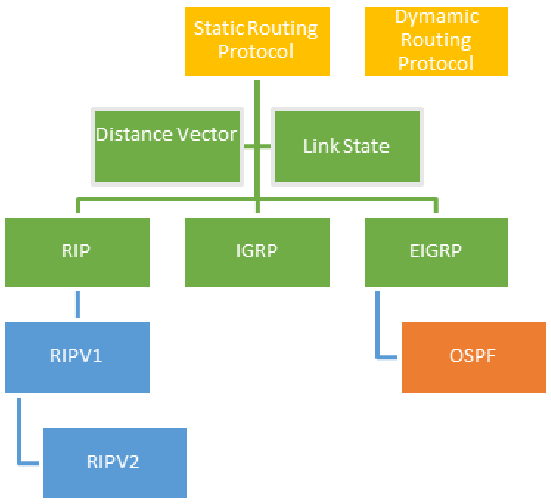
What is Routing Protocols?

Routing Protocols are the set of defined rules used by the routers to communicate between source & destination. Routers perform the traffic directing functions on the Internet; data packets are forwarded through the networks of the internet from router to router until they reach their destination computer.



A routing protocol shares this information first among immediate neighbors, and then throughout the network. This way, routers gain knowledge of the topology of the network. Network Router protocols helps you to specify way routers communicate with each other. It allows the network to select routes between any two nodes on a computer network.

What is Routing Algorithm?

A **Routing Algorithm** is a method for determining the routing of packets in a node. For each node of a network, the algorithm determines a routing table, which in each destination, matches an output line. The algorithm should lead to a consistent routing, that is to say without loop. This means that you should not route a packet a node to another node that could send back the package.

There are three main types of routing algorithms:

• Distance Vector (distance-vector routing);

• To link state (link state routing);

• Path to vector (path-vector routing).

Routing Table

A communication network connects a set of nodes through links so that traffic can move from an originating node to a destination node; for all the traffic to go to its destination, nodes in the network must provide directions so that the traffic goes toward the destination.

To do that, each node in the network maintains a routing table so that user traffic can be forwarded by looking up the routing table to the next hop.

we indicated that nodes need identifiers along with a link identifier so that those identified can be used in the routing table. In an IP network, nodes are routers and links are often identified by interfaces at each end of routers.

However, user traffic originates from host computers and goes to other host computers in the network; that is, the traffic does not originate or terminate at the router level (except routing information traffic between routers).

Thus, we first need to understand what entries are listed in the routing table at an IP router if the traffic eventually terminates at a host.

we need to refer to IP addressing and its relation to routing table entries. A routing table entry at a router can contain information at three levels: addressable networks (or IP prefixes, or network numbers), subnets, or directly at the host level, which is conceptually possible because the IP addressing structure allows all three levels to be specified without any change in the addressing scheme.

These three levels are often referred to using the generic term routes.
Furthermore, this also means that a router maintains entries for IP destinations, not to the router itself.

We will now illustrate the relationship between an IP addressing and routing table through a three-node example figure, we consider routing table entries for addressable networks at Class

C address boundaries, and thus, subnet masking is /24. In Figure 5.1, the IP core network consists of three routers:

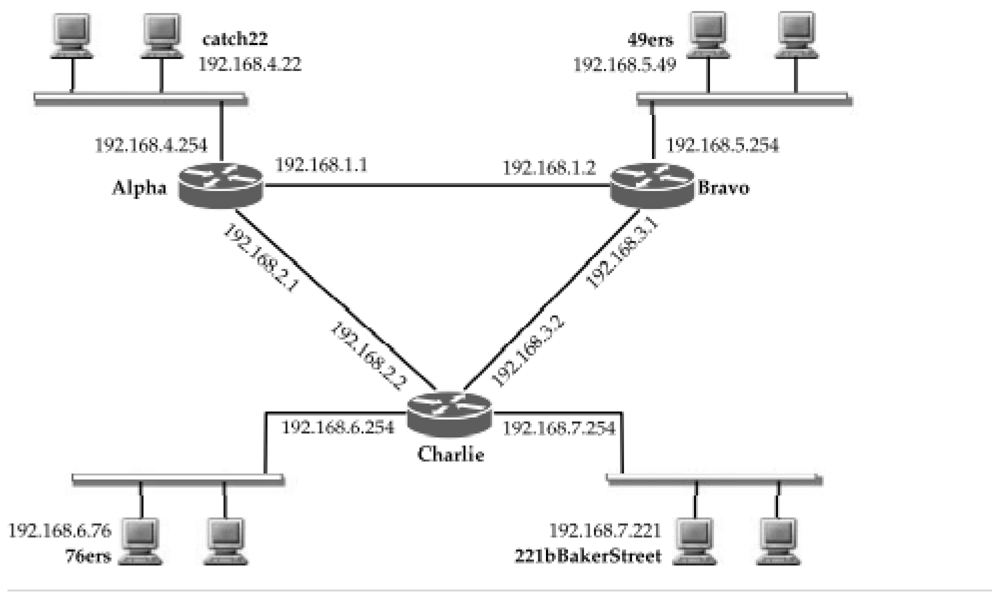
“Alpha,” “Bravo,” and “Charlie;” they help movement of traffic between the following subnets: 192.168.4.0, 192.168.5.0, 192.168.6.0, and 192.168.7.0; as you can see, these are the networks attached to routers Alpha, Bravo, and Charlie, respectively.

You will also notice that we use another set of IP addresses/subnets to provide interfacing between different routers; specifically, address block 192.168.1.0 between routers Alpha and Bravo, 192.168.2.0 between routers Alpha and Charlie, and 192.168.3.0 between routers Bravo and Charlie. Furthermore, each interface that connects to a router has a specific IP address; for example, IP address 192.168.1.2 is on an interface on router Bravo that router Alpha sees while IP address 192.168.3.1 is on another interface that router Charlie sees while 192.168.5.254 is on yet another interface that the addressable network 192.168.5.0 sees.

We have shown the routing table at each router for all different address blocks in Table 5.1. Now consider host “catch22” with IP address 192.168.4.22 in the network 192.168.4.0 that has an IP packet to send to host “49ers” with IP address 192.168.5.49 in network 192.168.5.0.

This packet will arrive at router Alpha on the interface with IP address 192.168.4.254; through routing table lookup, router Alpha realizes that the next hop is 192.168.1.2 for network 192.168.5.0 and will forward the packet to router Bravo.

On receiving this packet, router Bravo realizes that network 192.168.5.0 is directly connected and thus will send it out on interface 192.168.5.254. Now, consider an IP packet going in the reverse direction from 49ers to catch22. The packet will arrive at the interface with IP address 192.168.5.254 at router Bravo. Immediately, router Bravo realizes that for this packet, the next hop is 192.168.1.1 to forward to router Alpha. On receiving this packet, router Alpha will recognize that network 192.168.4.0 is



F I G U R E 5.1 IP network illustration

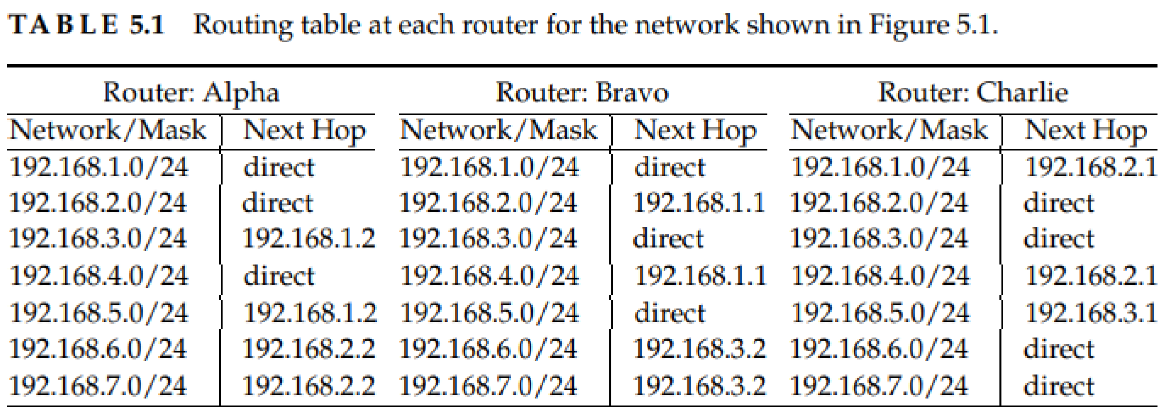


TABLE 5.1 Routing table at each router for the network shown in Figure 5.1.

Router: Alpha		Router: Bravo		Router: Charlie	
Network/Mask	Next Hop	Network/Mask	Next Hop	Network/Mask	Next Hop
192.168.1.0/24	direct	192.168.1.0/24	direct	192.168.1.0/24	192.168.2.1
192.168.2.0/24	direct	192.168.2.0/24	192.168.1.1	192.168.2.0/24	direct
192.168.3.0/24	192.168.1.2	192.168.3.0/24	direct	192.168.3.0/24	direct
192.168.4.0/24	direct	192.168.4.0/24	192.168.1.1	192.168.4.0/24	192.168.2.1
192.168.5.0/24	192.168.1.2	192.168.5.0/24	direct	192.168.5.0/24	192.168.3.1
192.168.6.0/24	192.168.2.2	192.168.6.0/24	192.168.3.2	192.168.6.0/24	direct
192.168.7.0/24	192.168.2.2	192.168.7.0/24	192.168.3.2	192.168.7.0/24	direct

Thus, catch22 sees Alpha as 192.168.4.254, while Bravo sees the same router as 192.168.1.1. From an interface point of view, both are correct. A router has to have at least two interfaces (otherwise, it is not routing/forwarding anything!).

A router is assigned a router ID, which is either one of the interface addresses or a different address altogether. For example, typically the interface address with the highest IP address is assigned as the address of the router. For ease of human tracking, a router with its different interfaces is typically associated with an easy to remember domain name, say Alpha.NetworkRouting.net; then, interface addresses are assigned relevant domain names such as 4net.Alpha.NetworkRouting.net and 1net.Alpha.NetworkRouting.net, so that the subnets can be easily identified and their association with a router is easy to follow.

In the above illustration, we have used a Class C address block for addressable networks. We can easily add a subnet in the routing table that is based on variable-length subnet masking (VLSM) where the subnet mask needs to be explicitly noted due to CIDR.

Further more, a host can have an entry in the routing table as well. Suppose a host with IP address 192.168.8.88 is directly connected to router Charlie through a point-to-point link (not shown in Figure 5.1). If this is so, all routers will have an entry in the routing table for 192.168.8.88 (see Exercise 5.8).

Usually, direct connection of hosts to a router is not advisable since this can lead to significant growth in the routing table, thus impacting packet processing and routing table lookup (see Chapter 15). From the above illustration, you may also notice that the term network is used in multiple ways. Consider the following statement: user traffic moves from a network to another network that is routed through one or more routers in the IP network.

Certainly, this is a confusing statement. To translate this, the first two uses of network refer to a network identified through an IP prefix where traffic originates or terminates at hosts, while the third use of network refers to a network in the topological sense where routers are nodes connected by li The first two uses of network are also referred to as route. Since a routing table can have an entry directly for a specific host (at least in theory), the term route is a good term without being explicit as to whether it is a network number or a host.
A network identified using an IP prefix will be referred to as network number or addressable network, or simply as IP prefix; we will also use the term route interchangeably. This then avoids any confusion with the generic term network used throughout the book.

Communication of Routing Information An important aspect of the TCP/IP protocol stack is that all types of communications must take place within the same TCP/IP stack framework; that is, there are no separate networks or channels or dedicated circuits for communicating control or routing messages separately from user traffic.

To accommodate **different types of messages or information**, the TCP/IP stack provides functionalities at both the IP layer and the transport layer; this is done differently for different routing protocols.

For example, in the case of the RIP protocol, messages are communicated above the transport layer using a UDP-based port number; specifically, port number 520 is used with UDP as the transport protocol. How about other routing protocols? BGP is assigned port number 179 along with TCP as the transport protocol. However, for several routing protocols, identification is done directly above the IP layer using protocol number field; for example, protocol number 9 for IDRP protocol, 88 for EIGRP, and 89 for OSPF protocol. It may be noted that reliability of message transfer in BGP is inherently addressed since TCP is used; however, for OSPF and EIGRP, which require reliable delivery of routing information, reliability cannot be inherently guaranteed since they are directly above the IP layer; thus, for communication of routing information in OSPF, for example, through flooding, it is required that the protocol implementation ensures that communication is reliable by using acknowledgment and retransmission (if needed). In any case, while it may sound strange, all routing protocols act as applications in the TCP/IP framework where RIP and BGP are application layer protocols while OSPF and IS-IS are protocols that sit just above the IP layer. In other words, to get the routing information out for the IP layer to establish routing/forwarding of user traffic, the network relies on a higher layer protocol.

Static Routes While routing protocols are useful to determine routes dynamically, it is sometimes desirable in an operational environment to indicate routes that remain static. These routes, referred to as static routes, are required to be configured manually at routers. For example, sometimes a network identified by an IP prefix is connected to only one router in another network; this happens to be the only route out to the Internet. Such a network is called a stub network; in such a case, a static route can be manually configured to a stub network. Static routes can also be defined when two autonomous systems must exchange routing information. It is important to set up any static routes carefully. For example, if not careful, it is possible to get into open jaw routes. This terms means that there is a path defined from an origin to a destination; however, the destination takes a different path in return that does not make it back to the origin.

Routing Information Protocol, Version 1 (RIPv1) RIP is the first routing protocol used in the TCP/IP-based network in an intradomain environment.

While the RIP specification was first described in RFC 1058 in 1988 [290], it was available when RIP was packaged with 4.3 Berkeley Software Distribution (BSD) as part of “routed” daemon software in the early 1980s.

The following passage from RFC 1058 is interesting to note: “The Routing Information Protocol (RIP) described here is loosely based on the program routed, distributed with the 4.3 Berkeley Software Distribution. However, there are several other implementations of what is

supposed to be the same protocol. Unfortunately, these various implementations disagree in various details.
The specifications here represent a combination of features taken from various implementations.” The name RIP can be deceiving since all routing protocols need to exchange “routing information.” RIP should be understood as an instance of a distance vector protocol family, regardless of its name. It was one of the few protocols for which an implementation was available before a specification was officially complete.

The original RIP is now referred to as RIP version 1, or RIPv1 in short. It has since evolved to RIPv2, which is standardized in RFC 2453 [442]. RIP remains one of the popular routing protocols for a small network environment.

In fact, most DSL/cable modem routers such as the ones from Linksys come bundled with RIP. Thus, if you want to set up a private layer-3 IP network in your home or small office, you can do so by using multiple routers where you can invoke RIP. 5.3.1 Communication and Message

Format Since distance vector information is obtained from a neighboring router, the communication of routing information is always between two neighboring routers in the case of RIP. Furthermore, since RIP is UDP based, there is no guarantee that a routing information message is received by a receiving router.
Also, no session is set up; a routing packet is just encapsulated and sent to the neighbor, normally through broadcast. Thus, we can envision a routing packet in the TCP/IP stack as shown in Figure 5.2. Next we consider the format of a RIPv1 message; this is shown in Figure 5.3. As a commonly accepted convention in IP, the packet format for RIPv1 is shown in 32-bit (4-byte) boundaries. A RIPv1 message has a common header of 4 bytes, followed by a 20- byte message for each route for which the message is communicating, up to a maximum of 25 routes/addresses. Thus, the maximum size of a RIP message (including IP/UDP headers)

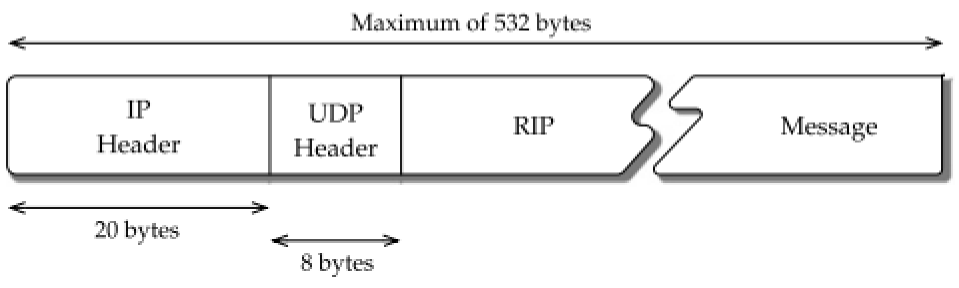


FIGURE 5.2 RIP message structure, with IP and UDP header.

1. Distance Vector Routing Protocol – These protocols selects best path in the basis of hop counts to reach a destination network in the particular direction. Dynamic protocol like RIP is an example of distance vector routing protocol. Hop count is each router which occurs in between the source and the destination network. The path with the least hop count will be chosen as the best path.

Features –

- Updates of network are exchanged periodically.
- Updates (routing information) is always broadcast.
- Full routing tables are sent in updates.
- Routers always trust on routing information received from neighbor routers. This is also known as routing on rumors.

Disadvantages –

- As the routing information are exchanged periodically, unnecessary traffic is generated which consumes available bandwidth.
- As full routing tables are exchanged, therefore it has security issues. If an authorized person enters the network, then the whole topology will be very easy to understand.
- Also broadcasting of network periodically creates unnecessary traffic.

2. Link State Routing Protocol – These protocols know more about the Internetwork than any other distance vector routing protocol. These re also known as SPF (Shortest Path First) protocol. OSPF is an example of link state routing protocol.

Features –

- Hello messages, also known as keep-alive messages are used for neighbor discovery and recovery.
- Concept of triggered updates are used i.e updates are triggered only when there is a topology change .
- Only that much updates are exchanged which is requested by the neighbor router.

Link state routing protocol maintains three tables namely:

1. **Neighbor table-** the table which contains information about the neighbors of the router only, i.e, to which adjacency has been formed.
2. **Topology table-** This table contains information about the whole topology i.e contains both best and backup routes to particular advertised network.
3. **Routing table-** This table contains all the best routes to the advertised network.

Advantages –

- As it maintains separate tables for both best route and the backup routes (whole topology) therefore it has more knowledge of the inter network than any other distance vector routing protocol.
- Concept of triggered updates are used therefore no more unnecessary bandwidth consumption is seen like in distance vector routing protocol.
- Partial updates are triggered when there is a topology change, not a full update like distance vector routing protocol where whole routing table is exchanged.

3. Advanced Distance vector routing protocol – It is also known as hybrid routing protocol which uses the concept of both distance vector and link state routing protocol. EIGRP acts as a link state routing protocol as it uses the concept of Hello protocol for neighbor discovery and forming adjacency. Also, partial updates are triggered when a change occurs. EIGRP acts as distance vector routing protocol as it learned routes from directly connected neighbours.

The **distance-vector routing Protocol** is a type of algorithm used by routing protocols to discover routes on an interconnected network. The primary distance-vector routing protocol algorithm is the Bellman-Ford algorithm. Another type of routing protocol algorithm is the *link-state* approach.

Routing protocols that use distance-vector routing protocols include RIP (**Routing Information Protocol**), Cisco's IGRP (**Internet Gateway Routing Protocol**), and Apple's RTMP (**Routing Table Maintenance Protocol**). The most common link-state routing protocol is OSPF (**Open Shortest Path First**). ***Dynamic routing***, as opposed to static (manually entered) routing, requires routing protocol algorithms

Distance Vector Routing Algorithm

- **The Distance vector algorithm is iterative, asynchronous and distributed.**
  - **Distributed:** It is distributed in that each node receives information from one or more of its directly attached neighbors, performs calculation and then distributes the result back to its neighbors.
  - **Iterative:** It is iterative in that its process continues until no more information is available to be exchanged between neighbors.
  - **Asynchronous:** It does not require that all of its nodes operate in the lock step with each other.
- The Distance vector algorithm is a dynamic algorithm.
- It is mainly used in ARPANET, and RIP.

- Each router maintains a distance table known as **Vector**.

Three Keys to understand the working of Distance Vector Routing Algorithm:

- **Knowledge about the whole network:** Each router shares its knowledge through the entire network. The Router sends its collected knowledge about the network to its neighbors.
- **Routing only to neighbors:** The router sends its knowledge about the network to only those routers which have direct links. The router sends whatever it has about the network through the ports. The information is received by the router and uses the information to update its own routing table.
- **Information sharing at regular intervals:** Within 30 seconds, the router sends the information to the neighboring routers.

Distance Vector Routing Algorithm

Let $d_x(y)$ be the cost of the least-cost path from node x to node y. The least costs are related by Bellman-Ford equation,

$$d_x(y) = \min_v\{c(x,v) + d_v(y)\}$$

Where the minv is the equation taken for all x neighbors. After traveling from x to v, if we consider the least-cost path from v to y, the path cost will be $c(x,v)+d_v(y)$. The least cost from x to y is the minimum of $c(x,v)+d_v(y)$ taken over all neighbors.

With the Distance Vector Routing algorithm, the node x contains the following routing information:

- For each neighbor v, the cost c(x,v) is the path cost from x to directly attached neighbor, v.
- The distance vector x, i.e., $D_x = [\ D_x(y) : y \text{ in } N\ ]$, containing its cost to all destinations, y, in N.
- The distance vector of each of its neighbors, i.e., $D_v = [\ D_v(y) : y \text{ in } N\ ]$ for each neighbor v of x.

Distance vector routing is an asynchronous algorithm in which node x sends the copy of its distance vector to all its neighbors.

When node x receives the new distance vector from one of its neighboring vector, v, it saves the distance vector of v and uses the Bellman-Ford equation to update its own distance vector.

The equation is given below:

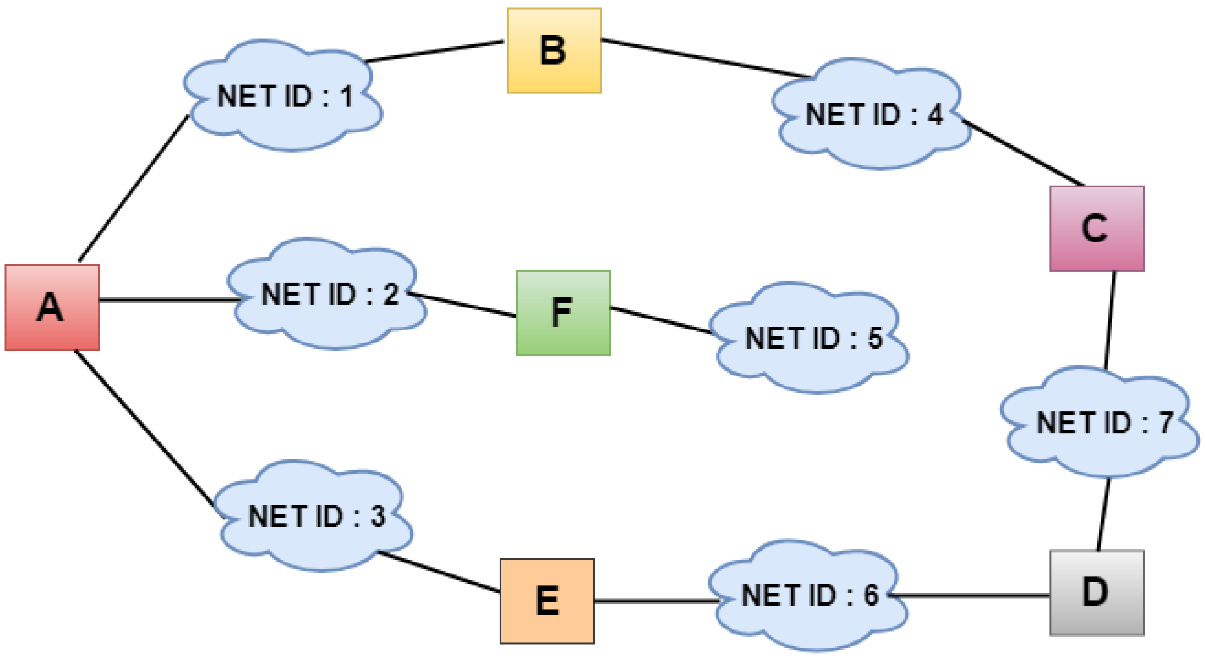
$$d_x(y) = \min_v\{\ c(x,v) + d_v(y)\} \quad \text{for each node y in N}$$

The node x has updated its own distance vector table by using the above equation and sends its updated table to all its neighbors so that they can update their own distance vectors.

Algorithm

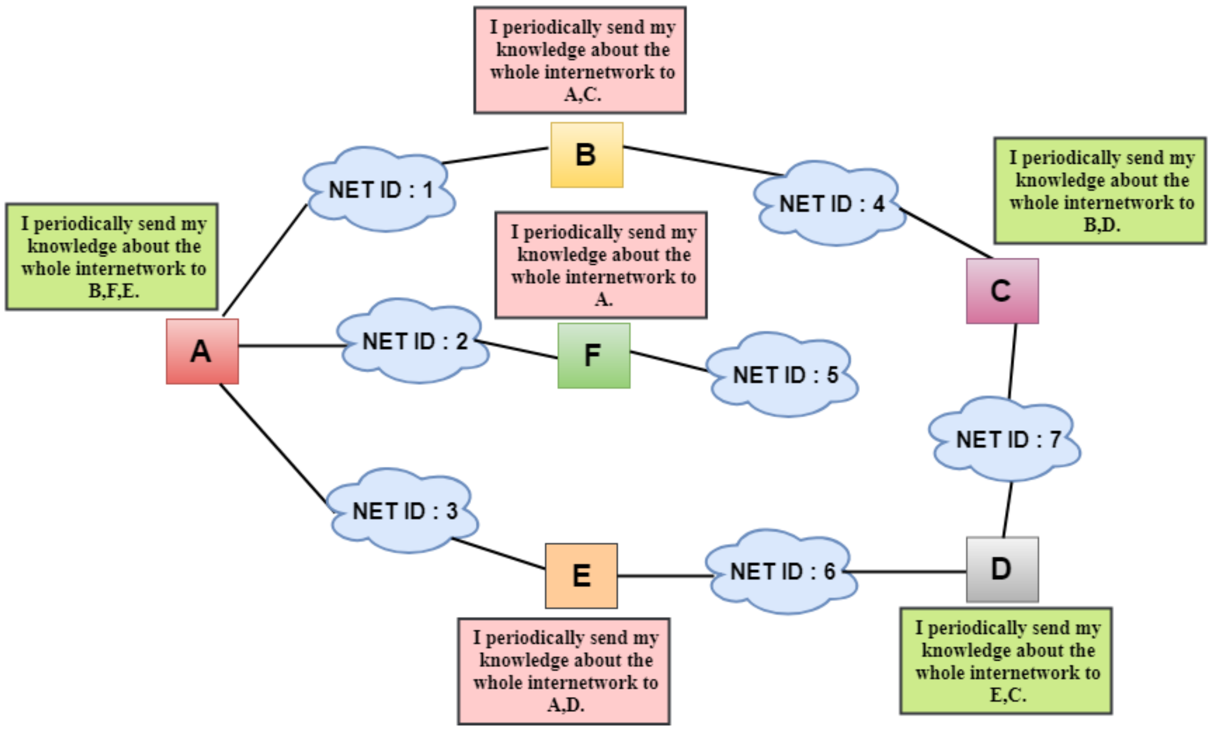
```
At each node x,

Initialization


for all destinations y in N:
$D_x(y) = c(x,y)$     // If y is not a neighbor then $c(x,y) = \infty$
for each neighbor w
$D_w(y) = ?$     for all destination y in N.
for each neighbor w
send distance vector $D_x = [ D_x(y) : y \text{ in } N ]$ to w
loop
  wait(until I receive any distance vector from some neighbor w)
  for each y in N:
  $D_x(y) = \text{minv}\{c(x,v)+D_v(y)\}$
If $D_x(y)$ is changed for any destination y
Send distance vector $D_x = [ D_x(y) : y \text{ in } N ]$ to all neighbors
forever
```

Sharing Information



- In the above figure, each cloud represents the network, and the number inside the cloud represents the network ID.
- All the LANs are connected by routers, and they are represented in boxes labeled as A, B, C, D, E, F.

- Distance vector routing algorithm simplifies the routing process by assuming the cost of every link is one unit. Therefore, the efficiency of transmission can be measured by the number of links to reach the destination.
- In Distance vector routing, the cost is based on hop count.


I periodically send my knowledge about the whole internetwork to A,C.
B
I periodically send my knowledge about the whole internetwork to B,F,E.
NET ID : 1
NET ID : 4
I periodically send my knowledge about the whole internetwork to B,D.
I periodically send my knowledge about the whole internetwork to A.
C
A
NET ID : 2
F
NET ID : 5
NET ID : 7
NET ID : 3
E
NET ID : 6
D
I periodically send my knowledge about the whole internetwork to A,D.
I periodically send my knowledge about the whole internetwork to E,C.

In the above figure, we observe that the router sends the knowledge to the immediate neighbors. The neighbors add this knowledge to their own knowledge and sends the updated table to their own neighbors. In this way, routers get its own information plus the new information about the neighbors.

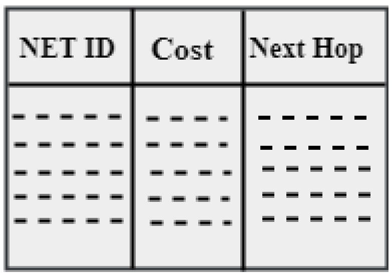
Routing Table

Two process occurs:

- Creating the Table
- Updating the Table

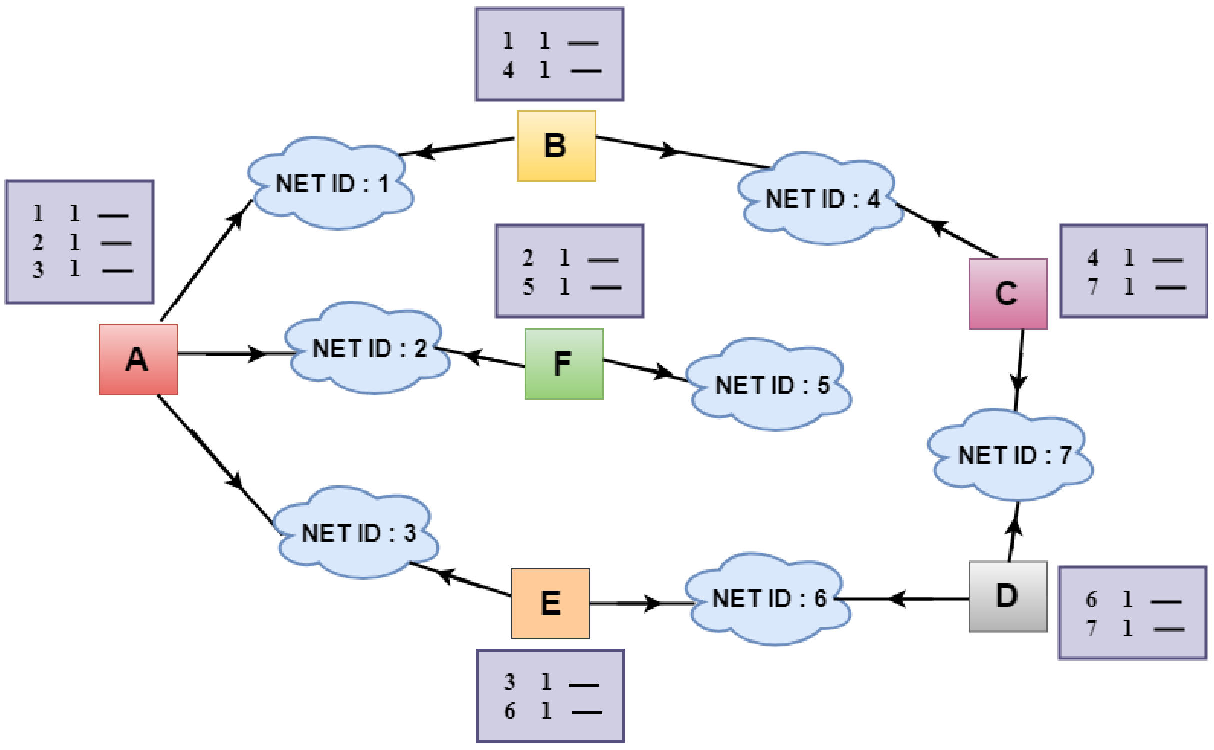
Creating the Table

Initially, the routing table is created for each router that contains atleast three types of information such as Network ID, the cost and the next hop.



NET ID	Cost	Next Hop
- - - - -	- - - -	- - - - -
- - - - -	- - - -	- - - - -
- - - - -	- - - -	- - - - -
- - - - -	- - - -	- - - - -
- - - - -	- - - -	- - - - -

- **NET ID:** The Network ID defines the final destination of the packet.
- **Cost:** The cost is the number of hops that packet must take to get there.
- **Next hop:** It is the router to which the packet must be delivered.



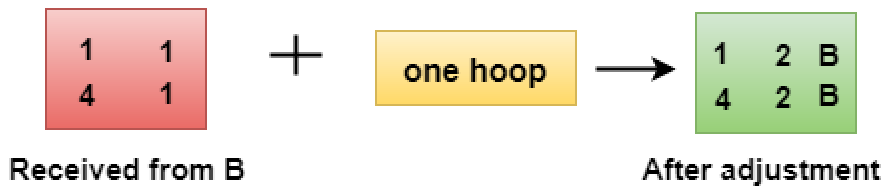
- In the above figure, the original routing tables are shown of all the routers. In a routing table, the first column represents the network ID, the second column represents the cost of the link, and the third column is empty.
- These routing tables are sent to all the neighbors.

For Example:

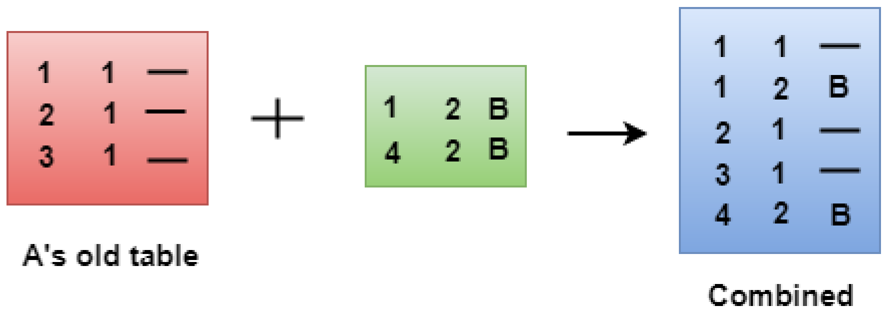
1. A sends its routing table to B, F & E.
2. B sends its routing table to A & C.
3. C sends its routing table to B & D.
4. D sends its routing table to E & C.
5. E sends its routing table to A & D.
6. F sends its routing table to A.

Updating the Table

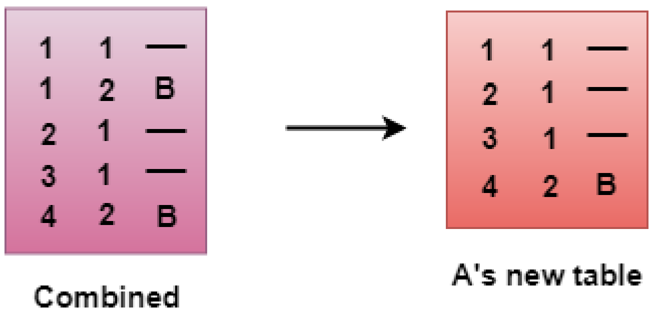
- When A receives a routing table from B, then it uses its information to update the table.
- The routing table of B shows how the packets can move to the networks 1 and 4.
- The B is a neighbor to the A router, the packets from A to B can reach in one hop. So, 1 is added to all the costs given in the B's table and the sum will be the cost to reach a particular network.



- After adjustment, A then combines this table with its own table to create a combined table.

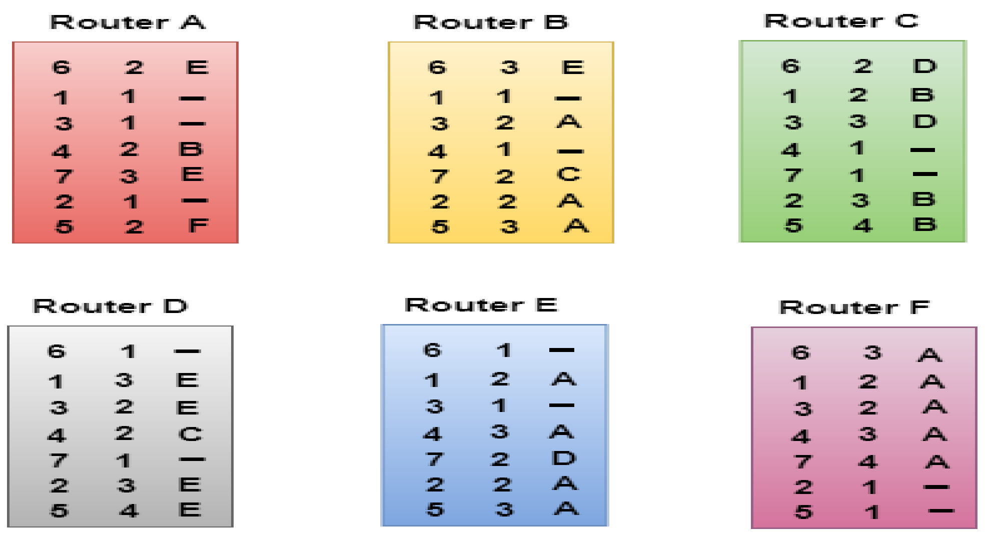


- The combined table may contain some duplicate data. In the above figure, the combined table of router A contains the duplicate data, so it keeps only those data which has the lowest cost. For example, A can send the data to network 1 in two ways. The first, which uses no next router, so it costs one hop. The second requires two hops (A to B, then B to Network 1). The first option has the lowest cost, therefore it is kept and the second one is dropped.



- The process of creating the routing table continues for all routers. Every router receives the information from the neighbors, and update the routing table.

Final routing tables of all the routers are given below:



Router A

6	2	E
1	1	—
3	1	—
4	2	B
7	3	E
2	1	—
5	2	F

Router B

6	3	E
1	1	—
3	2	A
4	1	—
7	2	C
2	2	A
5	3	A

Router C

6	2	D
1	2	B
3	3	D
4	1	—
7	1	—
2	3	B
5	4	B

Router D

6	1	—
1	3	E
3	2	E
4	2	C
7	1	—
2	3	E
5	4	E

Router E

6	1	—
1	2	A
3	1	—
4	3	A
7	2	D
2	2	A
5	3	A

Router F

6	3	A
1	2	A
3	2	A
4	3	A
7	4	A
2	1	—
5	1	—

Link State Routing

Link state routing is a technique in which each router shares the knowledge of its neighborhood with every other router in the internetwork.

The three keys to understand the Link State Routing algorithm:

- **Knowledge about the neighborhood:** Instead of sending its routing table, a router sends the information about its neighborhood only. A router broadcast its identities and cost of the directly attached links to other routers.
- **Flooding:** Each router sends the information to every other router on the internetwork except its neighbors. This process is known as Flooding. Every router that receives the packet sends the copies to all its neighbors. Finally, each and every router receives a copy of the same information.
- **Information sharing:** A router sends the information to every other router only when the change occurs in the information.

Link State Routing has two phases:

Reliable Flooding

- **Initial state:** Each node knows the cost of its neighbors.
- **Final state:** Each node knows the entire graph.

Route Calculation

Each node uses Dijkstra's algorithm on the graph to calculate the optimal routes to all nodes.

- The Link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.
- The Dijkstra's algorithm is an iterative, and it has the property that after k^{th} iteration of the algorithm, the least cost paths are well known for k destination nodes.

Let's describe some notations:

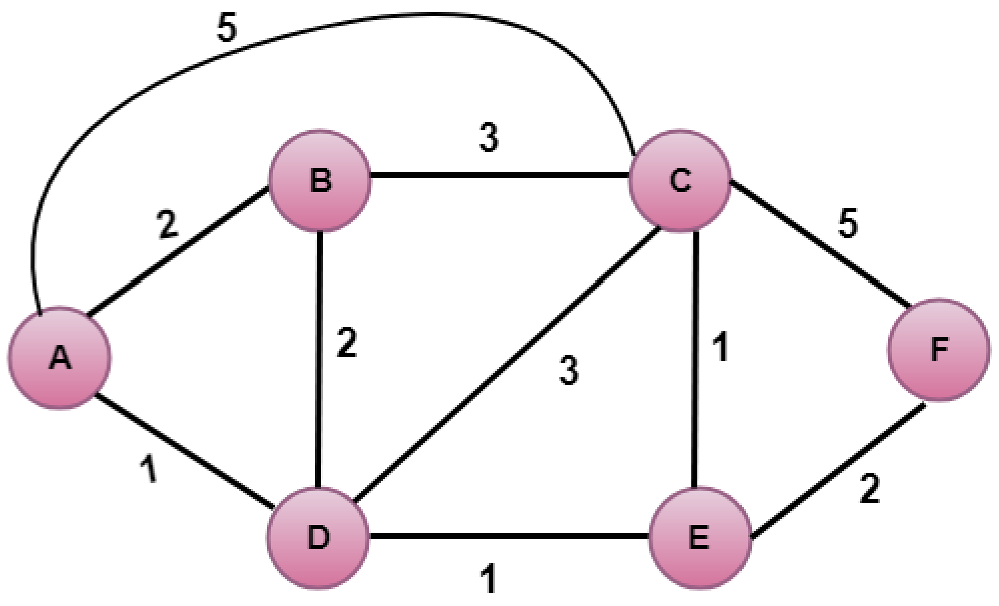
- **c(i , j):** Link cost from node i to node j. If i and j nodes are not directly linked, then c(i , j) = ∞.
- **D(v):** It defines the cost of the path from source code to destination v that has the least cost currently.
- **P(v):** It defines the previous node (neighbor of v) along with current least cost path from source to v.
- **N:** It is the total number of nodes available in the network.

Algorithm

```
Initialization
N = {A}     // A is a root node.
for all nodes v
if v adjacent to A
then D(v) = c(A,v)
else D(v) = infinity
loop
find w not in N such that D(w) is a minimum.
Add w to N
Update D(v) for all v adjacent to w and not in N:
D(v) = min(D(v) , D(w) + c(w,v))
Until all nodes in N
```

In the above algorithm, an initialization step is followed by the loop. The number of times the loop is executed is equal to the total number of nodes available in the network.

Let's understand through an example:



In the above figure, source vertex is A.

Step 1:

The first step is an initialization step. The currently known least cost path from A to its directly attached neighbors, B, C, D are 2,5,1 respectively. The cost from A to B is set to 2, from A to D is set to 1 and from A to C is set to 5. The cost from A to E and F are set to infinity as they are not directly linked to A.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(
1	A	2,A	5,A	1,A	

Step 2:

In the above table, we observe that vertex D contains the least cost path in step 1. Therefore, it is added in N. Now, we need to determine a least-cost path through D vertex.

a) Calculating shortest path from A to B

1. v = B, w = D
2. D(B) = min(D(B) , D(D) + c(D,B))
3. = min(2, 1+2)>
4. = min(2, 3)
5. The minimum value is 2. Therefore, the currently shortest path from A to B is 2.

b) Calculating shortest path from A to C

1. v = C, w = D
2. D(B) = min(D(C) , D(D) + c(D,C))

3. = min(5, 1+3)
4. = min(5, 4)
5. The minimum value is 4. Therefore, the currently shortest path from A to C is 4.</p>

c) Calculating shortest path from A to E

1. v = E, w = D
2. D(B) = min(D(E) , D(D) + c(D,E))
3. = min(∞, 1+1)
4. = min(∞, 2)
5. The minimum value is 2. Therefore, the currently shortest path from A to E is 2.

Note: The vertex D has no direct link to vertex E. Therefore, the value of D(F) is infinity.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(
1	A	2,A	5,A	1,A	
2	AD	2,A	4,D		

Step 3:

In the above table, we observe that both E and B have the least cost path in step 2. Let's consider the E vertex. Now, we determine the least cost path of remaining vertices through E.

a) Calculating the shortest path from A to B.

1. v = B, w = E
2. D(B) = min(D(B) , D(E) + c(E,B))
3. = min(2 , 2+ ∞)
4. = min(2, ∞)
5. The minimum value is 2. Therefore, the currently shortest path from A to B is 2.

b) Calculating the shortest path from A to C.

1. v = C, w = E
2. D(B) = min(D(C) , D(E) + c(E,C))
3. = min(4 , 2+1)
4. = min(4,3)
5. The minimum value is 3. Therefore, the currently shortest path from A to C is 3.

c) Calculating the shortest path from A to F.

1. v = F, w = E
2. D(B) = min(D(F) , D(E) + c(E,F))

3. = min(∞ , 2+2)
4. = min(∞ ,4)
5. The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D
1	A	2,A	5,A	1,A	
2	AD	2,A	4,D		
3	ADE	2,A	3,E		

Step 4:

In the above table, we observe that B vertex has the least cost path in step 3. Therefore, it is added in N. Now, we determine the least cost path of remaining vertices through B.

a) Calculating the shortest path from A to C.

1. v = C, w = B
2. D(B) = min(D(C) , D(B) + c(B,C))
3. = min(3 , 2+3)
4. = min(3,5)
5. The minimum value is 3. Therefore, the currently shortest path from A to C is 3.

b) Calculating the shortest path from A to F.

1. v = F, w = B
2. D(B) = min(D(F) , D(B) + c(B,F))
3. = min(4, ∞)
4. = min(4, ∞)
5. The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	I
1	A	2,A	5,A	1,A	
2	AD	2,A	4,D		
3	ADE	2,A	3,E		

4	ADEB		3,E		

Step 5:

In the above table, we observe that C vertex has the least cost path in step 4. Therefore, it is added in N. Now, we determine the least cost path of remaining vertices through C.

a) Calculating the shortest path from A to F.

1. v = F, w = C
2. D(B) = min(D(F) , D(C) + c(C,F))
3. = min(4, 3+5)
4. = min(4,8)
5. The minimum value is 4. Therefore, the currently shortest path from A to F is 4.

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞
3	ADE	2,A	3,E			4,E
4	ADEB		3,E			4,E
5	ADEBC					4,E

Final table:

Step	N	D(B),P(B)	D(C),P(C)	D(D),P(D)	D(E),P(E)	D(F),P(F)
1	A	2,A	5,A	1,A	∞	∞
2	AD	2,A	4,D		2,D	∞

3	ADE	2,A	3,E			4,E
4	ADEB		3,E			4,E
5	ADEBC					4,E
6	ADEBCF					

Disadvantage:

Heavy traffic is created in Line state routing due to Flooding. Flooding can cause an infinite looping, this problem can be solved by using Time-to-leave field.

RIP Protocol

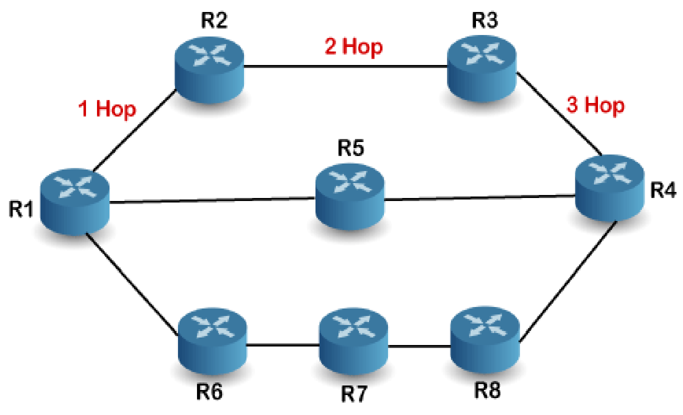
RIP stands for Routing Information Protocol. RIP is an intra-domain routing protocol used within an autonomous system. Here, intra-domain means routing the packets in a defined domain, for example, web browsing within an institutional area. To understand the RIP protocol, our main focus is to know the structure of the packet, how many fields it contains, and how these fields determine the routing table.

Before understanding the structure of the packet, we first look at the following points:

- RIP is based on the distance vector-based strategy, so we consider the entire structure as a graph where nodes are the routers, and the links are the networks.
- In a routing table, the first column is the destination, or we can say that it is a network address.
- The cost metric is the number of hops to reach the destination. The number of hops available in a network would be the cost. The hop count is the number of networks required to reach the destination.
- In RIP, infinity is defined as 16, which means that the RIP is useful for smaller networks or small autonomous systems. The maximum number of hops that RIP can contain is 15 hops, i.e., it should not have more than 15 hops as 16 is infinity.
- The next column contains the address of the router to which the packet is to be sent to reach the destination.

How is hop count determined?

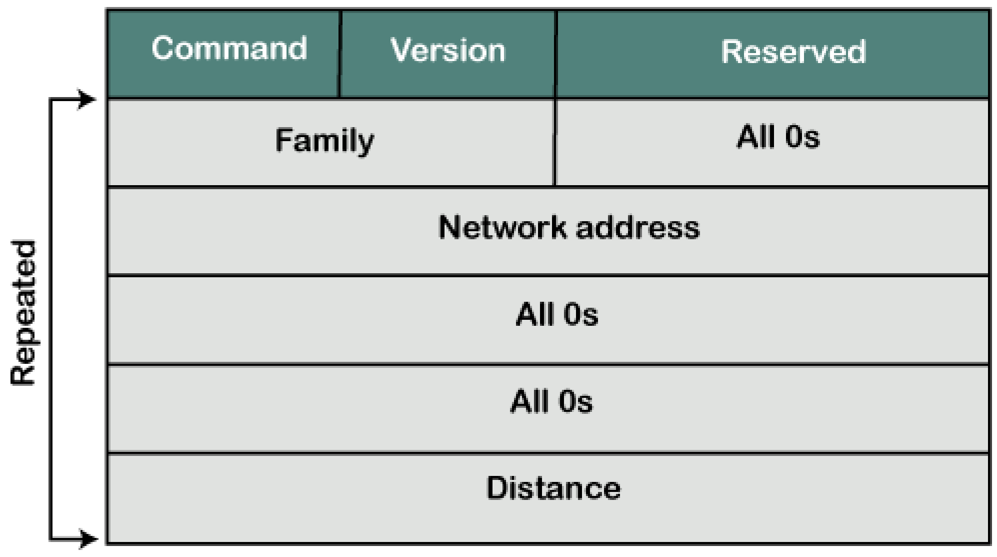
When the router sends the packet to the network segment, then it is counted as a single hop.



In the above figure, when the router 1 forwards the packet to the router 2 then it will count as 1 hop count. Similarly, when the router 2 forwards the packet to the router 3 then it will count as 2 hop count, and when the router 3 forwards the packet to router 4, it will count as 3 hop count. In the same way, RIP can support maximum upto 15 hops, which means that the 16 routers can be configured in a RIP.

RIP Message Format

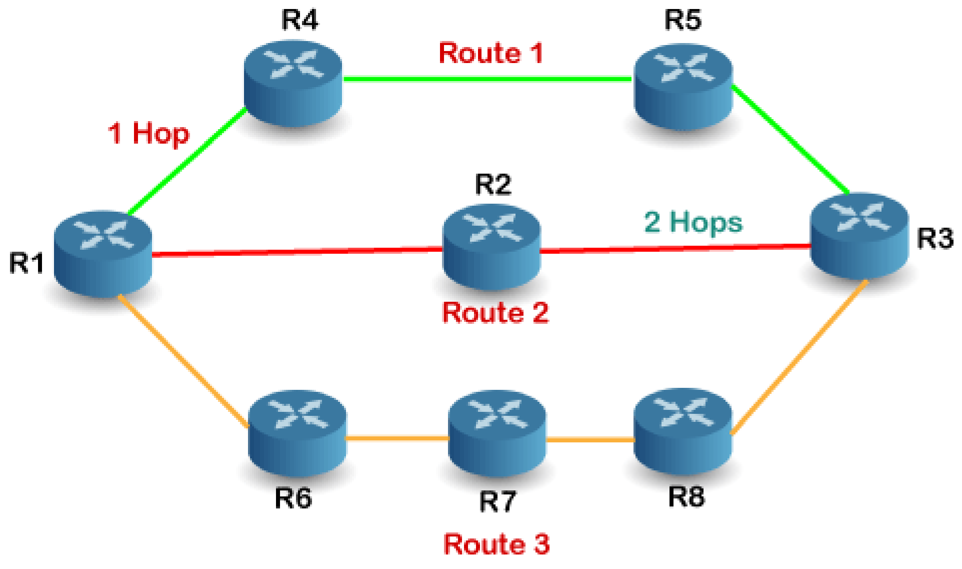
Now, we look at the structure of the RIP message format. The message format is used to share information among different routers. The RIP contains the following fields in a message:



- Command: It is an 8-bit field that is used for request or reply. The value of the request is 1, and the value of the reply is 2.
- Version: Here, version means that which version of the protocol we are using. Suppose we are using the protocol of version1, then we put the 1 in this field.

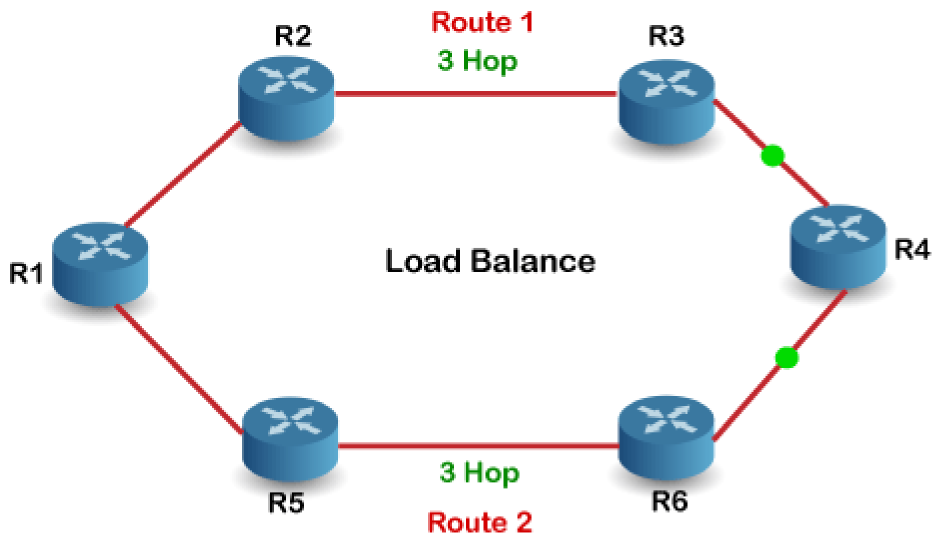
- Reserved: This is a reserved field, so it is filled with zeroes.
- Family: It is a 16-bit field. As we are using the TCP/IP family, so we put 2 value in this field.
- Network Address: It is defined as 14 bytes field. If we use the IPv4 version, then we use 4 bytes, and the other 10 bytes are all zeroes.
- Distance: The distance field specifies the hop count, i.e., the number of hops used to reach the destination.

How does the RIP work?



If there are 8 routers in a network where Router 1 wants to send the data to Router 3. If the network is configured with RIP, it will choose the route which has the least number of hops. There are three routes in the above network, i.e., Route 1, Route 2, and Route 3. The Route 2 contains the least number of hops, i.e., 2 where Route 1 contains 3 hops, and Route 3 contains 4 hops, so RIP will choose Route 2.

Let's look at another example.

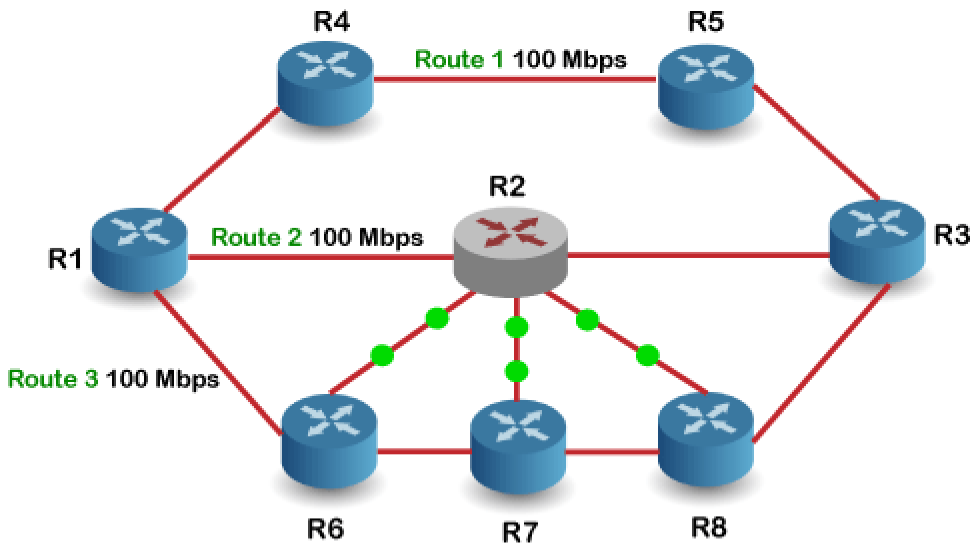


Suppose R1 wants to send the data to R4. There are two possible routes to send data from r1 to r2. As both the routes contain the same number of hops, i.e., 3, so RIP will send the data to both the routes simultaneously. This way, it manages the load balancing, and data reach the destination a bit faster.

Disadvantages of RIP

The following are the disadvantages of RIP:

- In RIP, the route is chosen based on the hop count metric. If another route of better bandwidth is available, then that route would not be chosen. Let's understand this scenario through an example.



We can observe that Route 2 is chosen in the above figure as it has the least hop count. The Route 1 is free and data can be reached more faster; instead of this, data is sent to the Route 2 that makes the Route 2 slower due to the heavy traffic. This is one of the biggest disadvantages of RIP.

- The RIP is a classful routing protocol, so it does not support the VLSM (Variable Length Subnet Mask). The classful routing protocol is a protocol that does not include the subnet mask information in the routing updates.
- It broadcasts the routing updates to the entire network that creates a lot of traffic. In RIP, the routing table updates every 30 seconds. Whenever the updates occur, it sends the copy of the update to all the neighbors except the one that has caused the update. The sending of updates to all the neighbors creates a lot of traffic. This rule is known as a split-horizon rule.
- It faces a problem of Slow convergence. Whenever the router or link fails, then it often takes minutes to stabilize or take an alternative route; This problem is known as Slow convergence.

- RIP supports maximum 15 hops which means that the maximum 16 hops can be configured in a RIP
- The Administrative distance value is 120 (Ad value). If the Ad value is less, then the protocol is more reliable than the protocol with more Ad value.
- The RIP protocol has the highest Ad value, so it is not as reliable as the other routing protocols.

How RIP updates its Routing table

The following timers are used to update the routing table:

- **RIP update timer : 30 sec**

The routers configured with RIP send their updates to all the neighboring routers every 30 seconds.

- **RIP Invalid timer : 180 sec**

The RIP invalid timer is 180 seconds, which means that if the router is disconnected from the network or some link goes down, then the neighbor router will wait for 180 seconds to take the update. If it does not receive the update within 180 seconds, then it will mark the particular route as not reachable.

- **RIP Flush timer : 240 sec**

The RIP flush timer is 240 second which is almost equal to 4 min means that if the router does not receive the update within 240 seconds then the neighbor route will remove that particular route from the routing table which is a very slow process as 4 minutes is a long time to wait.

Advantages of RIP

The following are the advantages of a RIP protocol:

- It is easy to configure
- It has less complexity
- The CPU utilization is less.

Single-Commodity Network Flow

Single-Commodity Network Flow We start with the single-commodity network flow problem. This means that only a node pair has positive demand volume, thus, the name single-commodity where the term commodity refers to a demand.

For illustration of network flow models, we will use a three-node network in this section. 4.2.1 A Three-Node Illustration Consider a three-node network where 5 units of demand volume need to be carried between node 1 and node 2 (see Figure 4.1);

we assume that the demand volume is a deterministic number. We are given that all links in the network are bidirectional and have a capacity of 10 units each. It is easy to see that the direct link 1-2 can easily accommodate the 5 units of demand volume since there the direct link can handle up to 10 units of capacity; this remains the case as long as the demand volume between node 1 and node 2 is 10 units or less. As soon as the demand volume becomes more than 10 units, it is clear that the direct link path cannot

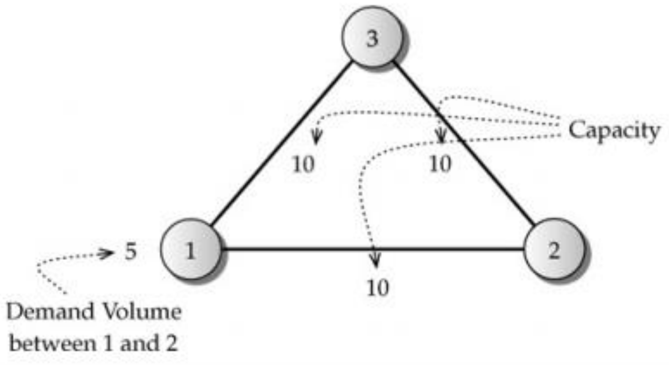


FIGURE 4.1 Three-node network with single demand between node 1 and node 2.

carry all of the demand volume between node 1 and node 2. In other words, any demand in excess of 10 units would need to be carried on the second path 1-3-2 .

This simple illustration illustrates that not all demand volume can always be carried on a single path or the shortest, hop-based path; the capacity limit on a link along a path matters. In addition, we have made an implicit assumption up to this point that the direct link path 1-2 is less costly per unit of demand flow than the two-link alternate path 1-3-2.

However, in many networks, this may not always be true. If we instead suppose that the per-unit cost of the two-link path 1-3-2 is 1 while the per-unit cost on the direct link 1-2 is 2, then it would be more natural or optimal to route demand volume first on the alternate path 1-3-2 for up to the first 10 units of demand volume, and then route any demand volume above the first 10 units on the direct link path 1-2.

The above illustration helps us to see that the actual routing decision should depend on the goal of routing, irrespective of the hop count.

This means that we need a generic way to represent the problem so that various situations can be addressed in a capacitated network in order to find the best solution. 4.2.2 Formal Description and Minimum Cost Routing Objective We are now ready to present the above discussion in a formal manner using unknowns or variables.

We assume here that the capacity of each link is the same and is given by c. Let the demand volume for node pair 1:2 be denoted by h. For example, in the above illustration capacity c was set to 10.

Since the demand volume for the node pair 1:2 can possibly be divided between the direct link path 1-2 and the two-link path 1-3-2 , we can use two unknowns or variables to represent this aspect. Let x12 be the amount of the total demand volume h to be routed on direct link path 1-2 , and let x132 be any amount of the demand volume to be routed on the alternate path 1-3-2 (see Figure 4.2). Note the use of subscripts so that it is easy to track a route with flows. Since the total demand volume is required to be carried over these two paths, we can write x12 + x132 = h. (4.2.1a) This requirement is known as the demand flow constraint, or simply the demand constraint. It is clear that the variables cannot take negative values since a path may not carry any negative

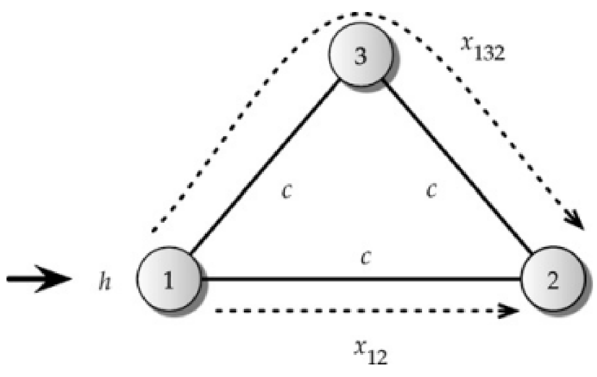


FIGURE 4.2 Single-commodity network flow modeling: three-node network.

demand; this means the lowest value that can be taken is zero. Thus, we include the following additional conditions on the variables: x12 ≥ 0, x132 ≥ 0. (4.2.1b) In addition, we need to address the capacity limit on each link. Certainly, any flow on a path due to routing cannot exceed the capacity on any of the links that this path uses.

An implicit assumption here is that the flow and the capacity are using the same measurement units; we will discuss deviations from this assumption in later chapters. Since we assume that the capacity limit is the same on all links in this three-node network, we can write x12 ≤ c, x132 ≤ c. (4.2.1c) The first one addresses the flow on the direct link 1-2 being less than its capacity; flow x132 uses two links 1-3 and 2-3, and we can use only a single condition here since the capacity is assumed to be the same on each link.

Constraints (4.2.1c) are called capacity constraints. From the above discussion, we can see that we need conditions (4.2.1a), (4.2.1b), and (4.2.1c) to define the basic system.

It is important to note that it is not a system of equations; while the first one, i.e., (4.2.1a), is an equation, the second and the third ones, i.e., (4.2.1b) and (4.2.1c), are inequalities.

Together, the system of equations and inequalities given by Eq. (4.2.1), which consists of conditions (4.2.1a), (4.2.1b), and (4.2.1c), is referred to as constraints of the problem. Even when all the constraints are known, our entire problem is not complete since we have not yet identified the goal of the problem.

In fact, without defining a goal, system (4.2.1) has infinite numbers of solutions since an infinite combination of values can be assigned to x12 and x132 that satisfies constraints

(4.2.1a), (4.2.1b), and (4.2.1c). As the first goal, we consider the cost of routing flows. To do that, we introduce a generic nonnegative cost per unit of flow on each path: $\xi12$ (≥ 0) for direct path 1-2 and $\xi132$ (≥ 0) for alternate path 1-3-2.
Thus, the total cost of the demand flow can be written as Total cost = $\xi12x12 + \xi132x132$.

The total cost is referred to as the objective function. In general, the objective function will be denoted by F. If the goal is to minimize the total cost of routing, we can write the complete problem as follows: minimize{x12, x132} $F = \xi12x12 + \xi132x132$ subject to $x12 + x132 = h$ $x12 \leq c$, $x132 \leq c$ $x12 \geq 0$, $x132 \geq 0$. (4.2.3) The problem presented in Eq. (4.2.3) is a single-commodity network flow problem; it is also referred to as a linear programming problem since the requirements given by Eq. (4.2.1) are all linear, which are either equations or inequalities, and the goal given by Eq. (4.2.2) is also linear.

In general, a representation as given in Eq. (4.2.3) is referred to as the formulation of an optimization problem. The system given by Eq. (4.2.1) is referred to as constraints. To avoid any confusion, we will identify the variables in any formulation by marking them as subscripts with minimize. Thus, in the above problem, we have noted that x12 and x132 are variables by indicating so as subscripts with minimize.

Often, the list of variables can become long; thus, we will also use a short notation such as x in the subscript with minimize to indicate that all xs are variables.

Because of the way the goal is described in Eq. (4.2.3), the problem is also known as the minimum cost routing or minimum cost network flow problem. An optimal solution to an optimization problem is a solution that satisfies the constraints of the problem, i.e., it is a feasible solution and the objective function value attained is the lowest (if it is a minimization problem) possible for any feasible solution.

For clarity, the optimal solution to a problem such as Eq. (4.2.3) will be denoted with asterisks in the superscript, for example, $x{*}$ 12 and $x{*}$ 132.

INSTANCE 1: We now consider some specific cases discussed earlier in Section 4.2.1 to obtain solutions to problem (4.2.1). First, we consider the capacity to be 10, i.e., $c = 10$. If the unit cost is based on a unit flow per link, then we can clearly write cost components as $\xi12 = 1$ (since it is a direct link path) and $\xi132 = 2$ (due to two links making a path). This will then correspond to the first case discussed in Section 4.2.1. In this case, optimal flows can be written as: $x^*_{12} = 10$, $x^*_{132} = 0$ when $0 \le h \le 10$ $x^*_{12} = 10$, $x^*_{132} = h - 10$ when $h \ge 10$, and $h \le 20$. (4.2.4) If $h > 20$, it is clear that the network does not have enough capacity to carry all of the demand volume—this is referred to as an infeasible situation and the problem is considered to be infeasible.

Variation in Objective: Load Balancing In model (4.2.3), we have considered the goal to be based on minimizing the routing cost by incorporating the unit cost of a path. While this is applicable in some communication networks, other goals or objectives are also applicable in other communication networks.

We now consider another goal—minimization of maximum link utilization. This goal is also referred to as load balancing flows in the network. To illustrate this, we will again use constraints (4.2.1) discussed above. The link utilization is defined as the amount of flow on a link divided by the capacity on that link. We know that the only flow using link 1-2 is x_{12} while x_{132} uses both links 1-3

VARIATION IN CAPACITY

We now consider a simple variation in which the link capacities in the network are not the same. To consider this case, we keep the capacity of link 1-2 at c but increase the capacity of the other two links to 10 times that of 1-2 , i.e., to 10 c. Note that the utilization on links 1-3 and 3-2 are now $x_{132}/(10\,c)$, and Formulation (4.2.6) changes to the following: minimize$\{x\}$ $F = \max \frac{x_{12}}{c}, \frac{x_{132}}{10\,c}$ subject to $x_{12} + x_{132} = h$ $x_{12} \le c$, $x_{132} \le 10\,c$ $x_{12} \ge 0$, $x_{132} \ge 0$.

Variation in Objective: Average Delay Another goal commonly defined, especially in data networks, is the minimization of the average packet delay. For this illustration, we consider again the three-node network with demand volume h between node 1 and node 2; the capacity of all links is set to c.

The average delay in a network (see Appendix B.13 for details) with flow x_{12} on the direct link path and x_{132} on the alternate path can be captured through the expression $\frac{x_{12}}{c - x_{12}} + \frac{2x_{132}}{c - x_{132}}$. Here again, the capacity is normalized so that the measurement units for flow and capacity are the same. The goal of minimizing the average delay is to solve the following problem: minimize$\{x\}$ $F = \frac{x_{12}}{c-x_{12}} + \frac{2x_{132}}{c-x_{132}}$ subject to $x_{12} + x_{132} = h$ $x_{12} \le c$, $x_{132} \le c$ $x_{12} \ge 0$, $x_{132} \ge 0$.

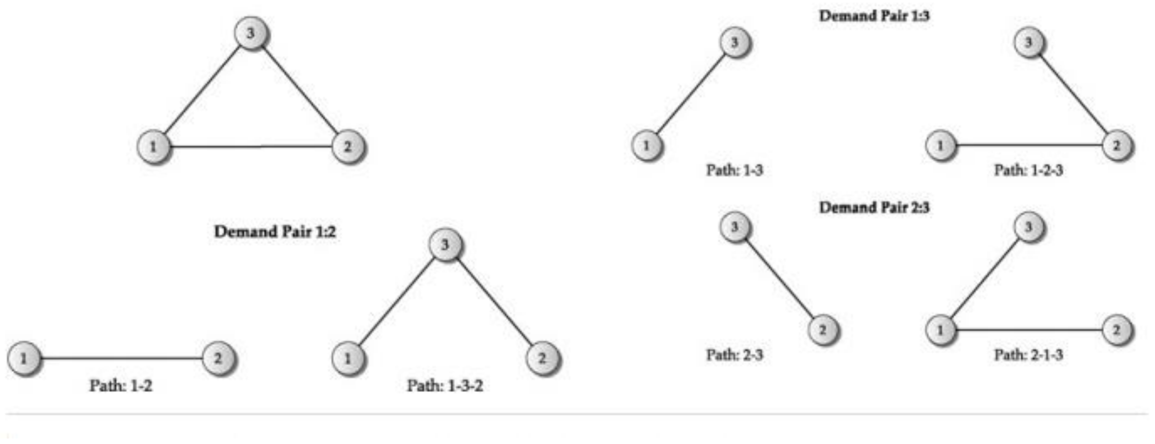
Multicommodity Network Flow

Multicommodity Network Flow: Three-Node Example In this section, we consider multiple commodities, that is, multiple demand pairs have positive demand volumes. As with the single-commodity case, we will consider again the three different objectives.

We will again use a three-node network to explain the multicommodity network flow problem. 4.3.1 Minimum Cost Routing Case For the multicommodity case in a three-node network, all three demand pairs can have positive demand volumes.

For clarity, we will use a subscript with demand volume notation h to identify different demands; thus, the demand volume between nodes 1 and 2 will be identified as h12, between 1 and 3 as h13, and between 2 and 3 as h23.

For each demand pair, the volume of demand can be accommodated using two paths: one is the direct link path and the other is the alternate path through the third node. In Figure 4.4, we show all the possible paths for each demand pair.
The amount of flow on each path is the unknown that is to be determined based on an objective; we denote the unknowns as x12 for path 1-2 for demand pair 1:2, and x132 for path 1-3-2 , and so on.



F I G U R E 4.4 Three node example with all possible paths.

Much as shown earlier for the single-commodity flow problem, we can write that the demand volume for a node pair must be carried over the two paths. Thus, for demand pair 1:2,

we can write $x12 + x132 = h12$. (4.3.1a)
Similarly, for demand pairs 1:3 and 2:3,
we can write the following: $x13 + x123 = h13$ (4.3.1b)
$x23 + x213 = h23$. (4.3.1c)
These unknown flow amounts, while satisfying the demand volume requirements, must also satisfy capacity limits on any link.

We denote capacities of links 1-2 , 1-3 , and 2-3 by c12, c13, and c23, respectively. By following the paths listed in Figure 4.4,

we note that three different paths from three different demand pairs use link 1-2; they are paths 1-2, 1-2-3, and 2-1-3 (see Figure 4.5).

Since the sum of the flow over these three paths cannot exceed the capacity, c12, of link 1-2 , we can write the following inequality (constraint): $x12 + x123 + x213 \leq c12$. (4.3.2a) Similarly, for the other two links 1-3 and 2-3, we can write $x13 + x132 + x213 \leq c13$ (4.3.2b) $x23 + x132 + x123 \leq c23$. (4.3.2c) We next consider the objective function for minimum cost routing.

If the unit costs of routing on paths 1-2, 1-3-2, 1-3, 1-2-3, 2-3, and 2-1-3 are denoted by $\xi12$, $\xi132$, $\xi13$, $\xi123$, $\xi23$, and $\xi213$, respectively,

then the total routing cost can be written as **total cost = $\xi12x12 + \xi132x132 + \xi13x13 + \xi123x123 + \xi23x23 + \xi213x213$**

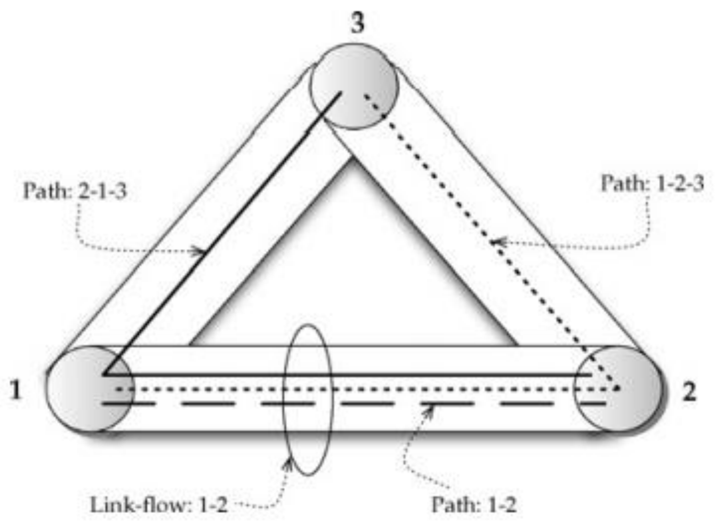


FIGURE 4.5 Link flow on link 1-2 for paths for different demand pairs.

Example 4.3 Multicommodity network flow with integer solution. Considering again demand volumes to be h12 = 5, h13 = 10, and h23 = 7, and capacities to be c12 = 10, c13 = 10, c23 = 15, in CPLEX,

we can enter the above ILP problem in the following way where integrity of variables is explicitly listed:

```
Minimize 2 x12 + x132 + 2 x13 + x123 + 2 x23 + x213
subject to
    d1: x12 + x132 = 5
    d2: x13 + x123 = 10
    d3: x23 + x213 = 7
    c1: x12 + x123 + x213 <= 10
    c2: x132 + x13 + x213 <= 10
   c3: x132 + x123 + x23 <= 15

   Bounds 0 <= x12 <= 10
    0 <= x132 <= 10
    0 <= x13 <= 10
   0 <= x123 <= 10
   0 <= x23 <= 10
   0 <= x213 <= 10
   Integer x12 x132 x13 x123 x23 x213
```

End An important point to note here is that when variables are explicitly declared as integers, an upper bound for these variables is required to be specified since CPLEX assumes that the default for the upper bound is 1.

In the above case, we have artificially set the upper bound at 10 since from demand volume values we know that no variables will take more than 10 units of flow at optimality. The optimal solution with integrality requirement is obtained as x^{*} 12 = 1, x^{*} 132 = 4, x^{*} 13 = 4, x^{*} 123 = 6, x^{*} 23 = 5, x^{*} 213 = 2 and the total cost at optimality is 32