3.1 Introduction

- A routing table can he either static or dynamic.
- A static table is one with manual entries A dynamic table, on the other hand, is one that is updated automatically when there is a change somewhere on the Internet.
- Today, an internet needs dynamic routing tables. The tables need to be updated as soon as there is a change in the internet. For instance, they need to be updated when a router is down, and they need to be updated whenever a better route has been found.
- Routing protocols have been created in response to the demand for dynamic routing tables.
- A routing protocol is a combination of rules and procedures that lets routers on the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighbourhood. The sharing of information allows a router in San Francisco to know about the failure of a network in Texas. The routing protocols also include procedures for combining information received from other routers.

3.1.1 Optimization

- A router receives a packet from a network and passes it to another network.
- A router is usually attached to several networks.
- When it receives a packet, to which network should it pass the packet? The decision is based on optimization: Which of the available pathways is the optimum pathway? What is the definition of the term optimum?
- One approach is to assign a cost for passing through a network. We call this cost a metric.
- However, the metric assigned to each network depends on the type of protocol.
- Some simple protocols,
  - **Routing Information Protocol (RIP)**
    - It treats all networks as equals.
    - The cost of passing through a network is the same; it is one hop count. So if a packet passes through 10 networks to reach the destination, the total cost is 10 hop counts.
  - **Open Shortest Path First (OSPF)**
    - It allows the administrator to assign a cost for passing through a network based on the type of service required.
    - A route through a network can have different costs (metrics).
    - For example, if maximum throughput is the desired type of service, a satellite link has a lower metric than a fiber optic line. On the other hand, if minimum delay is the desired type of service, a fiber-optic line has a lower metric than a satellite link. Routers use routing tables to help decide the best route.
    - OSPF protocol allows each router to have several routing tables based on the required type of service.
  - **Border Gateway Protocol (BGP)**
    - It allows administrator to set the policy based on the different criterion.
    - The policy defines what paths should be chosen.

0

3.2 Intra- and Interdomain Routing

- Today, an internet can be so large that one routing protocol cannot handle the task of updating the routing tables of all routers. For this reason, an internet is divided into autonomous systems.
- An autonomous system (AS) is a group of networks and routers under the authority of a single administration.
- Routing inside an autonomous system is referred to as intradomain 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. However, only one interdomain routing protocol handles routing between autonomous systems as shown in the following figure 3.2.1.

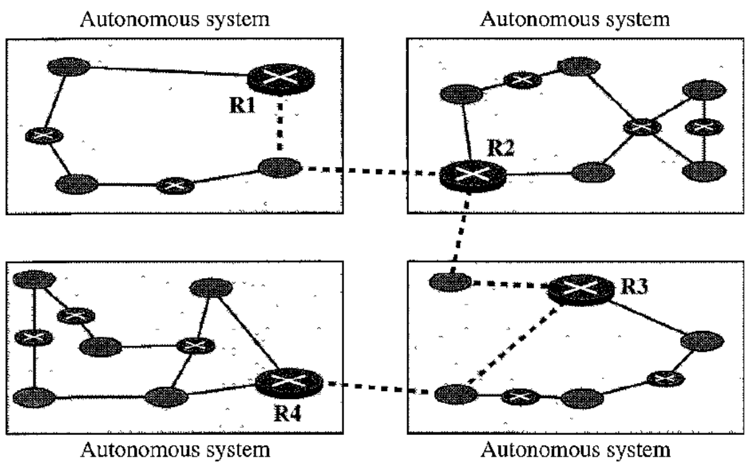


Figure 3.2.1 Autonomous Systems

- Several intradomain and interdomain routing protocols are in use.
- In this section, we cover only the most popular ones. We discuss two intradomain routing protocols: **distance vector** and **link state.** We also introduce one interdomain routing protocol: **path vector**. As shown in the figure 3.2.2.
- **Routing Information Protocol (RIP)** is an implementation of the distance vector protocol.
- **Open Shortest Path First (OSPF)** is an implementation of the link state protocol.
- **Border Gateway Protocol (BGP)** is an implementation of the path vector protocol.

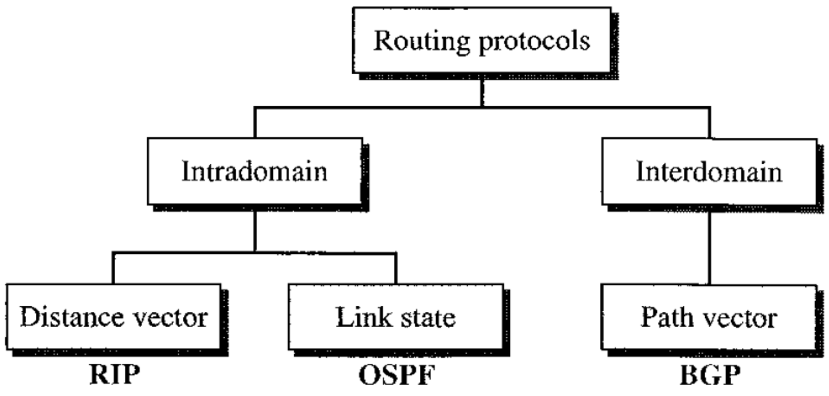


Figure 3.2.2 Popular Routing Protocols.

3.3 General Idea of Unicast Routing

- Unicast routing on the Internet, with a large number of routers and a huge number of hosts, can be done only by using hierarchical routing: routing in several steps using different routing algorithms.
- In unicast routing, a packet is routed, hop by hop, from its source to its destination by the help of forwarding tables.
- The source host needs no forwarding table because it delivers its packet to the default router in its local network.
- The destination host needs no forwarding table either because it receives the packet from its default router in its local network.
- This means that only the routers that glue together the networks in the internet need forwarding tables.
- With the above explanation, routing a packet from its source to its destination means routing the packet from a source router (the default router of the source host) to a destination router (the router connected to the destination network).
- Although a packet needs to visit the source and the destination routers, the question is what other routers the packet should visit. In other words, there are several routes that a packet can travel from the source to the destination; what must be determined is which route the packet should take.

An internet as a graph

- To find the best route, an internet can be modelled as a graph. A graph in computer science is a set of nodes and edges (lines) that connect the nodes.
- To model an internet as a graph, we can think of each router as a node and each network between a pair of routers as an edge.
- An internet is, in fact, modelled as a weighted graph, in which each edge is associated with a cost. If a weighted graph is used to represent a geographical area, the nodes can be cities and the edges can be roads connecting the cities; the weights, in this case, are distances between cities.
- In routing, however, the cost of an edge has a different interpretation in different routing protocols, which we discuss in a later section. For the moment, we assume that there is a cost associated with each edge. If there is no edge between the nodes, the cost is infinity.
- Figure 3.1.2.1 shows how an internet can be modeled as a graph.

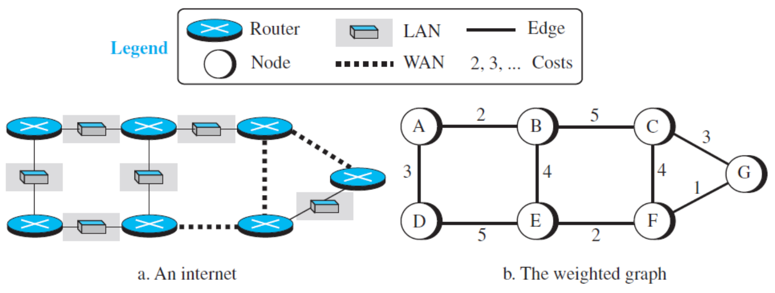


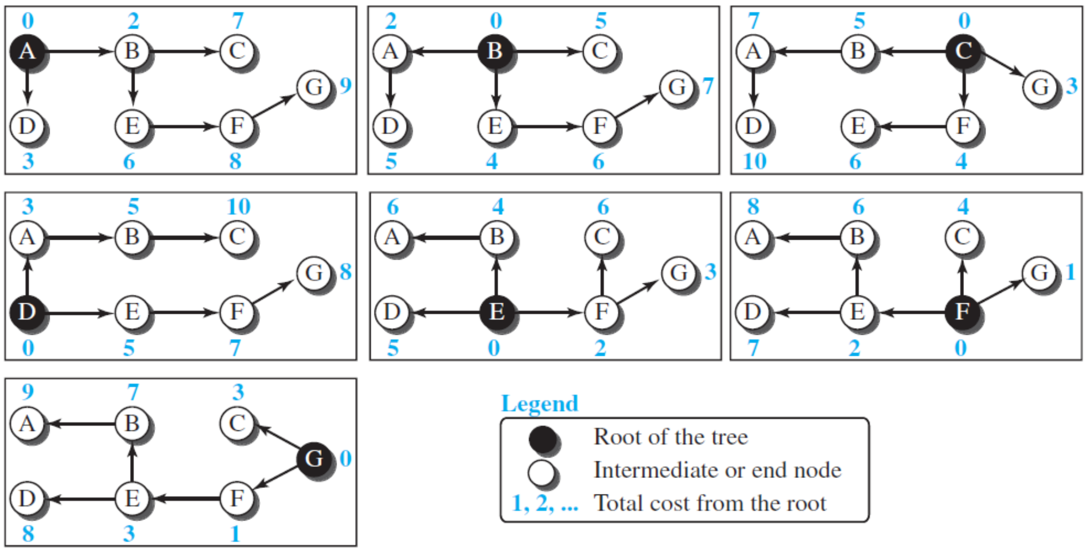
Figure 3.3.1 An internet and its graphical representation

3.3.1 Least-cost Routing

- When an internet is modelled as a weighted graph, one of the ways to interpret the best route from the source router to the destination router is to find the least cost between the two.
- In other words, the source router chooses a route to the destination router in such a way that the total cost for the route is the least cost among all possible routes.
- In Figure 3.3.1, the best route between A and E is A-B-E, with the cost of 6. This means that each router needs to find the least-cost route between itself and all the other routers to be able to route a packet using this criteria.

Least-cost trees

- If there are N routers in an internet, there are (N − 1) least-cost paths from each router to any other router. This means we need N × (N − 1) least-cost paths for the whole internet.
- If we have only 10 routers in an internet, we need 90 least-cost paths.
- A better way to see all of these paths is to combine them in a least-cost tree. A least-cost tree is a tree with the source router as the root that spans the whole graph (visits all other nodes) and in which the path between the root and any other node is the shortest.
- In this way, we can have only one shortest-path tree for each node; we have N least-cost trees for the whole internet.
- We show how to create a least-cost tree for each node later in this section; for the moment, Figure 3.3.1.1 shows the seven least-cost trees for the internet in Figure 3.3.1.



Legend
● Root of the tree
○ Intermediate or end node
1, 2, ... Total cost from the root

Figure 3.3.1.1 Least-cost trees for nodes in the internet of Figure 3.3.1

- The least-cost trees for a weighted graph can have several properties if they are created using consistent criteria.
    1. The least-cost route from X to Y in X's tree is the inverse of the least-cost route from Y to X in Y's tree; the cost in both directions is the same. For example, in Figure 3.3.1.1, the route from A to F in A's tree is (A → B → E → F), but the route from F to A in F's tree is (F → E → B → A), which is the inverse of the first route. The cost is 8 in each case.

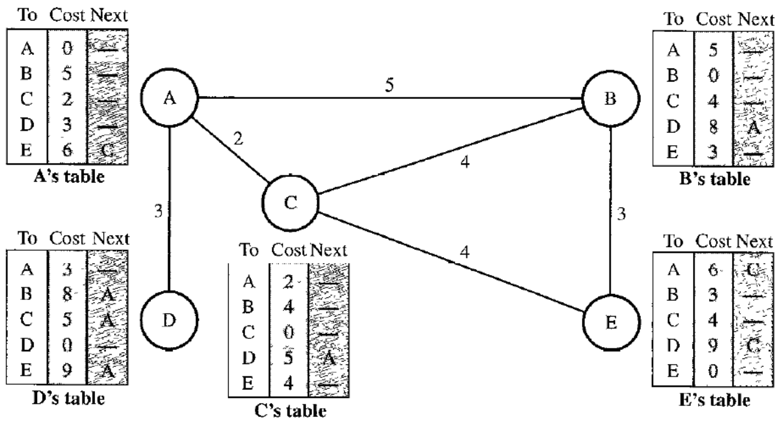
2. Instead of travelling from X to Z using X's tree, we can travel from X to Y using X's tree and continue from Y to Z using Y's tree. For example, in Figure 3.3.1.1, we can go from A to G in A's tree using the route (A → B → E → F → G). We can also go from A to E in A's tree (A → B → E) and then continue in E's tree using the route (E → F → G). The combination of the two routes in the second case is the same route as in the first case. The cost in the first case is 9; the cost in the second case is also 9 (6 + 3).

3.4 Routing Algorithms

- After discussing the general idea behind least-cost trees and the forwarding tables that can be made from them, now we concentrate on the routing algorithms.
- Several routing algorithms have been designed in the past. The differences between these methods are in the way they interpret the least cost and the way they create the least-cost tree for each node.
- In this section, we discuss the common algorithms; later we show how protocol in the Internet implements one of these algorithms.

3.4.1 Distance-Vector Routing

- In distance vector routing, the least-cost route between any two nodes is the route with minimum distance. In this protocol, as the name implies, each node maintains a vector (table) of minimum distances to every node. The table at each node also guides the packets to the desired node by showing the next stop in the route (next-hop routing). We can think of nodes as the cities in an area and the lines as the roads connecting them. ,A. table can show a tourist the minimum distance between cities.
- In Figure 3.4.1, we show a system of five nodes with their corresponding tables.



A's table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	6	C

B's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	8	A
E	3	—

D's table

To	Cost	Next
A	3	—
B	8	A
C	5	A
D	0	—
E	9	A

C's table

To	Cost	Next
A	2	—
B	4	—
C	0	—
D	5	A
E	4	—

E's table

To	Cost	Next
A	6	C
B	3	—
C	4	—
D	9	C
E	0	—

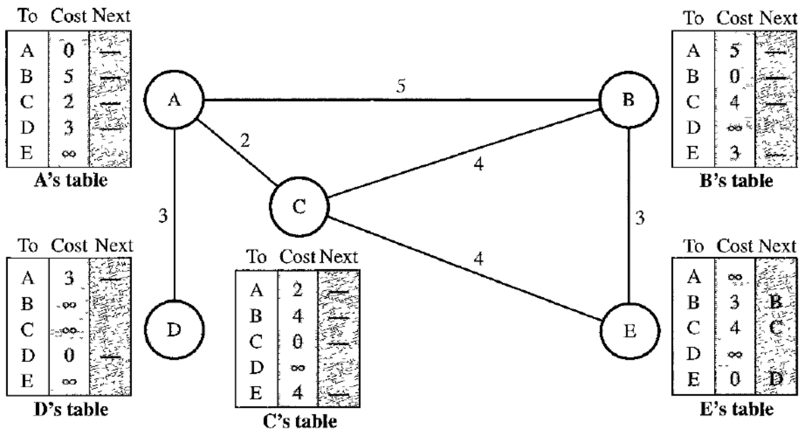
Figure 3.4.1 Distance Vector Routing Tables

- The table for node A shows how we can reach any node from this node. For example, our least cost to reach node E is 6. The route passes through C.

Initialization

- The tables in Figure 3.4.1 are stable; each node knows how to reach any other node and the cost.

- At the beginning, however, this is not the case. Each node can know only the distance between itself and its immediate neighbors, those directly connected to it.
- So, for the moment, we assume that each node can send a message to the immediate neighbors and find the distance between itself and these neighbors.
- Figure 3.4.2 shows the initial tables for each node.
- The distance for any entry that is not a neighbor is marked as infinite (unreachable).



A's table

To	Cost	Next
A	0	—
B	5	—
C	2	—
D	3	—
E	∞	

B's table

To	Cost	Next
A	5	—
B	0	—
C	4	—
D	∞	
E	3	—

D's table

To	Cost	Next
A	3	—
B	∞	
C	∞	
D	0	—
E	∞	

C's table

To	Cost	Next
A	2	—
B	4	—
C	0	—
D	∞	
E	4	—

E's table

To	Cost	Next
A	∞	
B	3	B
C	4	C
D	∞	
E	0	D

Figure 3.4.2 Initialization of tables in distance vector routing

Sharing

- The whole idea of distance vector routing is the sharing of information between neighhors.
- Although node A does not know about node E, node C does. So, if node C shares its routing table with A, node A can also know how to reach node E.
- On the other hand, node C does not know how to reach node D, but node A does. If node A shares its routing table with node C, node C also knows how to reach node D. In other words, nodes A and C, as immediate neighbors, can improve their routing tables if they help each other.
- There is only one problem. How much of the table must be shared with each neighbor? A node is not aware of a neighbor's table.
- The best solution for each node is to send its entire table to the neighbor and let the neighbor decide what part to use and what part to discard. However, the third column of a table (next stop) is not useful for the neighbor. When the neighbor receives a table, this column needs to be replaced with the sender's name. If any of the rows can be used, the next node is the sender of the table. A node therefore can send only the first two columns of its table to any neighbor. In other words, sharing here means sharing only the first two columns.

Updating

- When a node receives a two-column table from a neighbor, it needs to update its routing table. Updating takes three steps:
    1. The receiving node needs to add the cost between itself and the sending node to each value in the second column. The logic is clear. If node C claims that its distance to a destination is x mi, and the distance between A and C is y mi, then the distance between A and that destination, via C, is x + y mi.

2. The receiving node needs to add the name of the sending node to each row as the third column if the receiving node uses information from any row. The sending node is the next node in the route.
3. The receiving node needs to compare each row of its old table with the corresponding row of the modified version of the received table.
   a. If the next-node entry is different, the receiving node chooses the row with the smaller cost. If there is tie, the old one is kept.
   b. If the next-node entry is the same, the receiving node chooses the new row. For example, suppose node C has previously advertised a route to node X with distance 3. Suppose that now there is no path between C and X; node C now advertises this route with a distance of infinity. Node A must not ignore this value even though its old entry is smaller. The old route does not exist any more. The new route has a distance of infinity.

- Figure 3.4.3 shows how node A updates its routing table after receiving the partial table from node C.

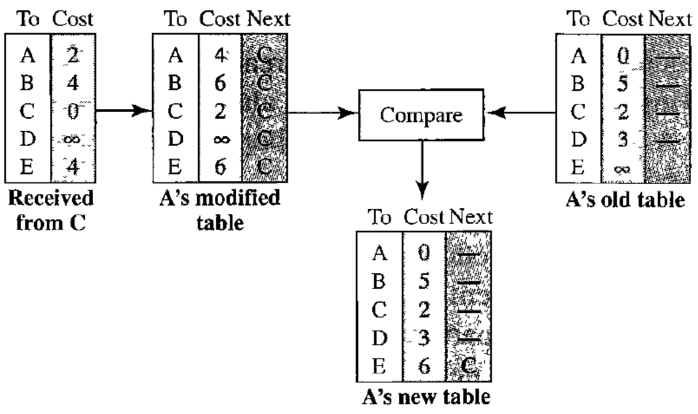


Figure 3.4.3 Updating in distance vector routing

- There are several points we need to emphasize here.
- First, as we know from mathematics, when we add any number to infinity, the result is still infinity.
- Second, the modified table shows how to reach A from A via C. If A needs to reach itself via C, it needs to go to C and come back, a distance of 4.
- Third, the only benefit from this updating of node A is the last entry, how to reach E. Previously, node A did not know how to reach E (distance of infinity); now it knows that the cost is 6 via C.
- Each node can update its table by using the tables received from other nodes. In a short time, if there is no change in the network itself, such as a failure in a link, each node reaches a stable condition in which the contents of its table remains the same.

When to share

- The question now is, When does a node send its partial routing table (only two columns) to all its immediate neighbors? The table is sent both periodically and when there is a change in the table.
- **Periodic Update**

- A node sends Its routing table, normally every 30 s, in a periodic update. The period depends on the protocol that is using distance vector routing.

- **Triggered Update**
  - A node sends its two-column routing table to its neighoors any time there is a change in its routing table. This is called a triggered update.
  - The change can result from the following.
    - A node receives a table from a neighbor, resulting in changes in its own table after updating.
    - A node detects some failure in the neighboring links which results in a distance change to infinity.

Count to Infinity

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

Two-Node loop

- One example of count to infinity is the two-node loop problem. To understand the problem, let us look at the scenario depicted in Figure 3.4.4.

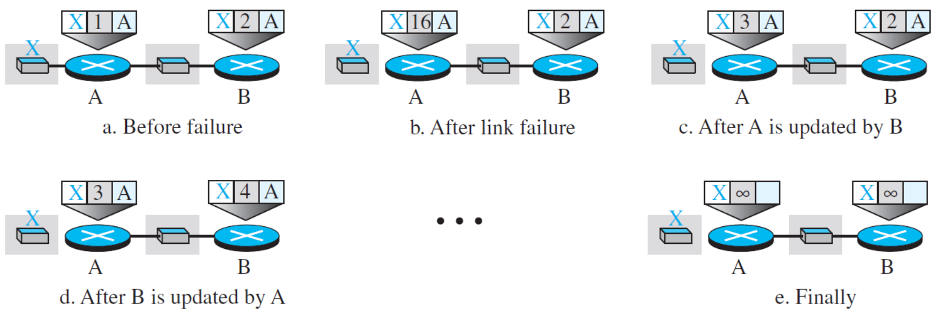


Figure 3.4.4 Two-node instability

- The figure shows a system with three nodes. We have shown only the portions of the forwarding table needed for our discussion.
- At the beginning,
  - both nodes A and B know how to reach node X.
  - But suddenly, the link between A and X fails.
  - Node A changes its table.
  - If A can send its table to B immediately, everything is fine. However, the system becomes unstable if B sends its forwarding table to A before receiving A's forwarding table.
  - Node A receives the update and, assuming that B has found a way to reach X, immediately updates its forwarding table.

  - Now A sends its new update to B. Now B thinks that something has been changed around A and updates its forwarding table.
  - The cost of reaching X increases gradually until it reaches infinity.
  - At this moment, both A and B know that X cannot be reached.
  - However, during this time the system is not stable. Node A thinks that the route to X is via B; node B thinks that the route to X is via A. If A receives a packet destined for X, the packet goes to B and then comes back to A. Similarly, if B receives a packet destined for X, it goes to A and comes back to B. Packets bounce between A and B, creating a two-node loop problem.
- A few solutions have been proposed for instability of this kind.

Split Horizon

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

Split Horizon and Poison Reverse

- Using the split-horizon strategy has one drawback. Normally, the corresponding protocol uses a timer, and if there is no news about a route, the node deletes the route from its table. When node B in the previous scenario eliminates the route to X from its advertisement to A, node A cannot guess whether this is due to the split-horizon strategy (the source of information was A) or because B has not received any news about X recently. In the poison reverse strategy B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: “Do not use this value; what I know about this route comes from you.”

Three Node Instability

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