Unicast Routing

Unit-III

Session Objectives

- To Introduce the concept of unicast routing
- To discuss common routing algorithms used in the Internet
- To explore unicast-routing protocols:RIP, OSPF, BGP

Session Outcomes

At the end of this session, participants will be able to

- Discuss the unicast routing concept and routing algorithms

Agenda

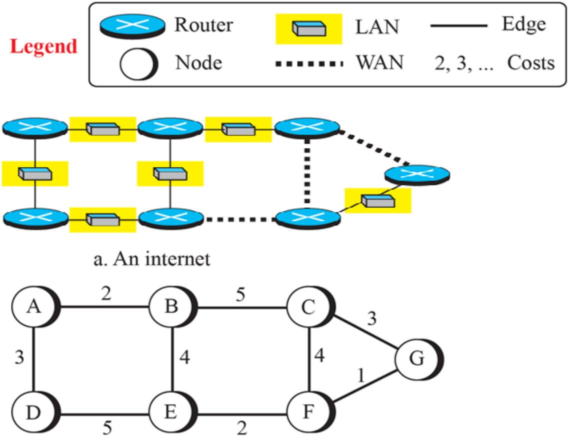
Presentation Outline

Introduction

- Unicast routing in 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.
- 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**
- Question is what other routers the packet should visit; which route the packet should tak

Internet as a Graph

- Each router as a node and each network between a pair of routers as an edge; **internet can be modeled as a weighted graph** in which each edge is associated with a cost
- If there is no edge between the nodes, the cost is infinity.

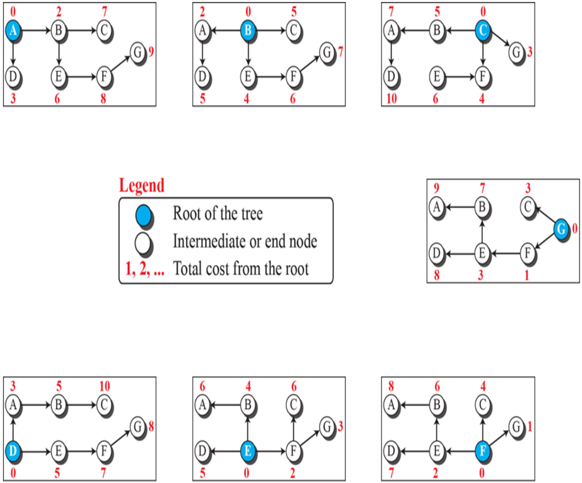


a. An internet

b. The weighted graph

Least-Cost Routing

- find the least cost between source router to the destination router
- N routers in an internet, then (N - 1) least-cost paths from each router to any other router. we need N x (N - 1) least-cost paths for the whole internet
- **Least-Cost Trees:**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
-


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

Least-cost trees properties

- 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;
- 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

Presentation Outline

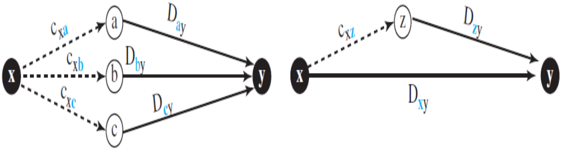
Distance-Vector(DV) Routing

- Each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbors.
- a router continuously tells all of its neighbors what it knows about the whole internet
- **Bellman-Ford Equation:** to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, . . .)

$$D_{xy} = \min\{(c_{xa} + D_{ay}), (c_{xb} + D_{by}), (c_{xc} + D_{cy}), \ldots\}$$

- D_{ij} is the shortest distance and c_{ij} is the cost between nodes i and j.

Distance-Vector(DV) Routing

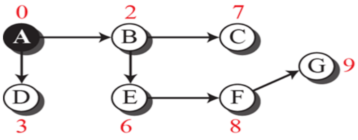


a. General case with three intermediate nodes

b. Updating a path with a new route

- Bellman-Ford equation enables us to build a new least-cost path from previously established least-cost paths
- we can think of (a−− >y), (b−− >y), and (c−− >y) as previously established least-cost paths and (x−− >y) as the new least-cost path.
- In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as z

Distance-Vector(DV) Routing



a. Tree for node A

	A
A	0
B	2
C	7
D	3
E	6
F	8
G	9

b. Distance vector for node A

- Name of the distance vector defines the root, the indexes define the destinations, and the value of each cell defines the least cost from the root to the destination.
- A distance vector **does not give the path** to the destinations as the least-cost tree does; it **gives only the least costs** to the destinations

First distance vector for an internet

- The node sends some greeting messages out of its interfaces and discovers the **identity of the immediate neighbors and the distance between itself and each neighbor.**
- these vectors are made asynchronously
- After each node has created its vector, it sends a copy of the vector to all its immediate neighbors. A
- After a node receives a distance vector from a neighbor, it updates its distance vector using the Bellman-Ford equation (second case)

$$D_{xy} = \min \{ D_{xy}, (c_{xz} + D_{zy}) \}$$

Updating distance vectors

	New B	Old B	A
A	2	2	0
B	0	0	2
C	5	5	∞
D	**5**	∞	3
E	4	4	∞
F	∞	∞	∞
G	∞	∞	∞

B[ ] = min (B[ ], 2 + A[ ])

a. First event: B receives a copy of A's vector.

Note:
X[ ]: the whole vector

	New B	Old B	E
A	2	2	∞
B	0	0	4
C	5	5	∞
D	5	5	5
E	4	4	0
F	**6**	∞	2
G	∞	∞	∞

B[ ] = min (B[ ], 4 + E[ ])

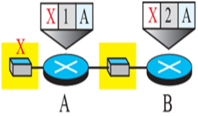
Distance-Vector Routing Algorithm for A Node

```
Distance_Vector_Routing ( )
{
    // Initialize (create initial vectors for the node)
    D[myself] = 0
    for (y = 1 to N)
    {
        if (y is a neighbor)
            D[y] = c[myself][y]
        else
            D[y] = ∞
    }
    send vector {D[1], D[2], …, D[N]} to all neighbors
    // Update (improve the vector with the vector received from a neighbor)
    repeat (forever)
    {
        wait (for a vector Dw from a neighbor w or any change in the link)
        for (y = 1 to N)
        {
            D[y] = min [D[y], (c[myself][w] + Dw[y])]        // Bellman-Ford equation
        }
        if (any change in the vector)
            send vector {D[1], D[2], …, D[N]} to all neighbors
    }
} // End of Distance Vector
```

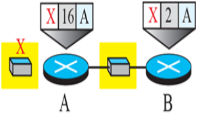
Count to Infinity

- Any **decrease in cost** (good news) propagates **quickly**, but any **increase in cost** (bad news) will propagate **slowly**.
- 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**.
- It sometimes **takes several updates** before the cost for a broken link is recorded as infinity by all routers

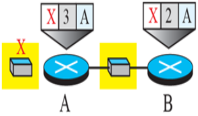
Two-Node Loop



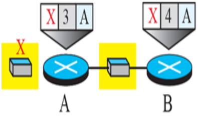
a. Before failure



b. After link failure

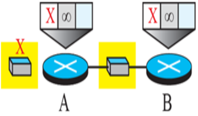


c. After A is updated by B



d. After B is updated by A

...



e. Finally

Split Horizon

- Instead of flooding the table through each interface, each node sends only part of its table through each interface.
- 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

Poison Reverse

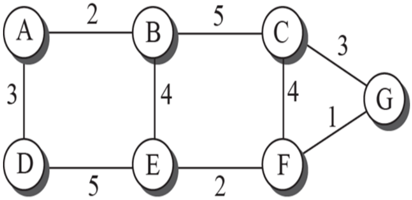
- Split-horizon strategy has one drawback; the corresponding protocol uses a timer, and if there is no news about a route, the node deletes the route from its table.
- In the poison reverse strategy B can still advertise the value for X, but if the source of information is A, it can replace the distance with infinity as a warning: "Do not use this value; what I know about this route comes from you."

Link-State Routing

- The **cost** associated with an edge defines the **state of the link**.
- Links with **lower costs are preferred** to links with higher costs;
- If the cost of a link is infinity, it means that the link does not exist or has been broken.
- Each node needs to have a complete map of the network, which means it needs to know the state of each link.
- The **collection of states for all links** is called the **link-state database (LSDB)**.
- There is only one LSDB for the whole internet; each node needs to have a duplicate of it

Link-State Routing

- The LSDB can be represented as a two-dimensional array(matrix) in which the value of each cell defines the cost of the corresponding link.



a. The weighted graph

	A	B	C	D	E	F	G
A	0	2	∞	3	∞	∞	∞
B	2	0	5	∞	4	∞	∞
C	∞	5	0	∞	∞	4	3
D	3	∞	∞	0	5	∞	∞
E	∞	4	∞	5	0	2	∞
F	∞	∞	4	∞	2	0	1
G	∞	∞	3	∞	∞	1	0

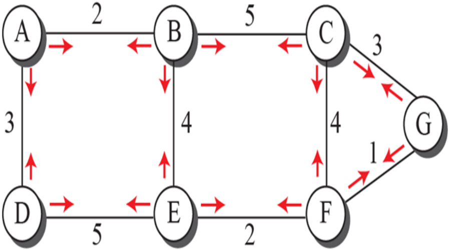
Flooding

- Link State info of neighbours are collected and tabulated
- LS Packets have sequence number that help to remove old info from their table

Node	Cost
B	2
D	3

Node	Cost
A	2
C	5
E	4

Node	Cost
B	5
F	4
G	3



Node	Cost
C	3
F	1

Node	Cost
A	3
E	5

Node	Cost
B	4
D	5

Node	Cost
C	4
E	2

Formation of Least-Cost Trees: Dijkstra Algorithm

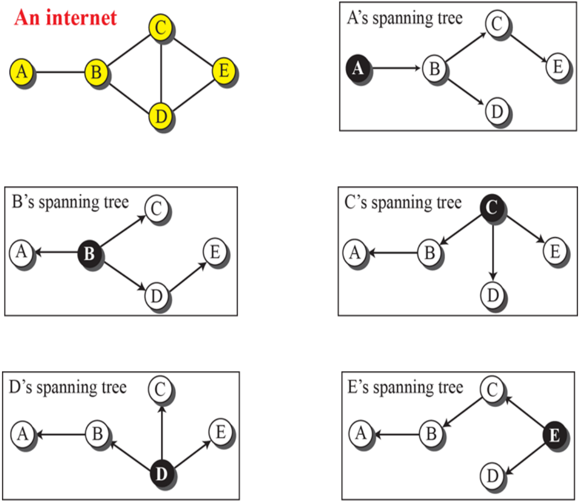
1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.
2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.
3. The node repeats step 2 until all nodes are added to the tree.

Dijkstra Algorithm


```
Dijkstra's Algorithm ( )
{
    // Initialization
    Tree = {root}                   // Tree is made only of the root
    for (y = 1 to N)                // N is the number of nodes
    {
        if (y is the root)
            D[y] = 0                // D[y] is shortest distance from root to node y
        else if (y is a neighbor)
            D[y] = c[root][y]       // c[x][y] is cost between nodes x and y in LSDB
        else
            D[y] = ∞
    }
    // Calculation
    repeat
    {
        find a node w, with D[w] minimum among all nodes not in the Tree
        Tree = Tree ∪ {w}           // Add w to tree
        // Update distances for all neighbor of w
        for (every node x, which is neighbor of w and not in the Tree)
        {
            D[x] = min{D[x], (D[w] + c[w][x])}
        }
    } until (all nodes included in the Tree)
} // End of Dijkstra
```

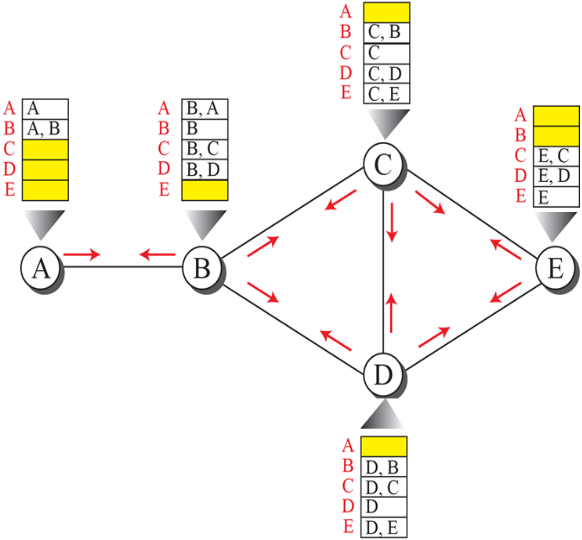
Path-Vector Routing

- When least-cost goal is not the priority; reach destination more efficiently without assigning costs to the route
- The source can control the path
- **Spanning Trees:** path from a source to all destinations is also determined by the best spanning tree.
- Each source has created its own spanning tree that meets its policy.
- The policy imposed by all sources is to use the minimum number of nodes to reach a destination.
- The spanning tree selected by A and E is such that the communication does not pass through D as a middle node.
- Similarly, the spanning tree selected by B is such that the communication does not pass through C as a middle node.


An internet
A
B
C
D
E
A's spanning tree
B's spanning tree
C's spanning tree
D's spanning tree
E's spanning tree

Creation of Spanning Trees

- Path-vector routing, is an asynchronous and distributed routing algorithm.
- When a node is booted, it creates a path vector based on the information it can obtain about its immediate neighbor.
- A node sends greeting messages to its immediate neighbors to collect these pieces of information


A
B
C
D
E
C, B
C
C, D
C, E
A
A, B
B, A
B
B, C
B, D
E, C
E, D
E
A
B
C
D
E
D, B
D, C
D
D, E

Path vectors

- Each node, after the creation of the initial path vector, sends it to all its immediate neighbors.
- Each node, when it receives a path vector from a neighbor, updates its path vector using an equation similar to the Bellman-Ford, but applying its own policy instead of looking for the least cost.

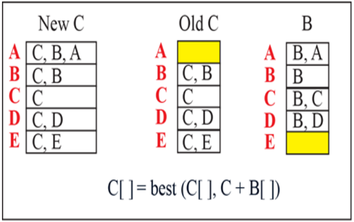
$$\text{Path}(\boldsymbol{x}, \boldsymbol{y}) = \text{best}\ \{\text{Path}(x, y), [(\boldsymbol{x} + \text{Path}(\mathbf{v}, \boldsymbol{y})]\} \qquad \text{for all } \mathbf{v}\text{'s in the internet.}$$

- If Path (v, y) includes x, that path is discarded to avoid a loop in the path

Creation of Spanning Trees

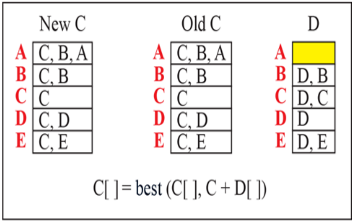
```
Path_Vector_Routing ( )
{
    // Initialization
    for (y = 1 to N)
    {
        if (y is myself)
            Path[y] = myself
        else if (y is a neighbor)
            Path[y] = myself + neighbor node
        else
            Path[y] = empty
    }
    Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    // Update
    repeat (forever)
    {
        wait (for a vector Path_w from a neighbor w)
        for (y = 1 to N)
        {
            if (Path_w includes myself)
                discard the path                    // Avoid any loop
            else
                Path[y] = best {Path[y], (myself + Path_w[y])}
        }
        If (there is a change in the vector)
            Send vector {Path[1], Path[2], …, Path[y]} to all neighbors
    }
} // End of Path Vector
```

Path-Vector Algorithm



Event 1: C receives a copy of B's vector

Note:
X [ ]: vector X
Y: node Y

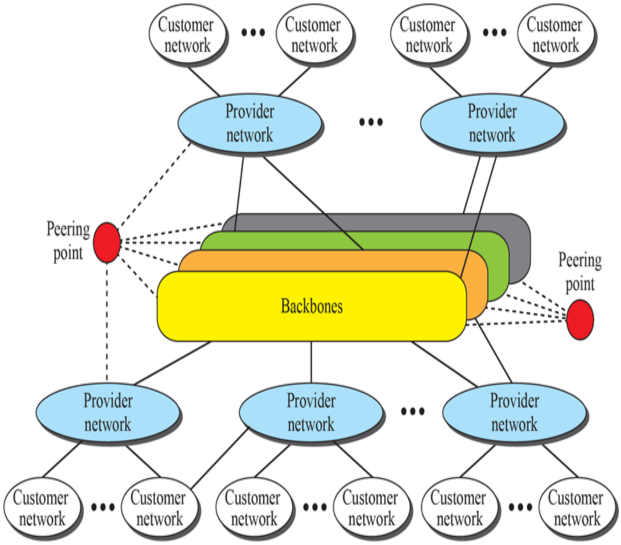


Event 2: C receives a copy of D's vector

UNICAST ROUTING PROTOCOLS

- There are several backbones run by private communication companies that provide global connectivity. These **backbones are connected by some peering points** that allow connectivity between backbones.
- At a lower level, there are some provider networks that use the backbones for global connectivity but provide services to Internet customers
- there are some customer networks that use the services provided by the provider networks.
- Any of these three entities **(backbone, provider network, or customer network) can be called an Internet Service Provider or ISP**. They provide services, but at different levels.

Internet structure



Hierarchical Routing

- Hierarchical routing means considering **each ISP as an autonomous system** (AS). Each AS can run a routing protocol that meets its needs, but the global Internet runs a global protocol to glue all ASs together.
- The **routing protocol run in each AS** is referred to as intra-AS routing protocol, intradomain routing protocol, or interior gateway protocol (IGP); the global routing protocol is referred to as inter-AS routing protocol, interdomain routing protocol, or exterior gateway protocol (EGP).
- The two common intradomain routing protocols are RIP and OSPF; the only interdomain routing protocol is BGP.

Autonomous Systems

- Each AS is given an autonomous number (ASN) by the ICANN - a 16-bit unsigned integer that uniquely defines an AS
- The autonomous systems, are categorized according to the way they are connected to other ASs.
- **Stub AS:** A stub AS has only one connection to another AS.
- The data traffic can be either initiated or terminated in a stub AS; the data cannot pass through it.
- A good example of a stub AS is the customer network, which is either the source or the sink of data.

Autonomous Systems

- **Multihomed AS:** A multihomed AS can have more than one connection to other ASs, but it does not allow data traffic to pass through it.
- A good example of such an AS is some of the customer ASs that may use the services of more than one provider network, but their policy does not allow data to be passed through them.
- **Transient AS:** A transient AS is connected to more than one other AS and also allows the traffic to pass through.
- The provider networks and the backbone are good examples of transient ASs.

Routing Information Protocol (RIP)

- Most widely used intradomain routing protocols based on the distance-vector routing algorithm
- RIP was started as part of the Xerox Network System (XNS),
- **Hop Count:**
- The cost is defined between a router and the network in which the destination host is located.
- Second, to make the implementation of the cost simpler, the **cost is defined as the number of hops**, which means the number of networks (subnets) a packet needs to travel
- the network in which the source host is connected is not counted

Routing Information Protocol

- The maximum cost of a path can be 15, which means 16 is considered as infinity (no connection). And hence, RIP can be used only in autonomous systems in which the diameter of the AS is not more than 15 hops.
- A forwarding table in RIP is a **three-column table** in which the first column is the address of the destination network, the second column is the address of the **next router** to which the packet should be forwarded, and the third column is the cost (**the number of hops**) to reach the destination network
- For example, R1 defines that the next router for the path to N4 is R2; R2 defines that the next router to N4 is R3; R3 defines that there is no next router for this path. The tree is then R1 –> R2 –> R3 –> N4.

Forwarding tables

Forwarding table for R1

Destination network	Next router	Cost in hops
N1	——	1
N2	——	1
N3	R2	2
N4	R2	3

Forwarding table for R2

Destination network	Next router	Cost in hops
N1	R1	2
N2	——	1
N3	——	1
N4	R3	2

Forwarding table for R3

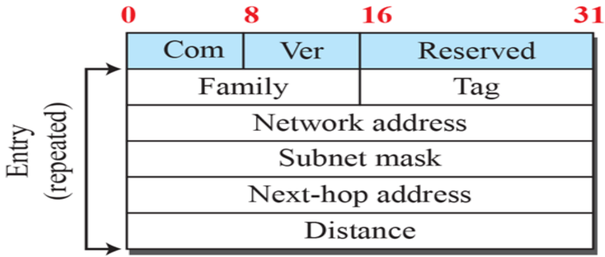
Destination network	Next router	Cost in hops
N1	R2	3
N2	R2	2
N3	——	1
N4	——	1

Routing Information Protocol

- RIP is implemented as a process that uses the service of UDP on the well-known port number 520.
- Runs as a daemon process, named routed (abbreviation for route daemon and pronounced route-dee).
- RIP messages are encapsulated inside UDP user datagrams, which in turn are encapsulated inside IP datagrams.
- In other words, RIP runs at the application layer, but creates forwarding tables for IP at the network later.
- Two versions 1 and 2 with backward compatibility

RIP Messages

- Two RIP processes, a client and a server, exchange messages.
- Each entry carries the information related to one line in the forwarding table of the router that sends the message.



Fields

Com: Command, request (1), response (2)
Ver: Version, current version is 2
Family: Family of protocol, for TCP/IP value is 2
Tag: Information about autonomous system
Network address: Destination address
Subnet mask: Prefix length
Next-hop address: Address length
Distance: Number of hops to the destination

RIP messages

- A request message is sent by a router that has just come up or by a router that has some time-out entries.
- A request message can ask about specific entries or all entries.
- A response (or update) message can be either solicited or unsolicited.
- A solicited response message is sent only in answer to a request message.
- It contains information about the destination specified in the corresponding request message.
- An unsolicited response message, is sent periodically, every 30 seconds or when there is a change in the forwarding table.

RIP Algorithm

RIP implements the same algorithm as the **distance-vector routing algorithm** with few changes

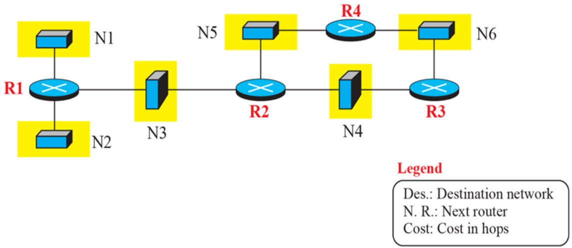
- Instead of sending only distance vectors, a **router needs to send the whole contents of its forwarding table** in a response message.
- The receiver **adds one hop to each cost and changes the next router field to the address of the sending router**.
- We call each route in the modified forwarding table the received route and each route in the old forwarding table the old route.
- The new forwarding table needs to be sorted according to the destination route

RIP Algorithm

The received router selects the old routes as the new ones except in the following three cases:

1. If the received route **does not exist** in the old forwarding table, it should be added to the route.
2. If the cost of the **received route is lower** than the cost of the old one, the received route should be selected as the new one.
3. If the cost of the **received route is higher than the cost of the old one, but the value of the next router is the same** in both routes, the received route should be selected as the new one.

Think “Outside the Box”



R1

Des.	N. R.	Cost
N1	———	1
N2	———	1
N3	———	1

R2

Des.	N. R.	Cost
N3	———	1
N4	———	1
N5	———	1

R3

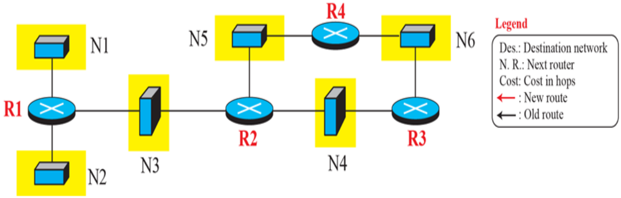
Des.	N. R.	Cost
N4	———	1
N6	———	1

R4

Des.	N. R.	Cost
N5	———	1
N6	———	1

Forwarding tables after all routers booted

Think "Outside the Box"



New R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1
N4	R2	2
N5	R2	2

Old R1

Des.	N. R.	Cost
N1	—	1
N2	—	1
N3	—	1

R2 Seen by R1

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R3

Des.	N. R.	Cost
N3	R2	2
N4	—	1
N5	R2	2
N6	—	1

Old R3

Des.	N. R.	Cost
N4	—	1
N6	—	1

R2 Seen by R3

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

New R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	—	1
N6	—	1

Old R4

Des.	N. R.	Cost
N5	—	1
N6	—	1

R2 Seen by R4

Des.	N. R.	Cost
N3	R2	2
N4	R2	2
N5	R2	2

Changes in the forwarding tables of R1, R3, and R4 after they receive a copy of R2's table

Timers in RIP

RIP uses three timers to support its operation

- **Periodic timer** controls the advertising of regular update messages. (between 25 and 35 seconds)
- The **expiration timer** governs the validity of a route. (180 seconds)
- **Garbage collection timer** is used to purge a route from the forwarding table.(120 seconds)

Summary

Discussed about

- Concept of unicast routing
- Common routing algorithms used in the Internet
- Unicast-routing protocols:RIP, OSPF, BGP

Test your Understanding

- The term that is used to place packet in its route to its destination is called ________
  a) Delayed
  b) Urgent
  c) Forwarding
  d) Delivering
- In Unicast routing, if instability is between three nodes, stability cannot be
  a) Stable
  b) Reversed
  c) **Guaranteed**
  d) Forward
- . In Unicast Routing, Dijkstra algorithm creates a shortest path tree from a ________
  a) Graph
  b) **Tree** c) Network d) Link