of the link to its neighbors (up or down) and the cost of each link.
- To provide each node with enough information to enable it to find the least-cost path to any destination.
- Linkstate routing protocols rely on two mechanisms: reliable dissemination of link-state information, and the calculation of routes from the sum of all the accumulated link-state knowledge.

Reliable Flooding

- *Reliable flooding* is the process of making sure that all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes.
- The *idea* is for a node to send its link-state information out on all of its directly connected links; each node that receives this information then forwards it out on all of *its* links. This process continues until the information has reached all the nodes in the network.
- Each node creates an update packet, also called a *link-state packet* (LSP), which contains the following information:
  - ✓ The ID of the node that created the LSP
  - ✓ A list of directly connected neighbors of that node, with the cost of the link to each one
  - ✓ A sequence number
  - ✓ A time to live for this packet
- The first two items are needed to enable route calculation; the last two are used to make the process of flooding the packet to all nodes reliable.
- Flooding works in the following way.
  - ✓ First, the transmission of LSPs between adjacent routers is made reliable using acknowledgments and retransmissions just as in the reliable link-layer protocol.
  - ✓ Consider a node X that receives a copy of an LSP that originated at some other node Y. Note that Y may be any other router in the same routing domain as X. X checks to see if it has already stored a copy of an LSP from Y. If not, it stores the LSP.
  - ✓ If it already has a copy, it compares the sequence numbers; if the new LSP has a larger sequence number, it is assumed to be the more recent, and that LSP is stored, replacing the old one.

- ✓ A smaller (or equal) sequence number would imply an LSP older (or not newer) than the one stored, so it would be discarded and no further action would be needed. If the received LSP was the newer one, X then sends a copy of that LSP to all of its neighbors except the neighbor from which the LSP was just received.
- ✓ The fact that the LSP is not sent back to the node from which it was received helps to bring an end to the flooding of an LSP. Since X passes the LSP on to all its neighbors, who then turn around and do the same thing, the most recent copy of the LSP eventually reaches all nodes.
- ✓ Figure 3.5 shows an LSP being flooded in a small network.
    - Each node becomes shaded as it stores the new LSP.
    - In Figure 3.5(a) the LSP arrives at node X, which sends it to neighbors A and C in Figure 3.5(b).
        - A and C do not send it back to X, but send it on to B. Since B receives two identical copies of the LSP, it will accept whichever arrived first and ignore the second as a duplicate.
        - It then passes the LSP onto D, which has no neighbors to flood it to, and the process is complete.

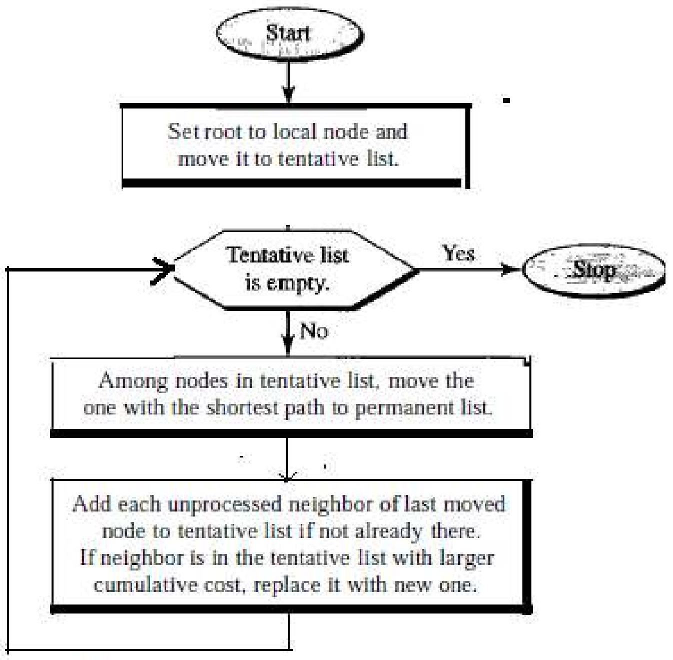
- Each node generates LSPs under two circumstances.
    - Either *the expiry of a periodic timer* or *a change in topology* can cause a node to generate a new LSP.
    - The failure of a neighbor or loss of connectivity to that neighbor can be detected using periodic "hello" packets.
    - Each node sends these to its immediate neighbors at defined intervals.
    - If a sufficiently long time passes without receipt of a "hello" from a neighbor, the link to that neighbor will be declared down, and a new LSP will be generated to reflect this fact.

Route Calculation

- Once a given node has a copy of the LSP from every other node, it is able to compute a complete map for the topology of the network, and from this map it is able to decide the best route to each destination.

Dijkstra's shortest-path algorithm

- After receiving all LSPs, each node will have a copy of the whole topology.
- 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.
- The Dijkstra algorithm creates a shortest path tree from a graph.
- The algorithm divides the nodes into two sets: *tentative* and *permanent*.
    - It finds the neighbors of a current node, makes them tentative, examines them, and


Start
Set root to local node and move it to tentative list.
Tentative list is empty.
Yes
Stop
No
Among nodes in tentative list, move the one with the shortest path to permanent list.
Add each unprocessed neighbor of last moved node to tentative list if not already there. If neighbor is in the tentative list with larger cumulative cost, replace it with new one.
Figure 3.5 Dijkstra algorithm

if they pass the criteria, makes them permanent. Define the algorithm by using the flowchart in Figure 3.5.

- The following shows the steps.

  At the end of each step, we show the permanent (filled circles) and the tentative (open circles) nodes and lists with the cumulative costs.

Example

1. Node A the root of the tree and move it to the tentative list. Our two lists are

   Permanent list: empty Tentative list: A(O)

2. Node A has the shortest cumulative cost from all nodes in the tentative list. Mve A to the pennanent list and add all neighbors ofA to the tentative list. Our new lists are

   Permanent list: A(O) Tentative list: B(5), C(2), D(3)

3. Node C has the shortest cumulative cost from all nodes in the tentative list. Move C to the permanent list. Node C has three neighbors, but node A is already processed, which makes the unprocessed neighbors just B and E. However, B is already in the tentative list with a cumulative cost of 5. Node A could also reach node B through C with a cumulative cost of 6. Since 5 is less than 6, we keep node B with a cumulative cost of 5 in the tentative list and do not replace it. Our new lists are

   Permanent list: A(O), E(2) Tentative list: B(5), 0(3), E(6)

4. Node D has the shortest cumulative cost of all the nodes in the tentative list. Move D to the permanent list. Node D has no unprocessed neighbor to be added to the tentative list. Our new lists are

   Permanent list: A(O), C(2), 0(3) Tentative list: B(5), E(6)

5. Node B has the shortest cumulative cost of all the nodes in the tentative list. Move B to the permanent list. We need to add all unprocessed neighbors of B to the tentative list (this is just node E). However, E(6) is already in the list with a smaller cumulative cost. The cumulative cost to node E, as the neighbor of B, is 8. node E(6) in the tentative list. Our new lists are

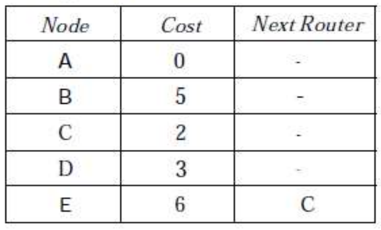
   Permanent list: A(O), B(5), C(2), 0(3) Tentative list: E(6)

6. Node E has the shortest cumulative cost from all nodes in the tentative list. Move E to the permanent list. Node E has no neighbor. Now the tentative list is empty. We stop; our shortest path tree is ready. The final lists are

   Permanent list: A(O), B(5), C(2), D(3), E(6) Tentative list: empty

Calculation of Routing Table from Shortest Path Tree Each node uses the shortest path tree protocol to construct its routing table. The routing table shows the cost of reaching each node from the root. Table shows the routing table for node A.

Routing table for node A



Node	Cost	Next Router
A	0	-
B	5	-
C	2	-
D	3	-
E	6	C

Dijikstrasalgorithm is defined as follows:

```
M = {s}
for each n in N −{s}
        C(n) = l(s,n)
while (N 6=M)
        M =M ∪ {w} such that C(w) is the minimum for all w in (N −M)
        for each n in (N −M)
                C(n) = MIN(C(n),C(w)+l(w,n))
```

1. Initialize the Confirmed list with an entry for myself; this entry has a cost of 0.

2. For the node just added to the Confirmed list in the previous step, call it node Next and select its LSP.
3. For each neighbor (Neighbor) of Next, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor.
   a. If Neighbor is currently on neither the Confirmed nor the Tentative list, then add (Neighbor, Cost, NextHop) to the Tentative list, where NextHop is the direction I go to reach Next.
   b. If Neighbor is currently on the Tentative list, and the Cost is less than the currently listed cost for Neighbor, then replace the current entry with (Neighbor, Cost, NextHop), where NextHop is the direction I go to reach Next.
4. If the Tentative list is empty, stop. Otherwise, pick the entry from the tentative list with the lowest cost, move it to the

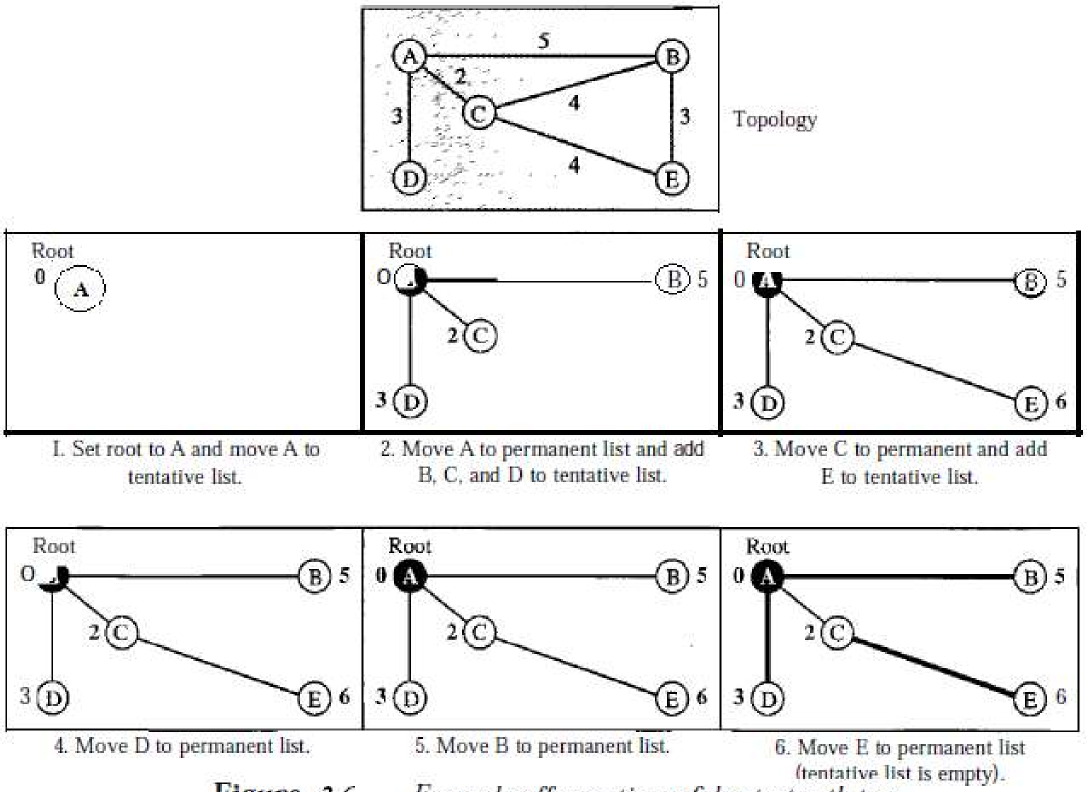

Topology
Root
1. Set root to A and move A to tentative list.
2. Move A to permanent list and add B, C, and D to tentative list.
3. Move C to permanent and add E to tentative list.
4. Move D to permanent list.
5. Move B to permanent list.
6. Move E to permanent list (tentative list is empty).

Figure 3.6 *Example offormation ofshortest path tree*

Confirmed list, and return to step 2.

Features of OSPF

1. *Authentication of routing messages*—Early versions of OSPF used a simple 8-byte password for authentication.
2. *Additional hierarchy*—Hierarchy is one of the fundamental tools used to make systems more scalable. OSPF introduces another layer of hierarchy into routing by allowing a domain to be partitioned into *areas*. This is a reduction in the amount of information that must be transmitted to and stored in each node.
3. *Load balancing*—OSPF allows multiple routes to the same place to be assigned the same cost and will cause traffic to be distributed evenly over those routes, thus making better use of available network capacity.

Types of OSPF Messages

- Header as shown in Figure 3.34.
  - *Version* field is currently set to 2

- *Type* field may take the values 1 through 5.
- *SourceAddr* identifies the sender of the message
- *AreaId* is a 32-bit identifier of the area in which the node is located.
- The entire packet, *except the authentication data*, is protected by a 16-bit checksum using the same algorithm as the IP header

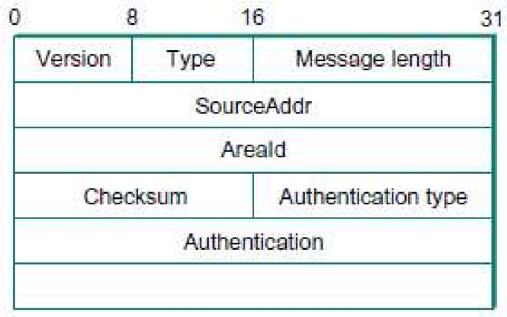


Figure 3.7 OSPF header format

- *Authentication type* is 0 if no authentication is used; otherwise, it may be 1, implying that a simple password is used, or 2, which indicates that a cryptographic authentication

• Figure 3.8 shows the packet format for a type 1 link-state advertisement.
- *Type 1* LSAs advertise the cost of links between routers.
- *Type 2* LSAs are used to advertise networks to which the advertising router is connected, while other types are used to support additional hierarchy
- The LS Age is the equivalent of a time to live, except that it counts up and the LSA expires when the age reaches a defined maximum value.
- The *Type* field tells us that this is a type 1 LSA -In a type 1 LSA,
- *Advertising router* field are identical. Each carries a 32-bit identifier for the router that created this LSA.

  While a number of assignment strategies may be used to assign this ID, it is essential that it be unique in the routing domain and that a given router consistently uses the same router ID.

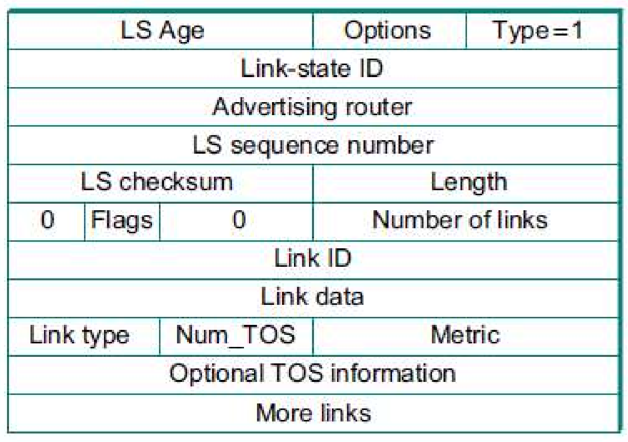


Figure 3.8 OSPF Header Format

- The *LS sequence number* is used exactly as described above to detect old or duplicate LSAs.
- The *LS checksum* is used to detect and correct the errors
- LS Age, to recompute a checksum every time LS Age is incremented.
- *Length* is the length in bytes of the complete LSA.
- *Link ID, Link Data* - identify the link.

- *Metric*. The metric is of course the cost of the link.
- *Type* tells us something about the link for example, if it is a point-to-point link.
- *TOS* information is present to allow OSPF to choose different routes for IP packets based on the value in their TOS field. For example, if we had a link in our network that was very good for delay-sensitive traffic, we could give it a low metric for the TOS value representing low delay and a high metric for everything else. OSPF would then pick a different shortest path for those packets that had their TOS field set to that value.

3.1.3. Metrics

- To calculate link costs that have proven effective in practice.
- One example to assign a cost of 1 to all links—the least-cost route will then be the one with the fewest hops.

Drawbacks,

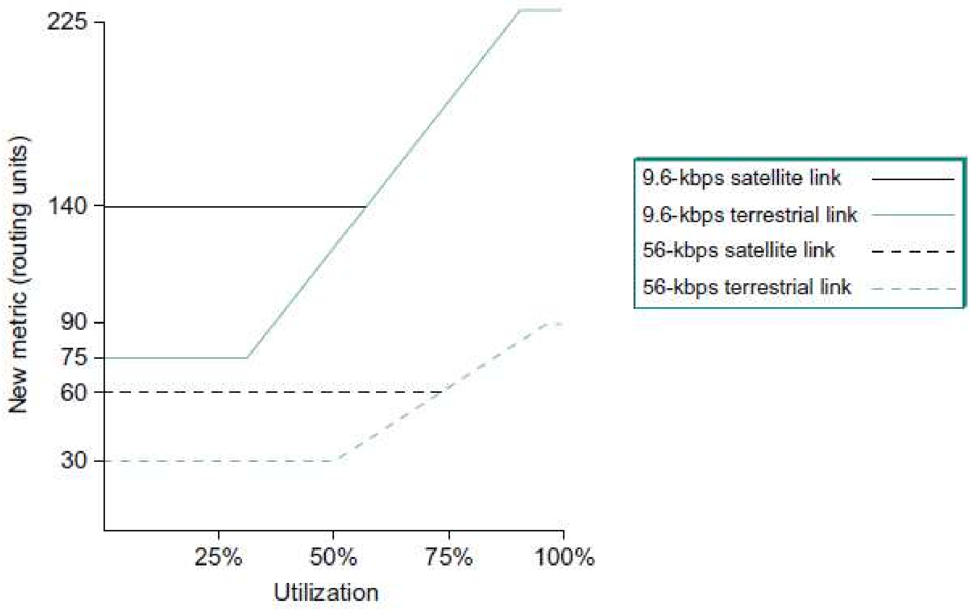
1. It does not distinguish between links on a latency basis. Thus, a satellite link with 250-ms latency looks just as attractive to the routing protocol as a terrestrial link with 1-ms latency.
2. It does not distinguish between routes on a capacity basis, making a 9.6-kbps link look just as good as a 45-Mbps link.
3. It does not distinguish between links based on their current load, making it impossible to route around overloaded links.

- The ARPANET was the testing ground for a number of different approaches to link-cost calculation.

Evolution of the ARPANET

- The original ARPANET routing metric measured the number of packets that were queued waiting to be transmitted on each link, meaning that a link with 10 packets queued waiting to be transmitted was assigned a larger cost weight than a link with 5 packets queued for transmission.
  - Using queue length as a routing metric did not work well, however, since queue length is an artificial measure of load—it moves packets toward the shortest queue rather than toward the destination.
  - A second version of the ARPANET routing algorithm, sometimes called the *new routing mechanism*, took both link bandwidth and latency into consideration and used delay, rather than just queue length, as a measure of load. This was done as follows.
  - o Each incoming packet was time stamped with its time of arrival at the router (ArrivalTime); its departure time from the router (DepartTime) was also recorded.
  - o When the link-level ACK was received from the other side, the node computed the delay for that packet as `Delay = (DepartTime−ArrivalTime)+TransmissionTime+Latency`, where
    - TransmissionTime and Latency were statically defined for the link and captured the link's bandwidth and latency, respectively.
    - DepartTime − ArrivalTime represents the amount of time the packet was delayed (queued) in the node due to load. If the ACK did not arrive, but instead the packet timed out, then DepartTime was reset to the time the packet was *retransmitted*.
- o Finally, the weight assigned to each link was derived from the average delay experienced by the packets recently sent over that link.

- The compression of the dynamic range was achieved by feeding the measured utilization, the link type, and the link speed into a function that is shown graphically in Figure 3.9. Observe the following:



3.9 Revised ARPANET routing metric versus link utilization

1. A highly loaded link never shows a cost of more than three times its cost when idle.
2. The most expens ive link is only seven times the cost of the least expensive.
3. A high-speed satellite link is more attractive than a low-speed terrestrial link.
4. Cost is a function of link utilization only at moderate to high loads.

- The slopes, offsets, and breakpoints for the curves in Figure 3.9 were arrived at by a great deal of trial and error, and they were carefully tuned to provide good performance.

3.2. Switch basics

A switch is a device that channels incoming data from any of multiple input ports to the specific output port that will take the data toward its proposed destination.

- Figure 3.10 shows a processor with three network interfaces used as a switch.
    - The figure shows a path that a packet might take from the time it arrives on interface 1 until it is output on interface 2.
    - Assumed here that the processor has a mechanism to move data directly from an interface to its main memory without having to be directly copied by the CPU, a technique called *direct memory access* (DMA).

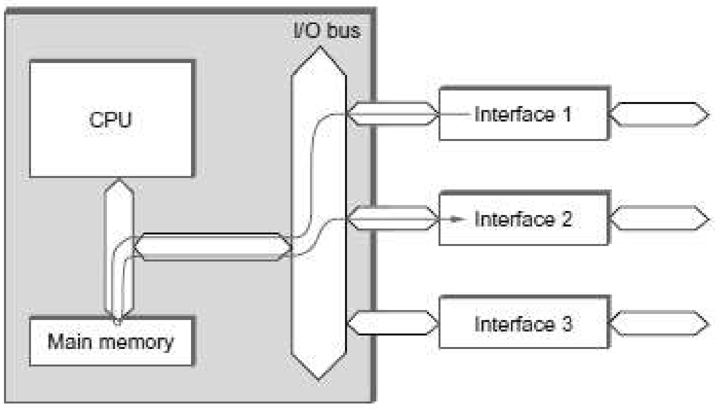


Figure 3.10 A general purpose processor used as a packet switch

  - Once the packet is in memory, the CPU examines its header to determine which interface the packet should be sent out on. It then uses DMA to move the packet out to the appropriate interface.
  - This Figure 3.10 does not show the packet going to the CPU because the CPU inspects only the header of the packet; it does not have to read every byte of data in the packet.
- o *Problem with a general-purpose processor as a switch* is that its performance is limited by the fact that all packets must pass through a single point of contention:
- o In the example shown, each packet crosses the I/O bus twice and is written to and read from main memory once.
- o The upper bound on aggregate throughput of such a device is, thus, either half the main memory bandwidth or half the I/O bus bandwidth, whichever is less.
- o For example, a machine with a 133-MHz, 64-bit-wide I/O bus can transmit data at a peak rate of a little over 8 Gbps. Since forwarding a packet involves crossing the bus twice, the actual limit is 4 Gbps—enough to build a switch with a fair number of 100-Mbps
- o Ethernet ports, for example, but hardly enough for a high-end router in the core of the Internet.
- o The upper bound also assumes that *moving data* is the only problem—a fair approximation for long packets but a bad one when packets are short.
- o The cost of processing each packet— parsing its header and deciding which output link to transmit it on—is likely to dominate.
- o Suppose, for example, that a processor can perform all the necessary processing to switch 2 million packets each second. This is sometimes called the *packet per second (pps) rate*.
- o If the average packet is short, say, 64 bytes, this would imply

$$\begin{aligned} \textit{Throughput} &= \textit{pps} \times (\textit{BitsPerPacket}) \\ &= 2\times106\times64\times8 \\ &= 1024\times106 \end{aligned}$$

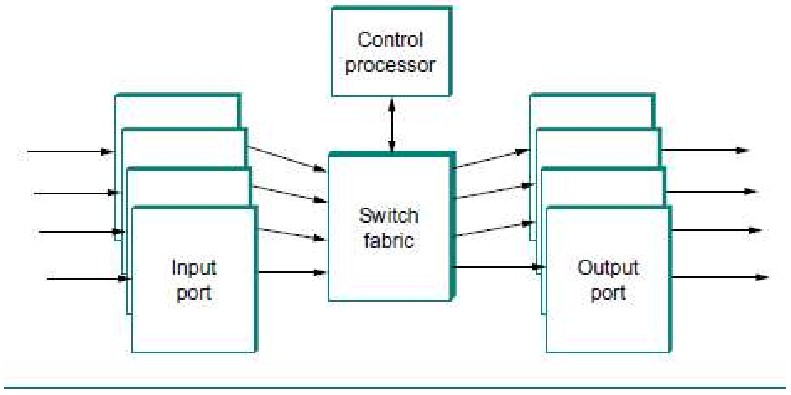
  - where a throughput of about 1 Gbps
  - Ethernet segment is shared among all users connected to the shared medium. Thus, for example, a 20-port switch with this aggregate throughput would only be able to cope with an average data rate of about 50 Mbps on each port.

- The hardware designers have come up with a large array of switch designs that reduce the amount of contention and provide high aggregate throughput.
- If every input has data to send to a single output, then they cannot all send it at once.
- If data destined for different outputs is arriving at different inputs, then a well-designed switch will be able to move data from inputs to outputs in parallel, thus increasing the aggregate throughput.

3.2.1. Ports

- See Figure 3.11 They consist of a number of *input* and *output ports* and a *fabric*.
- There is usually at least one control processor in charge of the whole switch that communicates with the ports either directly or, as shown here, via the switch fabric.
- The ports communicate with the outside world.

- They may contain fiber optic receivers and lasers, buffers to hold packets that are waiting to be switched or transmitted, and often a significant amount of other circuitry that enables the switch to function.
- The fabric has a very simple and well-defined job: When presented with a packet, deliver it to the right output port.
- One of the jobs of the ports, then, is to deal with the complexity of the real world in such a way that the fabric can do its relatively simple job. For example, suppose that this switch is supporting a virtual circuit model of communication.



Figue 3.11 A .4*4 Switch

- The ports maintain lists of virtual circuit identifiers that are currently in use, with information about what output a packet should be sent out on for each VCI and how the VCI needs to be remapped to ensure uniqueness on the outgoing link.
- The ports of an Ethernet switch store tables that map between Ethernet addresses and output ports
- When a packet is handed from an input port to the fabric, the port has figured out where the packet needs to go, and either the port sets up the fabric accordingly by communicating some control information to it, or it attaches enough information to the packet itself (e.g., an output port number) to allow the fabric to do its job automatically.
- Fabrics that switch packets by looking only at the information in the packet are referred to as *self-routing*, since they require no external control to route packets.
  - ➢ An example of a self-routing fabric is discussed below.
    - The input port is the first place to look for performance bottlenecks.
    - The input port has to receive a steady stream of packets, analyze information in the header of each one to determine which output port (or ports) the packet must be sent to, and pass the packet on to the fabric.
    - The type of header analysis that it performs can range from a simple table lookup on a VCI to complex matching algorithms that examine many fields in the header. This is the type of operation that sometimes becomes a problem when the average packet size is very small.
    - Consider, for example, 64-byte packets arriving on a port connected to an OC-48 (2.48 Gbps) link. Such a port needs to process packets at a rate of

$$2.48\times10^{9}\div(64\times8) = 4.83\times10^{6} \text{ pps}$$

- Another key function of ports is buffering.

Limitations of Simple input buffering

- Consider an input buffer implemented as a FIFO.
- As packets arrive at the switch, they are placed in the input buffer.
- The switch then tries to forward the packets at the front of each FIFO to their appropriate output port. However, if the packets at the front of several different input ports are destined for the same output port at the same time, then only one of them can be forwarded; the rest must stay in their input buffers.
- Those packets left at the front of the input buffer prevent other packets further back in the buffer from getting a chance to go to their chosen outputs, even though there may be no contention for those outputs. This phenomenon is called *head-of-line blocking*.
- A simple example of head-of-line blocking is given in Figure 3.12, where we see a packet destined for port 1 blocked behind a packet contending for port 2.

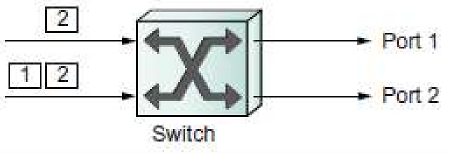


Figure 3.12 Simple illustration of head-of-line blocking

- It can be shown that when traffic is uniformly distributed among outputs, head-of-line blocking limits the throughput of an input-buffered switch to 59% of the theoretical maximum.

3.2.2. Fabrics

- A switch fabric should be able to move packets from input ports to output ports with minimal delay and in a way that meets the throughput goals of the switch.
- A high-performance fabric with n ports can often move one packet from each of its n ports to one of the output ports at the same time.

Fabric Types

- *Shared Bus*—The bus bandwidth determines the throughput of the switch, high-performance switches usually have specially designed busses rather than the standard busses found in PCs.
- *SharedMemory*—In a shared memory switch, packets are written into a memory location by an input port and then read from memory by the output ports.
  - It is the memory bandwidth that determines switch throughput, so wide and fast memory is typically used in this sort of design.
  - A shared memory switch is similar in principle to the shared bus switch, except it usually uses a specially designed, high-speed memory bus rather than an I/O bus.
- *Crossbar*—A crossbar switch is a matrix of pathways that can be configured to connect any input port to any output port.
  - Figure 3.13 shows a 4×4 crossbar switch.
  - *Problem with crossbars* is that, they require each output port to be able to accept packets from all inputs at once, implying that each port would have a memory bandwidth equal to the total switch throughput.
- *Self-routing*— self-routing fabrics rely on some information in the packet header to direct each packet to its correct output.

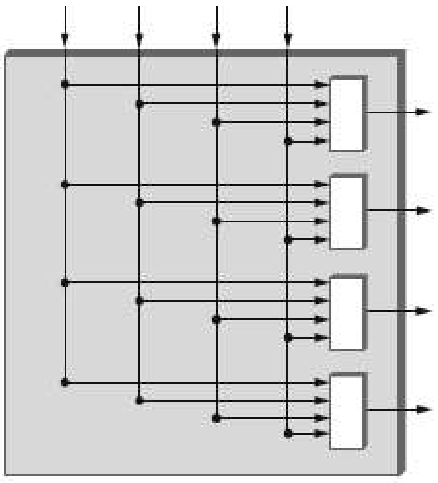


Figure 3.13 A 4*4 crossbar switch

- Usually a special "self-routing header" is appended to the packet by the input port after it has determined which output the packet needs to go to, as illustrated in Figure 3.14; this extra header is removed before the packet leaves the switch.
- Self-routing fabrics are often built from large numbers of very simple 2×2 switching elements interconnected in regular patterns, such as the *banyan* switching fabric shown in Figure 3.42.

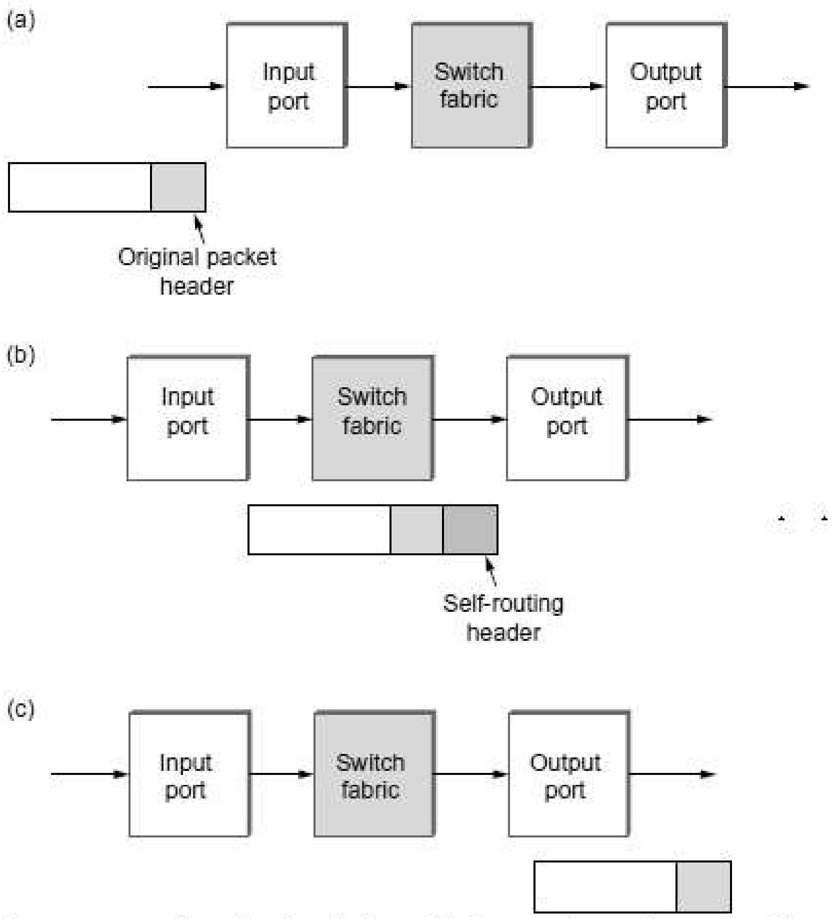


Figure 3.14 Aself-routing header is applied to a packet at input to enable the fabric to send the packet to the correct output, where it is removed: (a)Packet arrives at input port; (b) input port attaches self-routing header to direct packet to correct output; (c)self-routing header is removed at output port before packet leaves switch

- Self-routing fabrics are among the most scalable approaches to fabric design, and there has been a wealth of research on the topic, some of which is listed in the Further Reading section. Many self-routing fabrics resemble the one shown in Figure 3.15, consisting of regularly interconnected 2×2 switching elements. For example, the 2×2 switches in the banyan network perform a simple task:
- They look at 1 bit in each self routing header and route packets toward the upper output if it is zero or toward the lower output if it is one.

- If two packets arrive at a banyan element at the same time and both have the bit set to the same value, then they want to be routed to the same output and a collision will occur.

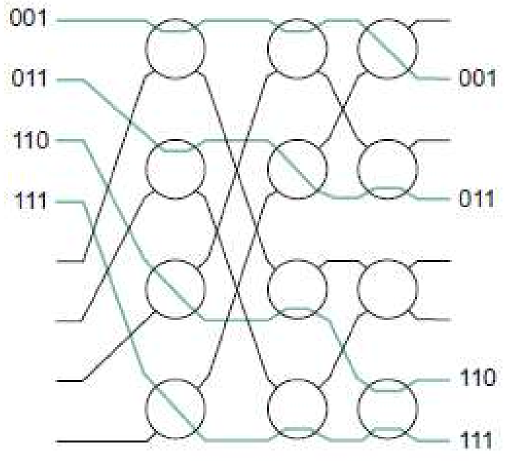


Figure 3.15 Routing packets through a banyan network. The 3-bit numbers represent values in the self-routing headers of four arriving packets

- The banyan network is a clever arrangement of 2×2 switching elements that routes all packets to the correct output without collisions if the packets are presented in ascending order.
- Works in an example, as shown in Figure 3.15, where the self-routing header contains the output port number encoded in binary.
  - The switch elements in the first column look at the most significant bit of the output port number and route packets to the top if that bit is a 0 or the bottom if it is a 1.
  - Switch elements in the second column look at the second bit in the header, and those in the last column look at the least significant bit.
  - The packets are routed to the correct destination port without collisions.
  - The top outputs from the first column of switches all lead to the top half of the network, thus getting packets with port numbers 0 to 3 into the right half of the network.
  - The next column gets packets to the right quarter of the network, and the final column gets them to the right output port.
  - The clever part is the way switches are arranged to avoid collisions.
  - Part of the arrangement includes the "perfect shuffle" wiring pattern at the start of the network. To build a complete switch fabric around a banyan network would require additional components to sort packets before they are presented to the banyan.
- The Batcher network, which is also built from a regular interconnection of 2×2 switching elements, sorts packets into descending order.
  - The Batcher network, the packets are then ready to be directed to the correct output, with no risk of collisions, by the banyan network.

3.2.3. Router Implementation

- To build a switch, ranging from a general-purpose processor with a suitable number of network interfaces to some sophisticated hardware designs.
- In general, the same range of options is available for building routers, many of which look something like Figure 3.16.
- The control processor is responsible for running the routing protocols discussed above, among other things, and generally acts as the central point of control of the router.
- The switching fabric transfers packets from one port to another, just as in a switch; and the ports provide a range of functionality to allow the router to interface to links of various types (e.g., Ethernet, SONET).

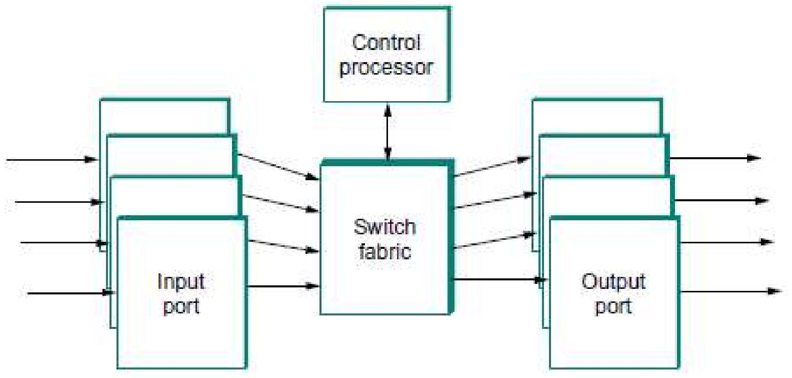


Figure 3.16 Block diagram of a router

- How router design differs from switch design.
  1. Routers must be designed to handle variable length packets, a constraint that does not apply to ATM switches but is certainly applicable to Ethernet or Frame Relay switches.
     - In such cases, the ports must be able to convert variable length packets into cells and back again. This is known as *segmentation and re-assembly* (SAR), a problem also faced by network adaptors for ATM networks.
     - Another consequence of the variable length of IP datagrams is that it can be harder to characterize the performance of a router than a switch that forwards only cells.
     - Routers can usually forward a certain number of packets per second, and this implies that the total throughput in *bits* per second depends on packet size.
     - Router designers generally have to make a choice as to what packet length they will support at *line rate*.
     - That is, if pps (packets per second) is the rate at which packets arriving on a particular port can be forwarded, and linerate is the physical speed of the port in bits per second, then there will be some packetsize in bits such that:

$$\text{packetsize} \times \text{pps} = \text{linerate}$$

  - This is the packet size at which the router can forward at line rate; it is likely to be able to sustain line rate for longer packets but not for shorter packets.
  2. Another choice might be the expected *average* packet size, which can be determined by studying traces of network traffic.
  - For example, measurements of the Internet backbone suggest that the average IP packet is around 300 bytes long.

3. When it comes to the task of forwarding IP packets, routers can be broadly characterized as having either a *centralized* or *distributed* forwarding model.
4. The IP forwarding algorithm itself. In bridges an most ATM switches, the forwarding algorithm simply involves looking up a fixed-length identifier (MAC address or VCI) in a table, finding the correct output port in the table, and sending the packet to that port.

3.3. Global Internet

- Internet has hundreds of thousands of networks connected to it.
- Figure 3.17 gives a simple depiction of the state of the Internet in 1990.

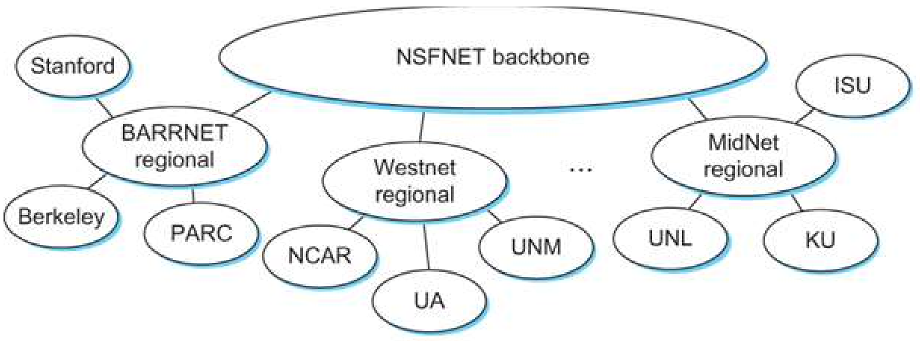


Figure 3.17 The tree structure of the Internet in 1990

- Figure 3.18 One of the salient features of this topology is that it consists of end user sites (e.g., Stanford University) that connect to service provider networks (e.g., BARRNET was a provider network that served sites in the San Francisco Bay Area).
- In 1990, many providers served a limited geographic region and were thus known as *regional networks*. The regional networkswere, in turn, connected by a nationwide backbone.
- In 1990, this backbone was funded by the National Science Foundation (NSF) and was therefore called the *NSFNET backbone*.

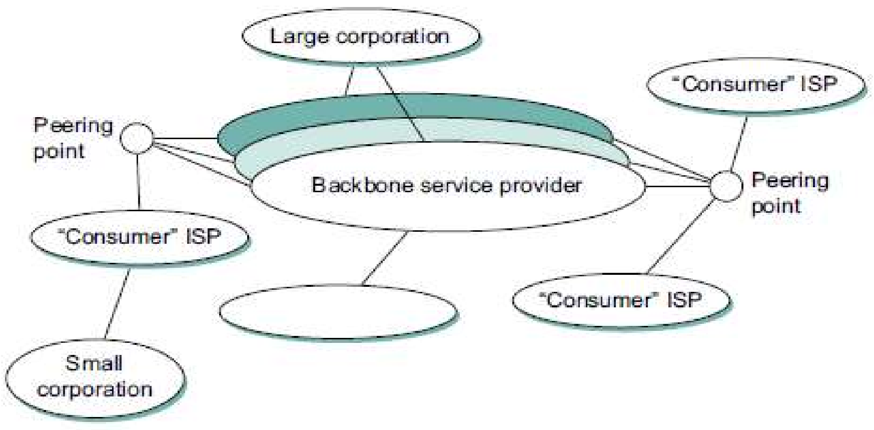


Figure 3.18 A simple multi-provider internet

3.3.1. Routing Areas

- An area is a set of routers that are administratively configured to exchange link-state information with each other.
- There is one special area—the backbone area, also known as area 0.

- An example of a routing domain divided into areas is shown in Figure 3.19.
    - Routers R1, R2, and R3 are members of the backbone area. They are also members of at least one non backbone area; R1 is actually a member of both area 1 and area 2.
    - A router that is a member of both the backbone area and a non backbone area is an *area border router (ABR)*.
    - All the routers in the area send link-state advertisements to each other and thus develop a complete, consistent

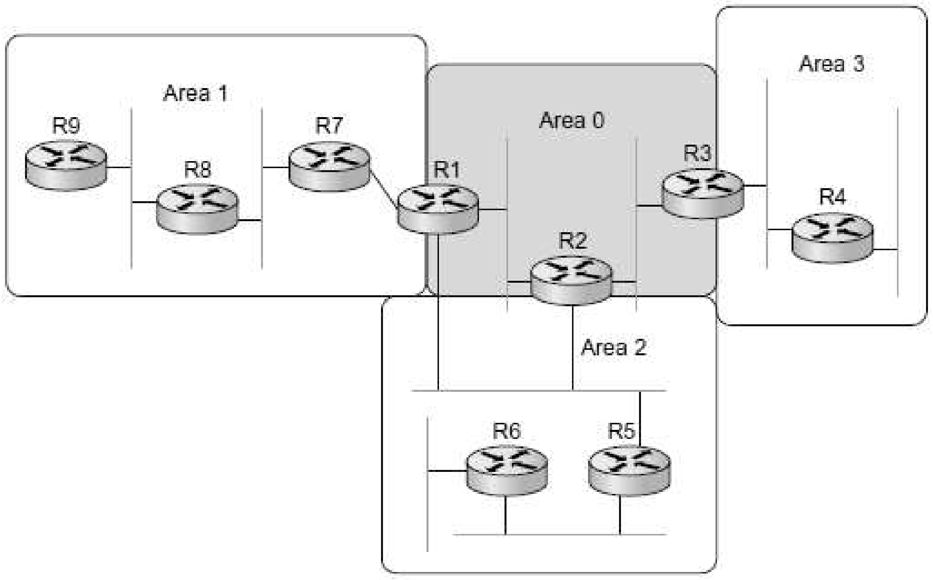


Figure 3.19 A domain divided into areas

map of the area.
    - For example, router R4 in area 3 will never see a link-state advertisement from router R8 in area 1. As a consequence, it will know nothing about the detailed topology of areas other than its own.
- The path of a packet that has to travel from one nonbackbone area to another as being split into three parts.

1. It travels from its source network to the backbone area, then it crosses the backbone, then it travels from the backbone to the destination network.

- For example, R1 receives link-state advertisements from all the routers in area 1 and can thus determine the cost of reaching any network in area 1.
- When R1 sends link-state advertisements into area 0, it advertises the costs of reaching the networks in area 1 much as if all those networks were directly connected to R1. This enables all the area 0 routers to learn the cost to reach all networks in area 1.
- The area border routers then summarize this information and advertise it into the non backbone areas. Thus, all routers learn how to reach all networks in the domain.
- Both R1 and R2 will be advertising costs to various networks, so it will become clear which is the better choice as the routers in area 2 run their shortest-path algorithm.

2. When dividing a domain into areas, the network administrator makes a tradeoff between scalability and optimality of routing. The use of areas forces all packets traveling from one area to another to go via the backbone area, even if a shorter path might have been available.

- For example, even if R4 and R5 were directly connected, packets would not flow between them because they are in different nonbackbone areas.

3. Finally, we note that there is a trick by which network administrators can more flexibly decide which routers go in area 0. This trick uses the idea of a *virtual link* between routers.

- For example, a virtual link could be configured from R8 to R1, thus making R8 part of the backbone.
- R8 would now participate in link-state advertisement flooding with the other routers in area 0.
- The cost of the virtual link from R8 to R1 is determined by the exchange of routing information that takes place in area 1. This technique can help to improve the optimality of routing.

3.3.2. Interdomain Routing (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

- The Internet is divided into hierarchical domains called autonomous systems.
- For example, a large corporation that manages its own network and has full control over it is an autonomous system.
- Internal network might be a single AS, as may the network of a single Internet Service Provider (ISP). Figure 3.20 shows a simple network with two autonomous systems.
- A local ISP that provides services to local customers is an autonomous system.
- Divide autonomous systems into three categories: stub, multihomed, and transit.

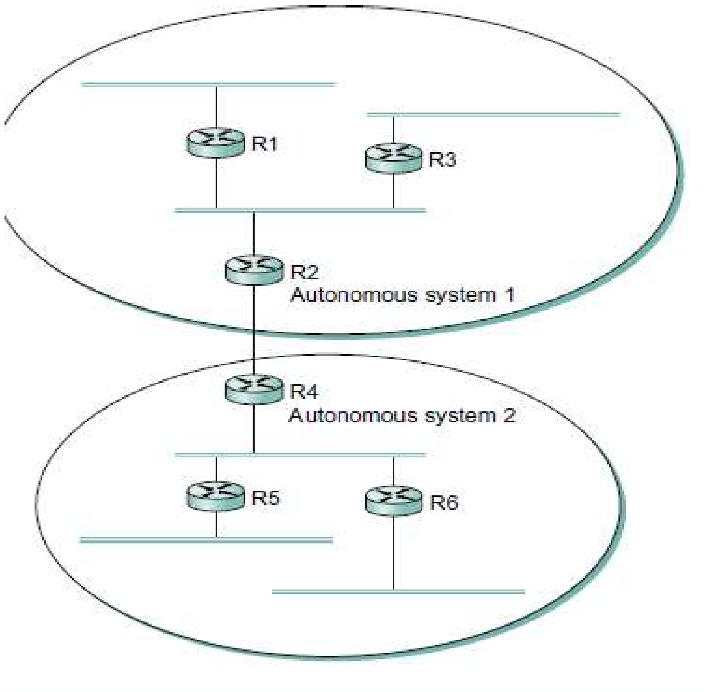


Figure 3.20 A network with two autonomous systems

1. *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.

2. *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.

3. *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).

Path Attributes

- 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 more-informed decision when applying its policy.
- Attributes are divided into two broad categories: well known and optional.
  - ➢ *wellknown attribute :* A wellknown attribute is one that every BGP router must recognize. It is divided into two categories:

  i. *well-known mandatory* is one that must appear in the description of a route. One wellknown 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. another well-known mandatory attribute is NEXT-HOP, which defines the next router to which the data packet should be sent.

  ii. *well-known discretionary* is one that must be recognized by each router, but is not required to be included in every update message.
- *optional attribute :*An optional attribute is one that needs not be recognized by every router. The optional attributes can also be subdivided into two categories: transitive and nontransitive.

  i. An *optional transitive attribute* is one that must be passed to the next router by the router that has not implemented this attribute.

  ii. An *optional nontransitive attribute* is one that must be discarded if the receiving router has not implemented it.

BGP Sessions

A session at the BGP level, as an application program, is a connection at the TCP level. 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 *semi permanent connections.*

External and Internal BGP

- BGP can have two types of sessions
  1. *External BGP (E-BGP)-* The E-BGP session is used to exchange information between two speaker nodes belonging to two different autonomous systems.
  2. *Internal BGP (I-BGP) sessions-* The I-BGP session is used to exchange routing information between two routers inside an autonomous system.
- BGP advertises *complete paths* as detailed list of autonomous systems to reach a particular network. It is sometimes called a *path-vector* protocol.

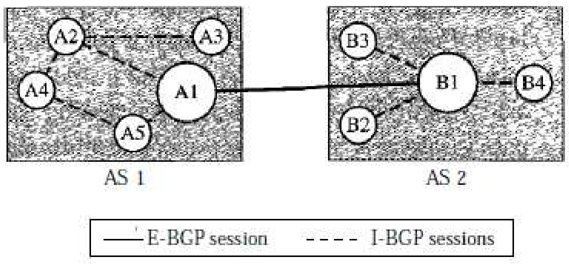


Figure 3.21 *Internal and external BGP sessions*

- It also enables routing loops to be readily detected.
- Example network in Figure 3.22.
  - Assume that the providers are transit networks, while the customer networks are stubs.
  - A BGP speaker for the AS of provider A (AS 2) would be able to advertise reachability information for each of the network numbers assigned to customers P and Q.
  - The networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached directly from AS 2. The backbone network, on receiving this advertisement, can advertise, The networks 128.96, 192.4.153, 192.4.32, and 192.4.3 can be reached along the path hAS 1, AS 2i.
  - Similarly, it could advertise, The networks 192.12.69, 192.4.54, and 192.4.23 can be reached along the path hAS 1, AS 3i.

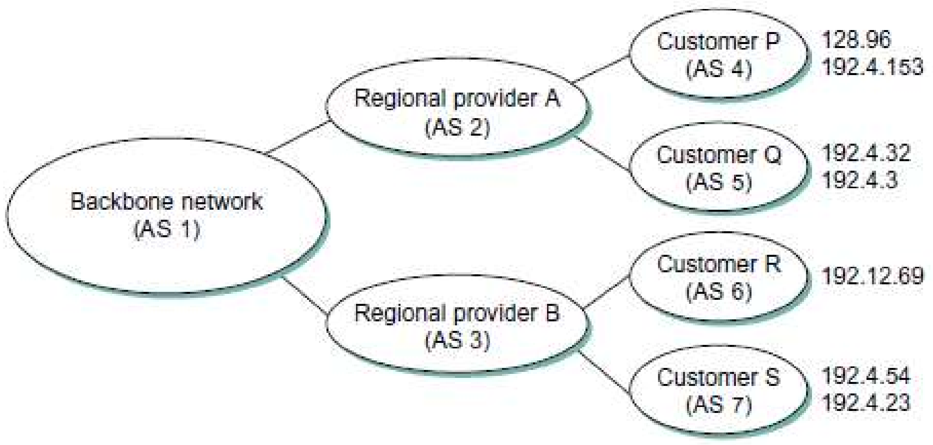


Figure 3.22 Example of a network running BGP

- BGP is to prevent the establishment of looping paths. For example, consider the network illustrated in Figure 3.23.
- It differs from Figure 3.22 only in the addition of an extra link between AS 2 and AS 3, but the effect now is that the graph of autonomous systems has a loop in it.
  - Suppose AS 1 learns that it can reach network 128.96 through AS 2, so it advertises this fact to AS 3, who in turn advertises it back to AS 2. In the absence of any loop prevention mechanism, AS 2 could now decide that AS 3 was the preferred route for packets destined for 128.96.
  - If AS 2 starts sending packets addressed to 128.96 to AS 3, AS 3 would send them to AS 1; AS 1 would send them back to AS 2; and they would loop forever.
  - The advertisement for a path to 128.96 received by AS 2 from AS 3 would contain an AS path of hAS 3, AS 1, AS 2, AS 4i. AS 2 sees itself in this path, and thus concludes that this is not a useful path for it to use.
- Given that links fail and policies change, BGP speakers need to be able to cancel previously advertised paths. This is done with a form of negative advertisement known as a *withdrawn route*.
- Both positive and negative reachability information are carried in a BGP update message, the format of which is shown in Figure 3.23.

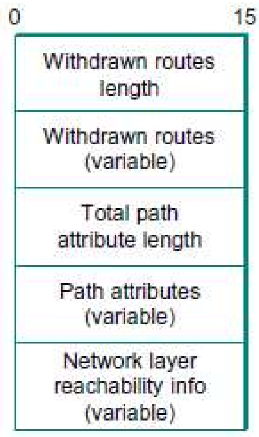

0
15
Withdrawn routes length
Withdrawn routes (variable)
Total path attribute length
Path attributes (variable)
Network layer reachability info (variable)

Figure 3.23 BGP-4 update packet format

- BGP speakers can count on TCP to be reliable, this means that any information that has been sent from one speaker to another does not need to be sent again.
- BGP speaker can simply send an occasional *keepalive* message that says, in effect, "I'm still here and nothing has changed." If that router were to crash or become disconnected from its peer, it would stop sending the keepalives, and the other routers that had learned routes from it would assume that those routes were no longer valid.

Common as Relationships and Policies

There are three relationships are illustrated in Figure 3.24.

1. *Provider-Consumer*
   - Providers are in the business of connecting their customers to the rest of the internet.
   - A customer might be a corporation, or it might be a smaller ISP.
   - So the common policy is to advertise all the routers I know about to my customer, and advertise routes I learn from my customer to everyone.

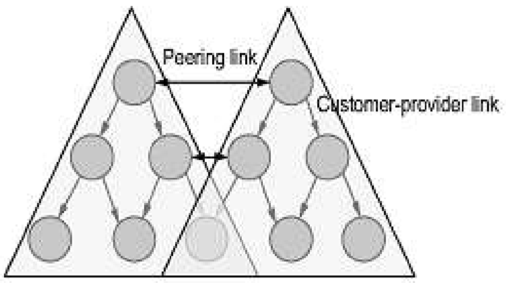


Figure 3.24 Common as relation ships

2. *Customer-Provider*
   - The customer wants to get traffic directed to him by his provider, and he wants to be able to send traffic to the rest of his provider.
   - The common policy in this case is to advertise my own prefixes and routes learned from my customers to my provider, advertise routes learned from one provider to another provider
3. *Peer*
   - Two providers who view themselves as equals usually peer so that they can get access to each other's customers without having to pay another provider.
   - The common policy in this case is to advertise routes learned from my customer to my peer, advertise routes learned from my peer to my customers, but don't advertise routes from my peer to any provider or *vice versa*.

Integrating Interdomain and Intradomain Routing

There are several ways how all the other routers in a domain get this information using BGP speaker.

- In the case of a stub AS that only connects to other autonomous systems at a single point, the border router is clearly the only choice for all routes that are outside the AS. Such a router can inject a *default route* into the intradomain routing protocol.
- The next step up in complexity is to have the border routers inject specific routes they have learned from outside the AS.
  - Consider, for example, the border router of a provider AS that connects to a customer AS. That router could learn that the network prefix 192.4.54/24 is located inside the customer AS, either through BGP or because the information is configured into the border router.
  - It could inject a route to that prefix into the routing protocol running inside the provider AS. This would be an advertisement of the sort, "I have a link to 192.4.54/24 of cost X." This would cause other routers in the provider AS to learn that this border router is the place to send packets destined for that prefix.

- The final level of complexity comes in backbone networks, which learn so much routing information from BGP that it becomes too costly to inject it into the intradomain protocol.
  - For example, if a border router wants to inject 10,000 prefixes that it learned about from another AS, it will have to send very big link-state packets to the other routers in that AS, and their shortest-path calculations are going to become very complex. For this reason, the routers in a backbone network use a variant of BGP called *interior*

3.3.3. IP Version 6 (IPv6)

An IPv6 address consists of 16 bytes (octets); it is 128 bits long.

Hexadecimal Colon Notation

- IPv6 specifies hexadecimal colon notation.
- In this notation, 128 bits is divided into eight sections, each 2 bytes in length.
- Two bytes in hexadecimal notation requires four hexadecimal digits.
- Therefore, the address consists of 32 hexadecimal digits, with every four digits separated by a colon, as shown in Figure 3.25.

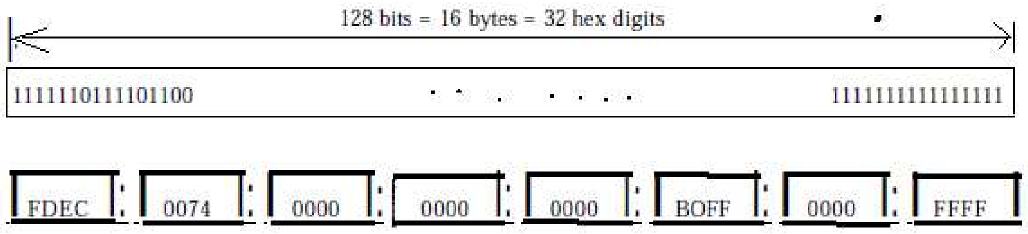


Figure 3.25 *IPv6 address in binary and hexadecimal colon notation*

Abbreviation

The leading zeros of a section (four digits between two colons) can be omitted. Only the leading zeros can be dropped, not the trailing zeros (see Figure 3.26).

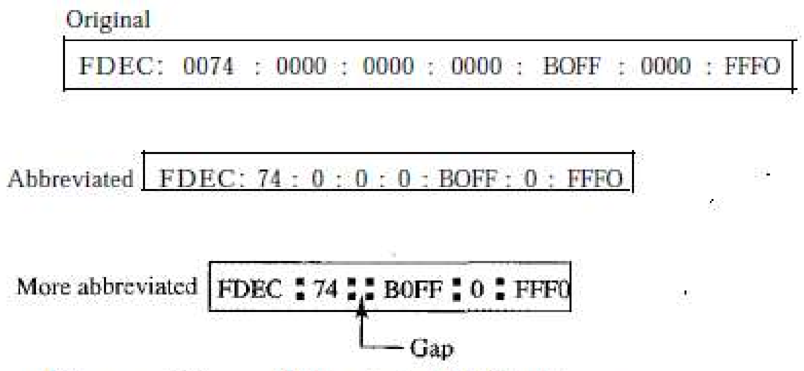

Original
FDEC: 0074 : 0000 : 0000 : 0000 : BOFF : 0000 : FFFO
Abbreviated
FDEC: 74 : 0 : 0 : 0 : BOFF : 0 : FFFO
More abbreviated
FDEC : 74 :: B0FF : 0 : FFF0
Gap

Figure 3.26 *Abbreviated IPv6 addresses*

Using this form of abbreviation, 0074 can be written as 74, 000F as F, and 0000 as 0.

Example

Expand the address 0:15::1:12:1213 to its original.

Solution

We first need to align the left side of the double colon to the left of the original pattern and the right side of the double colon to the right of the original pattern to find now many 0s we need to replace the double colon.

```
xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx
0    : 15                    :1      : 12  :1213
```

This means that the original address is

0000:0015:0000:0000:0000:0001 :0012:1213

Address Space

- IPv6 has a much larger address space; 2128 addresses are available.
- The designers of IPv6 divided the address into several categories.
- A few leftmost bits, called the *type prefix,* in each address define its category. The type prefix is variable in length, but it is designed such that no code is identical to the first part of any other code
- Table shows the prefix for each type of address. The third column shows the fraction of each type of address relative to the whole address space.

Type Prefix	*Type*	*Fraction*
00000000	Reserved	1/256
00000001	Unassigned	1/256
0000001	ISO network addresses	1/128
0000010	IPX (Novell) network addresses	1/128
0000011	Unassigned	1/128
00001	Unassigned	1/32
0001	Reserved	1/16
001	Reserved	1/8
010	Provider-based unicast addresses	1/8
011	Unassigned	1/8
100	Geographic-based unicast addresses	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1116
11110	Unassigned	1132
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
11111110 a	Unassigned	1/512
1111 111010	Link local addresses	111024
1111 1110 11	Site local addresses	1/1024
11111111	Multicast addresses	1/256

Unicast Addresses

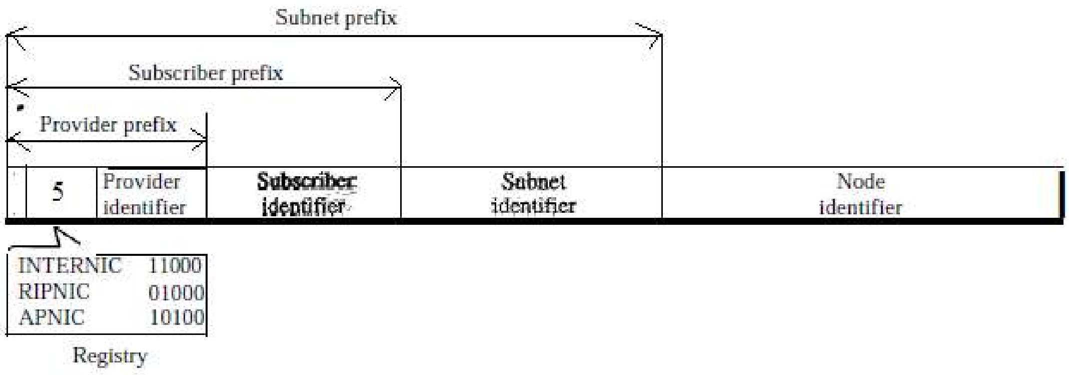


Figure 3.27 *Prefixes for provider-based unicast address*

- A *unicast address* defines a single computer.
- The packet sent to a unicast address must be delivered to that specific computer. IPv6 defines two types of unicast addresses: geographically based and provider-based. The address format is shown in Figure 3.27.
- Fields for the provider-based address are as follows:
  - **Type identifier.** This 3-bit field defines the address as a provider based address.
  - **Registry identifier.** This 5-bit field indicates the agency that has registered the address. Currently three registry centers have been defined. INTERNIC (code 11000) is the center for North America; RIPNIC (code 01000) is the center for European registration; and APNIC (code 10100) is for Asian and Pacific countries.
  - **Provider identifier.** This variable-length field identifies the provider for Internet access (such as an ISP). A 16-bit length is recommended for this field.
  - **Subscriber identifier.** When an organization subscribes to the Internet through a provider, it is assigned a subscriber identification. A 24-bit length is recommended for this field.
  - **Subnet identifier**. Each subscriber can have many different subnetworks, and each subnetwork can have an identifier. The subnet identifier defines a specific subnetwork under the territory of the subscriber. A 32-bit length is recommended for this field.
  - **Node identifier.** The last field defines the identity of the node connected to a subnet. A length of 48 bits is recommended for this field to make it compatible with the 48-bit link (physical) address used by Ethernet.

Multicast Addresses

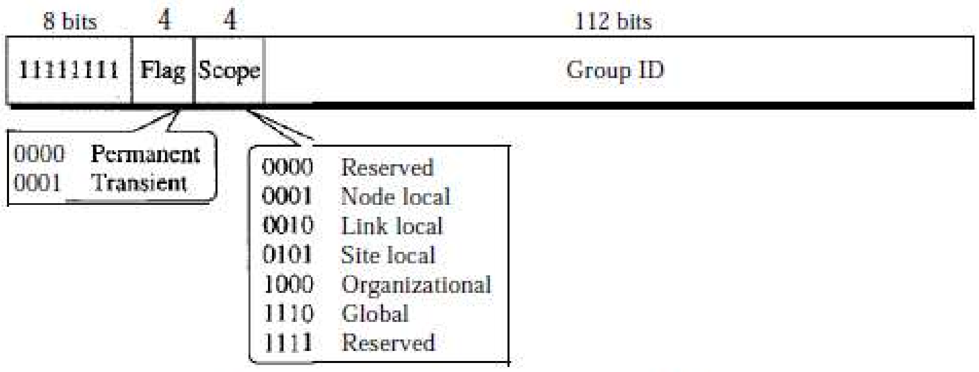


Figure 3.28 *Multicast address in IPv6*

- Multicast addresses are used to define a group of hosts instead of just one. A packet sent to a multicast address must be delivered to each member of the group. Figure 3.28 shows the format of a multicast address.
- The second field is a flag that defines the group address as either permanent or transient.
- A permanent group address is defined by the Internet authorities and can be accessed at all times.
- A transient group address, on the other hand, is used only temporarily.
- The third field defines the scope of the group address. Many different scopes have been defined, as shown in Figure 3.28.

Anycast Addresses

- IPv6 also defines anycast addresses.
- An anycast address, like a multicast address, also defines a group of nodes.
- The routers outside the ISP deliver a packet destined for the ISP to the nearest ISP router.
- No block is assigned for anycast addresses.

Reserved Addresses

- These addresses start with eight 0s (type prefix is 00000000).
- A few subcategories are defined in this category, as shown in Figure 3.29.

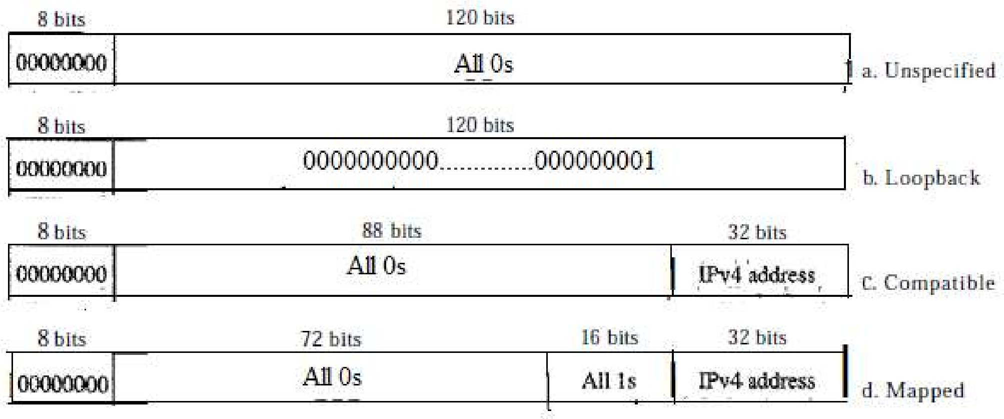


Figure 3.29 *Reserved addresses in IPv6*

  - An *unspecified address* is used when a host does not know its own address and sends an inquiry to find its address.
  - A *loopback address* is used by a host to test itself without going into the network.
  - A *compatible address* is used during the transition from IPv4 to IPv. It is used when a computer using IPv6 wants to send a message to another computer using IPv6, but the message needs to pass through a part of the network that still operates in IPv4.
  - A *mapped address* is also used during transition. It is used when a computer that has migrated to IPv6 wants to send a packet to a computer still using IPv4.

Loc.al Addresses

- It provide addressing for private networks.

- Two types of addresses are defined for this purpose, as shown in Figure 3.30.

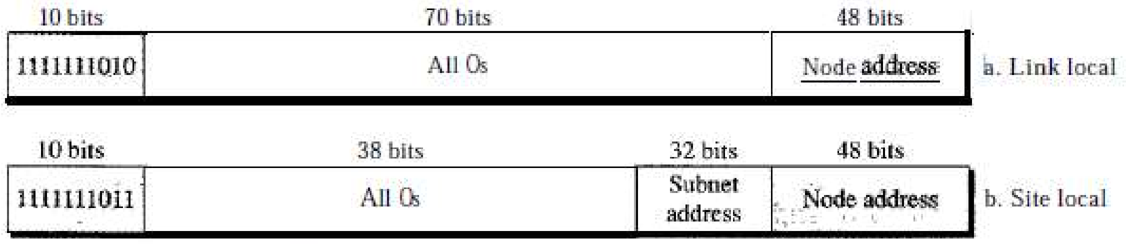


Figure 3.30 *Local addresses in IPv6*

  - A *link local address* is used in an isolated subnet
  - A *site local address* is used in an isolated site with several subnets.

Advantages

The next-generation IP, or IPv6, has some advantages over IPv4 that can be summarized as follows:

- *Larger address space*. An IPv6 address is 128 bits long Compared with the 32-bit address of IPv4, this is a huge (296) increase in the address space.
- *Better header format.* IPv6 uses a new header format in which options are separated from the base header and inserted, when needed, between the base header and the upper-layer data. This simplifies and speeds up the routing process because most of the options do not need to be checked by routers.
- *New options*. IPv6 has new options to allow for additional functionalities.
- *Allowance for extension.* IPv6 is designed to allow the extension of the protocol if required by new technologies or applications.
- Support for resource allocation. In IPv6, the type-of-service field has been removed, but a mechanism (calledjlow *label)* has been added to enable the source to request special handling of the packet. This mechanism can be used to support traffic such as real-time audio and video.
- Support for more security. The encryption and authentication options in IPv6 provide confidentiality and integrity of the packet.

Packet Format

- The IPv6 packet is shown in Figure 3.31.
- The payload consists of two parts: *optional extension headers* and *data* from an upper layer.

<u>Base Header</u>

- The base header occupies 40 bytes, whereas the extension headers and data from the upper layer contain up to 65,535 bytes of information.

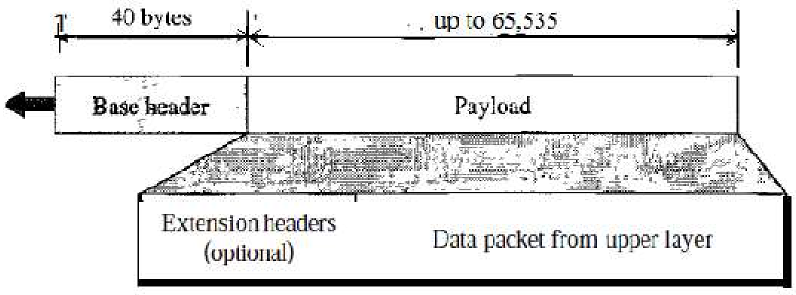


Figure 3.31 *IPv6 datagram header and payload*

- Figure 3.37 shows the base header with its eight fields. These fields are as follows:
  - *Version.* This 4-bit field defines the version number of the IP. For IPv6, the value is 6.

- *Payload length.* The 2-byte payload length field defines the length of the IP datagram excluding the base header.
- *Next header.* The next header is an 8-bit field defining the header that follows the base header in the datagram. The next header is either one of the optional extension headers used by IP or the header of an encapsulated packet such as UDP or TCP. Each extension header also contains this field. Below Table shows the values of next headers. Note that this field in version 4 is called the *protocol*

Next header codes for IPv6

Code	*Next Header*
0	Hop-by-hop option
2	ICMP
6	TCP
17	UDP
43	Source routing
44	Fragmentation
50	Encrypted security payload
51	Authentication
59	Null (no next header)
60	Destination option

- *Hop limit.* This 8-bit hop limit field serves the same purpose as the TIL field in IPv4.
- *Source address.* The source address field is a 16-byte (128-bit) Internet address that identifies the original source of the datagram.
- *Destination address.* The destination address field is a 16-byte (128-bit) Internet address that usually identifies the final destination of the datagram.
- *Priority.* The 4-bit priority field defines the priority of the packet with respect to traffic congestion.
  - *Congestion-Controlled Traffic*
    - If a source adapts itself to traffic slowdown when there is congestion, the traffic is referred to as **congestion-controlled traffic.**
    - For example, TCP, which uses the sliding window protocol, can easily respond to traffic. In congestion-controlled traffic, it is understood that packets may arrive delayed, lost, or out of order.
    - Congestion-controlled data are assigned priorities from 0 to 7, as listed in Table. A priority of 0 is the lowest; a priority of 7 is the highest.

Priorities for congestion-controlled traffic

Priority	*Meaning*
0	No specific traffic
1	Background data
2	Unattended data traffic
3	Reserved
4	Attended bulk data traffic
5	Reserved
6	Interactive traffic
7	Control traffic

  - The *priority descriptions* are as follows:
  - *No specific traffic*. A priority of 0 is assigned to a packet when the process does not define a priority.
  - *Background data.* This group (priority 1) defines data that are usually delivered in the background.
  - *Unattended data traffic.* If the user is not waiting (attending) for the data to be received, the packet will be given a priority of 2. E-mail belongs to this group. The recipient of an e-mail does not know when a message has arrived. In addition, an e-mail is usually stored before it is forwarded. A little bit of delay is of little consequence.
  - *Attended* ***bulk*** *data traffic.* A protocol that transfers data while the user is waiting (attending) to receive the data (possibly with delay) is given a priority of 4. FTP and HTTP belong to this group.

o *Interactive traffic.* Protocols such as TELNET that need user interaction are assigned the second-highest

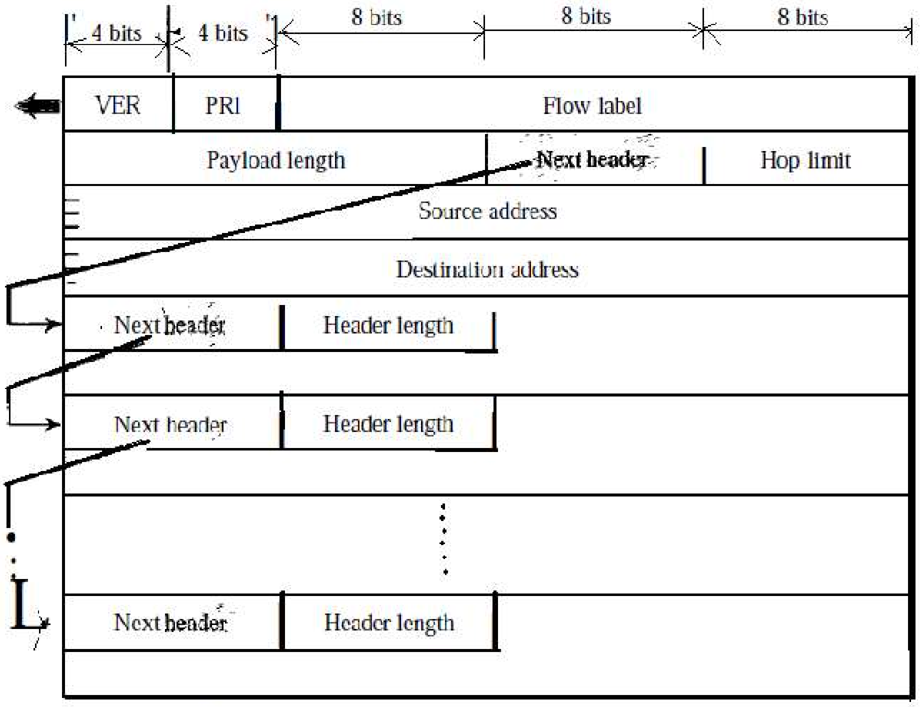


Figure 3.32 *Format of an IPv6 datagram*

priority (6) in this group.

o *Control traffic.* Control traffic is given the highest priority (7). Routing protocols such as OSPF and RIP and management protocols such as SNMP have this priority.

- *Noncongestion-Controlled Traffic*
  - This refers to a type of traffic that expects minimum delay.
  - Discarding of packets is not desirable.
  - Retransmission in most cases is impossible.
  - Priority numbers from 8 to 15 are assigned to non congestion-controlled traffic.
  - Data containing less redundancy (such as low-fidelity audio or video) can be given a higher priority (15).
  - Data containing more redundancy (such as high-fidelity audio or video) are given a lower priority (8).

Priorities for noncongestion-controlled traffic

Priority	*Meaning*
8	Data with greatest redundancy
...	...
15	Data with least redundancy

o ***Flow Label***

- The flow label is a 3-byte (24-bit) field that is designed to provide special handling for a particular flow of data.
- A sequence of packets, sent from a particular source to a particular destination, that needs special handling by routers is called a *flow* of packets.
- The combination of the source address and the value of the *flow label* uniquely define a flow of packets.
- A router that supports the handling of flow labels has a flow label table.

- The table has an entry for each active flow label; each entry defines the services required by the corresponding flow label.
- When the router receives a packet, it consults its flow label table to find the corresponding entry for the flow label value defined in the packet. It then provides the packet with the services mentioned in the entry.
- To allow the effective use of flow labels, three rules have been defined:
  i. The flow label is assigned to a packet by the source host. The label is a random number between 1 and 224 - 1. A source must not reuse a flow label for a new flow while the existing flow is still active.
  ii. If a host does not support the flow label, it sets this field to zero. If a router does not support the flow label, it simply ignores it.
  iii. All packets belonging to the same flow have the same source, same destination, same priority, and same options.

Comparison Between IPv4 and IPv6 Headers

1. The header length field is eliminated in IPv6 because the length of the header is fixed in this version.
2. The service type field is eliminated in IPv6. The priority and flow label fields together take over the function of the service type field.
3. The total length field is eliminated in IPv6 and replaced by the payload length field.
4. The identification, flag, and offset fields are eliminated from the base header in IPv6. They are included in the fragmentation extension header.
5. The TTL field is called hop limit in IPv6.
6. The protocol field is replaced by the next header field.
7. The header checksum is eliminated because the checksum is provided by upper-layer protocols; it is therefore not needed at this level.
8. The option fields in IPv4 are implemented as extension headers in IPv6.

Extension Headers

- The length of the base header is fixed at 40 bytes.
- Six types of extension headers have been defined, as shown in Figure 3.33.

i.Hop-by-Hop Option

- The hop-by-hop option is used when the source needs to pass information to all routers visited by the datagram.
- Three options have been defined:

a. The *Pad I option* is 1 byte long and is designed for alignment purposes.
b. *PadN option* is similar in concept to Pad I. The difference is that PadN is used when 2 or more bytes is needed

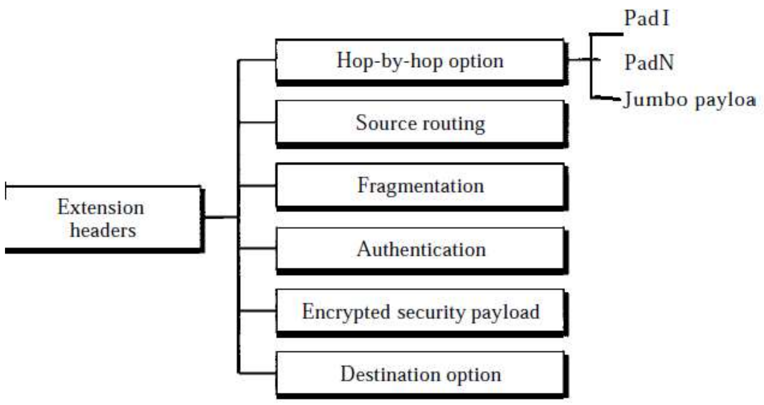


Figure 3.33 *Extension header types*

for alignment.

c. *jumbo payload option* is used to define a payload longer than 65,535 bytes.

ii.Fragmentation Only the original source can fragment. A source must use a path MTU discovery technique to find the smallest MTU supported by any network on the path.

- ***Authentication*** **:** The authentication extension header has a dual purpose: it validates the message sender and ensures the integrity of data.
- ***Encrypted Security Payload :*** The encrypted security payload (ESP) is an extension that provides confidentiality and guards against eavesdropping.
- ***Destination Option*** The destination option is used when the source needs to pass information to the destination only. Intermediate routers are not permitted access to this information.

Comparison Between IPv4 Options and IPv6 Extension Headers

1. The no-operation and end-of-option options in IPv4 are replaced by PadI and PadN options in IPv6.
2. The record route option is not implemented in IPv6 because it was not used.
3. The timestamp option is not implemented because it was not used.
4. The source route option is called the source route extension header in IPv6.
5. The fragmentation fields in the base header section of IPv4 have moved to the fragmentation extension header in IPv6.
6. The authentication extension header is new in IPv6.
7. The encrypted security payload extension header is new in IPv6.

3.4. Multicast

- In multicast communication, there is one source and a group of destinations.
- The relationship is one-to-many.
- In this type of communication, the source address is a unicast address, but the destination address is a group address, which defines one or more destinations.
- The group address identifies the members of the group.
- Figure 3.96 shows the idea behind multicasting.
- A multicast packet starts from the source S1 and goes to all destinations that belong to group G1.
- In multicasting, when a router receives a packet, it may forward it through several of its interfaces.
- In multicasting, the router may forward the received packet through several of its interfaces.

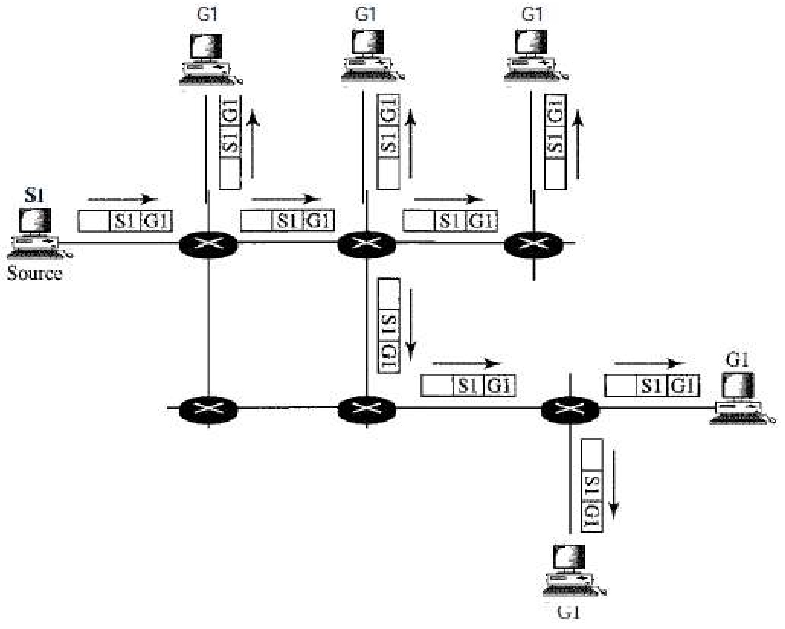


Figure 3.34 *Multicasting*

3.4.1. Addresses

- IP has a subrange of its address space reserved for multicast addresses.
- In IPv4, these addresses are assigned in the class D address space, and IPv6 also has a portion of its address space reserved for multicast group addresses.
- IP has to map 28-bit IP multicast addresses into 23-bit Ethernet multicast addresses.

3.4.2. Multicast routing

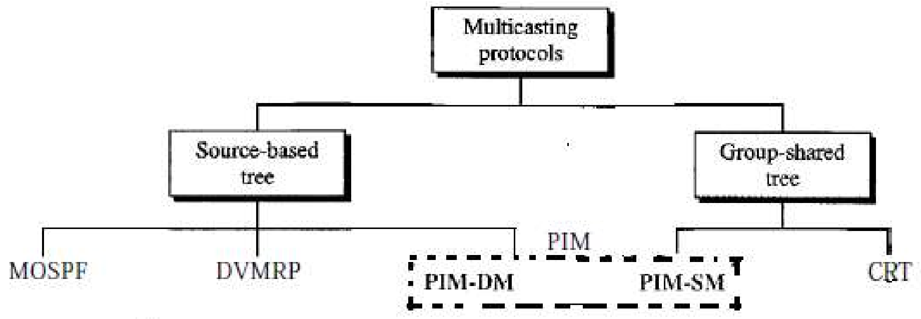
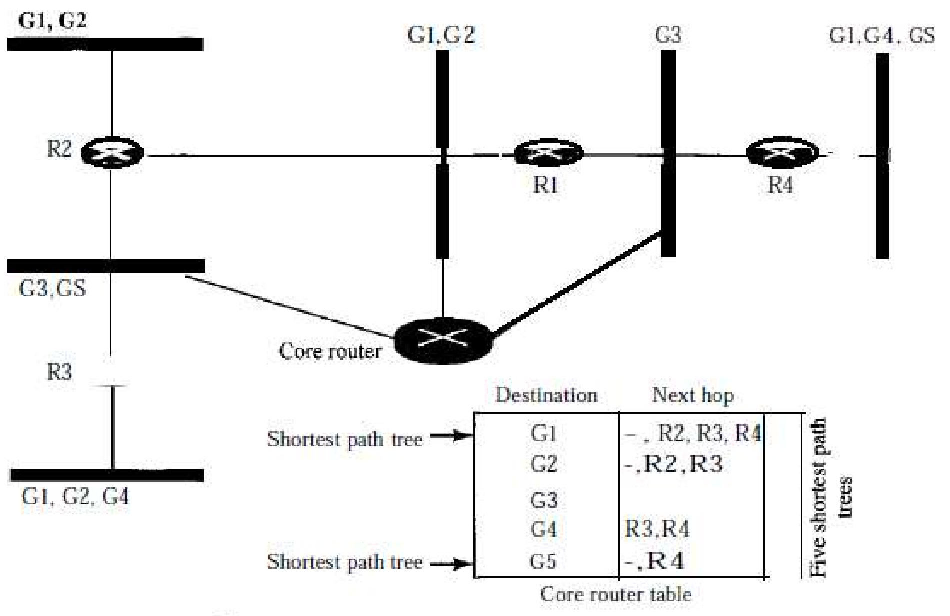


Figure 3.35 *Taxonomy of common multicast protocols*

Optimal Routing: Shortest Path Trees

- The process of optimal interdomain routing eventually results in the finding of the *shortest path tree.*
- The root of the tree is the source, and the leaves are the potential destinations. The path from the root to each destination is the shortest path.
- A multicast packet may have destinations in more than one network. Forwarding of a single packet to members of a group requires a shortest path tree.
- If *n* groups, we may need *n* shortest path trees.
- Two approaches have been used to solve the problem: *source-based trees* and *group-shared trees.*
  - *Source-Based Tree.*
    - ✓ In the source-based tree approach, each router needs to have one shortest path tree for each group.
    - ✓ The shortest path tree for a group defines the next hop for each network that has loyal member(s) for that



Destination	Next hop
G1	-, R2, R3, R4
G2	-,R2,R3
G3	
G4	R3,R4
G5	-,R4

Figure 3.37 *Group-shared tree approach*

group. In Figure 3.35, we assume that we have only five groups in the domain: GI, G2, G3, G4, and G5.

- ✓ At the moment GI has loyal members in four networks, G2 in three, G3 in two, G4 in two, and G5 in two.
- ✓ Figure 3.35 also shows the multicast routing table for router RI. There is one shortest path tree for each group; therefore there are five shortest path trees for five groups.
- ✓ If router Rl receives a packet with destination address G1, it needs to send a copy of the packet to the attached network, a copy to router R2, and a copy to router R4 so that all members of G1 can receive a copy.
- ✓ In this approach, if the number of groups is *m,* each router needs to have *m* shortest path trees, one for each group.

➢ *Group-Shared Tree*

- ✓ In the group-shared tree approach, instead of each router having *m* shortest path trees, only one designated router, called the center core, or rendezvous router, takes the responsibility of distributing multicast traffic.
- ✓ The core has *m* shortest path trees in its routing table. The rest of the routers in the domain have none.
- ✓ If a router receives a multicast packet, it encapsulates the packet in a unicast packet and sends it to the core router. The core router removes the multicast packet from its capsule, and consults its routing table to route the packet.

3.4.2.1. *Multicast Link State Routing: MOSPF*

- In Multicast Link State Routing each router creates a shortest path tree by using Dijkstra's algorithm. The routing table is a translation of the shortest path tree.
- Multicast link state routing uses the source-based tree approach.
- For multicast routing, a node needs to revise the interpretation of *state.* A node advertises every group which has any loyal member on the link.
- The information about the group comes from IGMP
- Each router running IGMP solicits the hosts on the link to find out the membership status.
- When a router receives all these LSPs, it creates *n (n* is the number of groups) topologies, from which *n* shortest path trees are made by using Dijkstra's algorithm. So each router has a routing table that represents as many shortest path trees as there are groups.
- The only problem with this protocol is the time and space needed to create and save the many shortest path trees. The solution is to create the trees only when needed.
- When a router receives a packet with a multicast destination address, it runs the Dijkstra algorithm to calculate the shortest path tree for that group. The result can be cached in case there are additional packets for that destination.

MOSPF

- Multicast Open Shortest Path First (MOSPF) protocol is an extension of the OSPF protocol that uses multicast link state routing to create source-based trees.
- The protocol requires a new link state update packet to associate the unicast address of a host with the group address or addresses the host is sponsoring. This packet is called the group-membership LSA.
- For efficiency, the router calculates the shortest path trees on demand (when it receives the first multicast packet).

- MOSPF is a data-driven protocol; the first time an MOSPF router sees a datagram with a given source and group address, the router constructs the Dijkstra shortest path tree.

Multicast Distance Vector: DVMRP

- Multicast routing does not allow a router to send its routing table to its neighbors. The idea is to create a table from scratch by using the information from the unicast distance vector tables.
- Multicast distance vector routing uses source-based trees, but the router never actually makes a routing table.
- When a router receives a multicast packet, it forwards the packet as though it is consulting a routing table.
- After its use (after a packet is forwarded) the table is destroyed.
- The multicast distance vector algorithm uses a process based on four decision-making strategies. Each strategy is built on its predecessor.

i. *Flooding.*

✓ A router receives a packet and, without even looking at the destination group address, sends it out from every interface except the one from which it was received.

✓ Every network with active members receives the packet. However, so will networks without active members. This is a broadcast, not a multicast.

✓ There is another problem: it creates loops. A packet that has left the router may come back again from another interlace or the same interlace and be forwarded again.

✓ Some flooding protocols keep a copy of the packet for a while and discard any duplicates to avoid loops. The next strategy, reverse path forwarding, corrects this defect.

ii. *Reverse Path Forwarding (RPF).*

✓ To prevent loops of flooding , only one copy is forwarded; the other copies are dropped.

✓ In RPF, a router forwards only the copy that has traveled the shortest path from the source to the router. To find this copy, RPF uses the unicast routing table.

✓ The router receives a packet and extracts the source address (a unicast address). It consults its unicast routing table as though it wants to send a packet to the source address.

✓ The routing table tells the router the next hop. If the multicast packet has just come from the hop defined in the table, the packet has traveled the shortest path from the source to the router because the shortest path is reciprocal in unicast distance vector routing protocols.

✓ If the path from A to B is the shortest, then it is also the shortest from B to A.

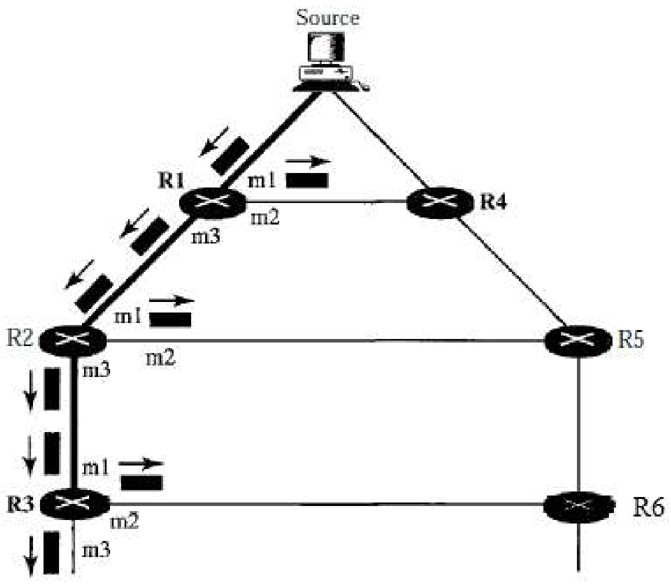


Figure 3.38 *Reverse path forwarding (RPF)*

The router forwards the packet if it has traveled from the shortest path; it discards it otherwise.

- ✓ Figure 3.38 shows part of a domain and a source. The shortest path tree as calculated by routers RI, R2, and R3 is shown by a thick line.
- ✓ When R1 receives a packet from the source through the interface rnl, it consults its routing table and finds that the shortest path from RI to the source is through interface mI. The packet is forwarded. However, if a copy of the packet has arrived through interface m2, it is discarded because m2 does not define the shortest path from RI to the source. This is the same with R2 and R3.
- ✓ what happens if a copy of a packet that arrives at the ml interface of R3, travels through R6, R5, R2, and then enters R3 through interface ml. This interface is the correct interface for R3. Is the copy of the packet forwarded?

The answer is that this scenario never happens because when the packet goes from R5 to R2, it will be discarded by R2 and never reaches R3. The upstream routers toward the source always discard a packet that has not gone through the shortest path, thus preventing confusion for the downstream routers.

iii. *Reverse Path Broadcasting (RPB).*

- RPF guarantees that each network receives a copy of the multicast packet without formation of loops.
- RPF does not guarantee that each network receives only one copy; a network may receive two or more copies. The reason is that RPF is not based on the destination address (a group address); forwarding is based on the source address.
- To visualize the problem, let us look at Figure 3.39.
- Net3 in this figure receives two copies of the packet even though each router just sends out one copy from each interface. There is duplication because a tree has not been made; instead of a tree we have a graph. Net3 has two parents: routers R2 andR4.
- To eliminate duplication, we must define only one parent router for each network.
- The restriction is: A network can receive a multicast packet from a particular source only through a designated parent router.
- For each source, the router sends the packet only out of those interfaces for which it is the designated parent. This policy is called reverse path broadcasting (RPB). RPB guarantees that the packet reaches every network and that every network receives only one copy.
- Figure 3.40 shows the difference between RPF and RPB.

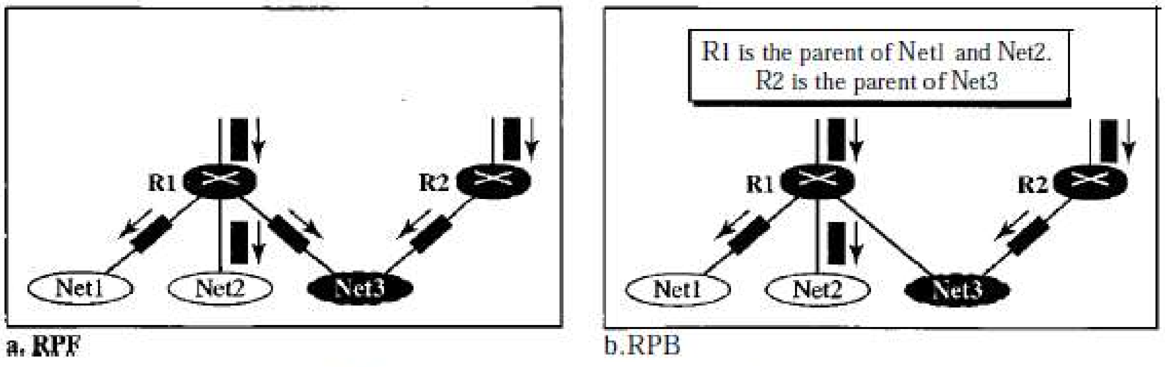


Figure 3.40 *RPF Versus RPB*

- The reader may ask how the designated parent is determined. The designated parent router can be the router with the shortest path to the source. Because routers periodically send updating packets to each other (in RIP), they can easily determine which router in the neighborhood has the shortest path to the source (when interpreting the source as the destination),

- If more than one router qualifies, the router with the smallest IP address is selected.
- RPB creates a shortest path broadcast tree from the source to each destination. It guarantees that each destination receives one and only one copy of the packet.

iv. *Reverse Path Multicasting (RPM).*

- To increase efficiency, the multicast packet must reach only those networks that have active members for that particular group. This is called reverse path multicasting (RPM).
- To convert broadcasting to multicasting, the protocol uses two procedures, pruning and grafting.
- Figure 3.41 shows the idea of pruning and grafting.

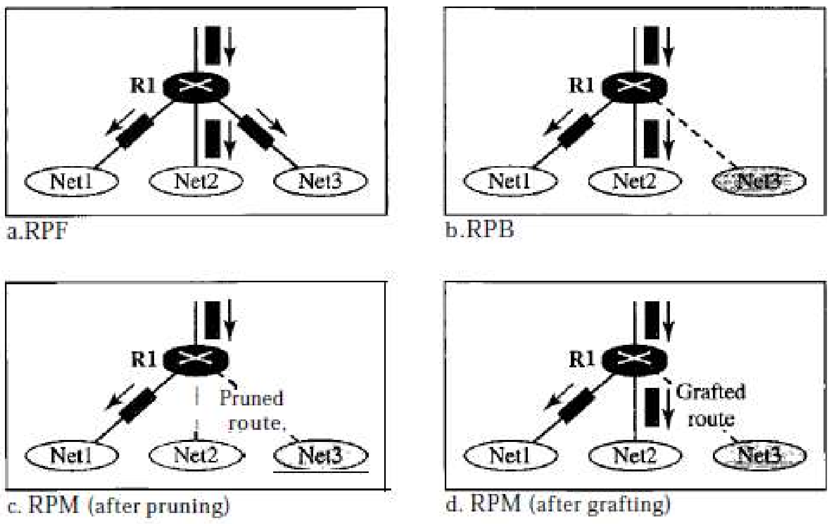


Figure 3.41 *RPF, RPB, and RPM*

- The designated parent router of each network is responsible for holding the membership information.
- The process starts when a router connected to a network finds that there is no interest in a multicast packet. The router sends a prune message to the upstream router so that it can exclude the corresponding interface. That is, the upstream router can stop sending multicast messages for this group through that interface. Now if this router receives prune messages from all downstream routers, it, in turn, sends a prune message to its upstream router.
- What if a leaf router (a router at the bottom of the tree) has sent a prune message but suddenly realizes, through IGMP, that one of its networks is again interested in receiving the multicast packet? It can send a graft message.
- The graft message forces the upstream router to resume sending the multicast messages. RPM adds pruning and grafting to RPB to create a multicast shortest path tree that supports dynamic membership changes.

3.4.2.2. DVMRP

- The Distance Vector Multicast Routing Protocol (DVMRP) is an implementation of multicast distance vector routing. It is a source-based routing protocol, based on RIP.

CBT

- The Core-Based Tree (CBT) protocol is a group-shared protocol that uses a core as the root of the tree. The autonomous system is divided into regions, and a core (center router or rendezvous router) is chosen for each region.

Formation of the Tree

✓ After the rendezvous point is selected, every router is informed of the unicast address of the selected router. Each router then sends a unicast join message similar to a grafting message) to show that it wants to join the group. This message passes through all routers that are located between the sender and the rendezvous router.

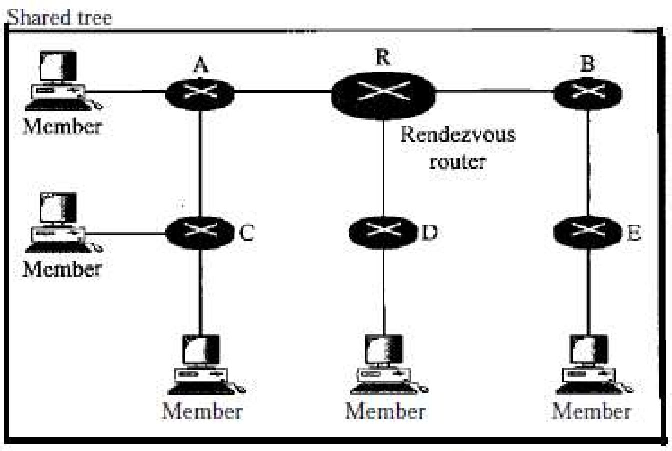


Figure 3.42 *Group-shared tree with rendezvous router*

✓ Each intermediate router extracts the necessary infonnation from the message, such as the unicast address of the sender and the interface through which the packet has arrived, and forwards the message to the next router in the path.

✓ When the rendezvous router has received all join messages from every member of the group, the tree is formed. Now every router knows its upstream router (the router that leads to the root) and the downstream router (the

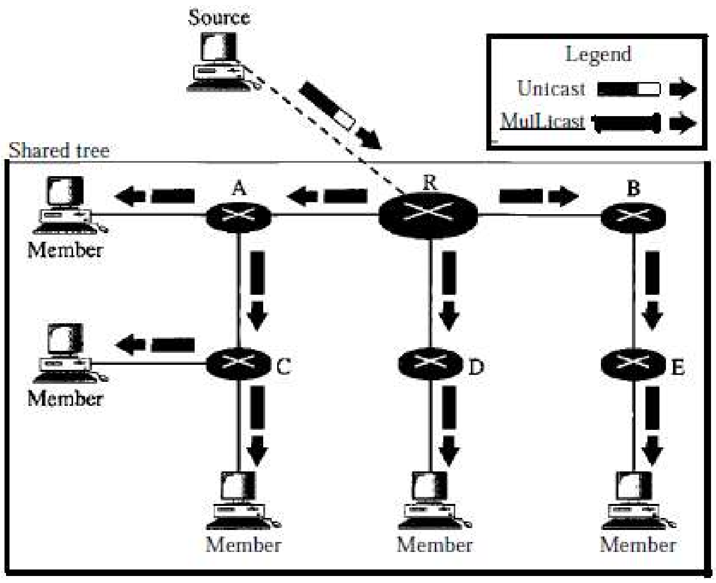


Figure 3.43 *Sending a multicast packet to the rendezvous router*

router that leads to the leaf).

- ✓ If a router wants to leave the group, it sends a leave message to its upstream router. The upstream router removes the link to that router from the tree and forwards the message to its upstream router, and so on.
- ✓ Figure 3.42 shows a group-shared tree with its rendezvous router.

Sending Multicast Packets

- ✓ After formation of the tree, any source (belonging to the group or not) can send a multicast packet to all members of the group. It simply sends the packet to the rendezvous router, using the unicast address of the rendezvous router; the rendezvous router distributes the packet to all members of the group. Figure 3.105 shows how a host can send a multicast packet to all members of the group.
- ✓ Note that the source host can be any of the hosts inside the shared tree or any host outside the shared tree. In Figure 3.43we show one located outside the shared tree.

Core-Based Tree (CBT) is a group-shared tree, center-based protocol using one tree per group. One of the routers in the tree is called the core. A packet is sent from the source to members of the group following this procedure:

1. The source, which mayor may not be part of the tree, encapsulates the multicast packet inside a unicast packet with the unicast destination address of the core and sends it to the core. This part of delivery is done using a unicast address; the only recipient is the core router.
2. The core decapsulates the unicast packet and forwards it to all interested intetfaces.
3. Each router that receives the multicast packet, in tum, forwards it to all interested interfaces.

3.4.2.3. PIM

Protocol Independent Multicast (PIM) is the name given to two independent multicast routing protocols: *Protocol Independent Multicast, Dense Mode (PIM-DM) and Protocol Independent Multicast, Sparse Mode (PIM-SM).*

Protocol Independent Multicast, Dense Mode (PIM-DM)

- ✓ PIM-DM is used when there is a possibility that each router is involved in multicasting (dense mode). In this environment, the use of a protocol that broadcasts the packet is justified because almost all routers are involved in the process.
- ✓ PIM-DM is used in a dense multicast environment, such as a LAN.
- ✓ PIM-DM is a source-based tree routing protocol that uses RPF and pruning and grafting strategies for multicasting.
- ✓ Its operation is like that of DVMRP; however, unlike DVMRP, it does not depend on a specific unicasting protocol.
- ✓ It assumes that the autonomous system is using a unicast protocol and each router has a table that can find the outgoing interface that has an optimal path to a destination. This unicast protocol can be a distance vector protocol (RIP) or link state protocol (OSPF).
- ✓ PIM-DM uses RPF and pruning and grafting strategies to handle multicasting.

PIM-SM

- ✓ PIM-SM is used when there is a slight possibility that each router is involved in multicasting (sparse mode). In this environment, the use of a protocol that broadcasts the packet is not justified; a protocol such as CBT that uses a group-shared tree is more appropriate.
- ✓ PIM-SM is used in a sparse multicast environment such as a WAN.
- ✓ PIM-SM is a group-shared tree routing protocol that has a rendezvous point (RP) as the source of the tree.

- ✓ Its operation is like CBT; however, it is simpler because it does not require acknowledgment from a join message.
- ✓ It creates a backup set of RPs for each region to cover RP failures.
- ✓ One of the characteristics of PIM-SM is that it can switch from a group-shared tree strategy to a source-based tree strategy when necessary. This can happen if there is a dense area of activity far from the RP. That area can be more efficiently handled with a source-based tree strategy instead of a group-shared tree strategy.
- ✓ PIM-SM is similar to CBT but uses a simpler procedure.

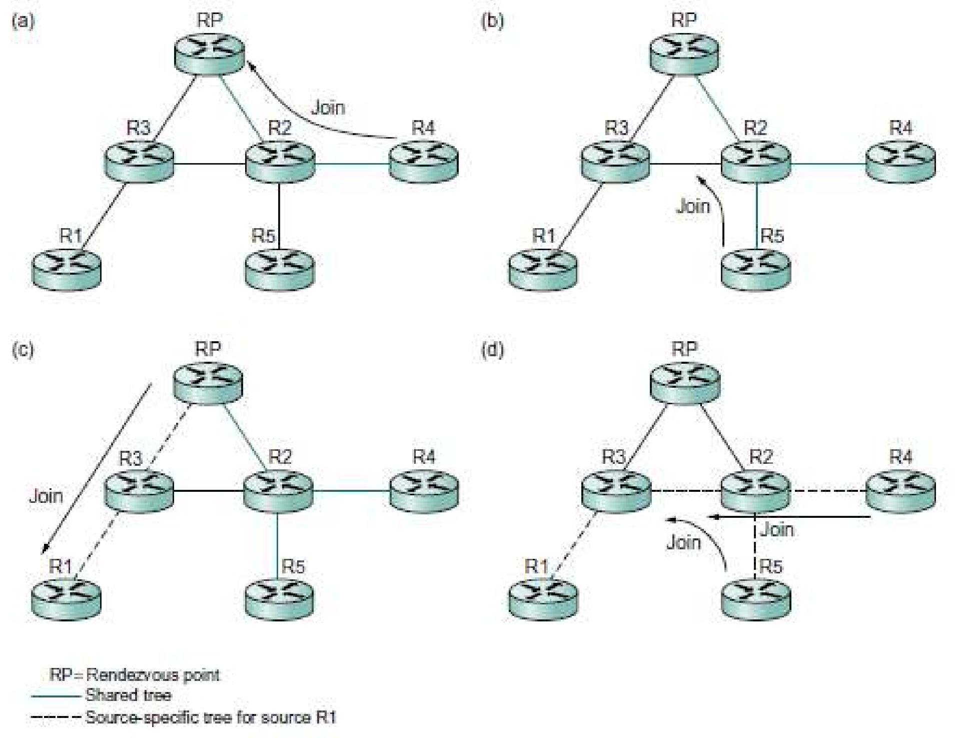


Figure 3.44 PIM operation: (a)R4 sends join to RIP and joins shared tree, (b) R5 joins shared tree; (c)RIP builds source-specific tree to R1 by sending join to R1 (d)R4 and R5 build source-soecific tree to R1 by sending joins to R1