Routing Protocols - Basics

- The routers in an internet are responsible for receiving and forwarding packets through the interconnected set of networks.
- Each router makes routing decision based on knowledge of the topology and traffic/delay conditions of the internet.
- To make dynamic routing decisions, routers exchange routing information using special routing protocols.
- Routing protocols can be either interior routing protocols or exterior routing protocols. Interior routing protocols are used to share routing information within an autonomous system; each AS may use a different interior routing protocol because the system is autonomous. Exterior routing protocols convey routing data between autonomous systems; each AS must use the same exterior protocol to ensure that they can communicate.
- Another key differentiation of routing protocols is on the basis of the algorithms and metrics they use. An algorithm refers to a method that the protocol uses for determining the best route between any pair of networks, and for sharing routing information between routers. A metric is a measure of 'cost' that is used to assess the efficiency of a particular route.
- Routing protocols employ two of the most commonly used routing protocols algorithms for gathering and using routing information. They are:
  - Distance vector routing algorithm , and
  - Link state routing algorithm.

Distance Vector Routing

- Distance vector(DV) routing algorithm, also called a Bellman Ford algorithm, is one where routes are selected based on the distance between networks.
- The distance metric is simple which is usually the number of 'hops', or routers between them.
- Distance vector routing requires that each node (router or host that implements the routing protocol) exchange information with its neighboring nodes. Two nodes are said to be neighbors if they are both directly connected to the same network.
- Each router sends a distance vector to all of its neighbors, and that vector contains the estimated path cost to all networks in the configuration.
- These routers then update their tables and send to their neighbors. This causes distance information to propagate across the internetwork, so that eventually each router obtains distance information about all networks on the internet.
- Furthermore, when there is a significant change in a link cost or when a link is unavailable, it may take a considerable amount of time for this information to propagate through the internet.
- The distance vector algorithm is iterative and distributed. It is distributed in a sense that each node receives some information from one or more of its directly attached neighbors, performs a calculation, and then distributes the results of its calculation back to its neighbors. It is iterative in a sense that process continues on until no more information is exchanged between neighbors.

Distance Vector Routing

- In DV algorithm each node x begins with $D_x(y)$, an estimate of the cost of the least cost path from itself to node y, for all nodes in N.
- Let $Dx = [D_x(y): y\ in\ N]$ be node x distance vector, which is the vector of cost estimates from x to all other nodes, y, in N.
- With the DV algorithm, each node x maintains the following routing information:
- For each neighbor v, the cost $c(x, v)$ from x to directly connected neighbor, v
- Node $x's$ distance vector, that is, $Dx = [D_x(y): y\ in\ N]$, containing $x's$ estimate of its cost to all destinations, y, in N.
- The distance vectors of each of its neighbors, that is, $Dv = [D_v(y): y\ in\ N]$ for each neighbor v of x.
- In the distributed algorithm, from time to time, each node sends a copy of its distance vector to each of its neighbors. When a node x receives a new distance vector from any of its neighbors v, it saves $v's$ distance vector, and then uses the Bellman Ford equation to update its own distance vector as follows:

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

Distance Vector 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 destinations y in N

for each neighbor w

send distance vector $Dx = [D_x(y): y \; in \; N]$ to w

loop

wait (until there is a change in link cost to some neighbor w or until a distance vector from some neighbor w is received)

for each y in N:

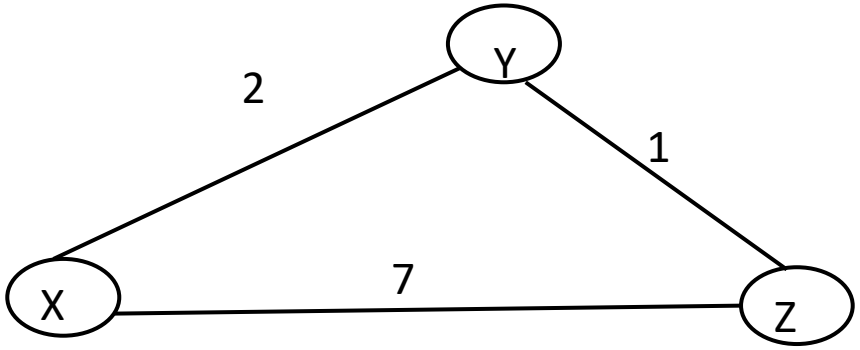
$$D_x(y) = \; min_v\{c(x,v) + \; D_v(y)\}$$

if $D_x(y)$ changed for any destination y

send distance vector $Dx = [D_x(y): y \; in \; N]$ to all neighbors

forever

Execution of Distance Vector Algorithm



Initial State

Cost to

From

D_x	X	Y	Z
X	0	2	7
Y	∞	∞	∞
Z	∞	∞	∞

$D_x(x) = 0$
$D_x(y) = 2$
$D_x(z) = 7$

Cost to

From

D_y	X	Y	Z
X	∞	∞	∞
Y	2	0	1
Z	∞	∞	∞

$D_y(x) = 2$
$D_y(y) = 0$
$D_y(z) = 1$

Cost to

From

D_z	X	Y	Z
X	∞	∞	∞
Y	∞	∞	∞
Z	7	1	0

$D_z(x) = 7$
$D_z(y) = 1$
$D_z(z) = 0$

Execution of Distance Vector Algorithm

- In the initial routing tables for each node each row is a distance vector—specifically, each node's routing table includes its own distance vector and that of each of its neighbors.
- Because at initialization node x has not received anything from node y or z, the entries in the second and third rows are initialized to infinity.
- After initialization, each node sends its distance vector to each of its two neighbors. After receiving the updates, each node recomputes its own distance vector.

Execution of Distance Vector Algorithm

Node 'y' and 'z' send their distance vector to 'x'

$$D_x(x) = 0$$
$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2$$
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3$$

Cost to

From D_x	X	Y	Z
X	0	2	3
Y	2	0	1
Z	7	1	0

Node 'x' and 'z' send their distance vector to 'y'

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{2+0, 1+7\} = 2$$
$$D_y(y) = 0$$
$$D_y(z) = \min\{c(y,z) + D_z(z), c(y,x) + D_x(z)\} = \min\{1+0, 2+7\} = 1$$

Cost to

From D_y	X	Y	Z
X	0	2	7
Y	2	0	1
Z	7	1	0

Node 'x' and 'y' send their distance vector to 'z'

$$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\} = \{7+0, 1+2\} = 3$$
$$D_z(y) = \min\{c(y,z) + D_y(y), c(z,x) + D_x(y)\} = \{1+0, 7+2\} = 1$$
$$D_z(z) = 0$$

Cost to

From D_z	X	Y	Z
X	0	2	7
Y	2	0	1
Z	3	1	0

Execution of Distance Vector Algorithm

- After the nodes recompute their distance vectors, they again send their updated distance vectors to their neighbors (if there has been a change).
- Only nodes x and z send updates and node y's distance vector didn't change so node y doesn't send an update.
- After receiving the updates, the nodes then recompute their distance vectors and update their routing tables.

Execution of Distance Vector Algorithm

Node 'z' sends its distance vector to 'x'

$$D_x(x) = 0$$
$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{2+0, 7+1\} = 2$$
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{2+1, 7+0\} = 3$$

Cost to

From D_x	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

Node 'x' sends its distance vector to 'z'

$$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\} = \{7+0, 1+2\} = 3$$
$$D_z(y) = \min\{c(z,y) + D_y(y), c(z,x) + D_x(y)\} = \{1+0, 7+2\} = 1$$
$$D_z(z) = 0$$

Cost to

From D_z	X	Y	Z
X	0	2	3
Y	2	0	1
Z	3	1	0

Execution of Distance Vector Algorithm

- The process of receiving updated distance vectors from neighbors, recomputing routing table entries, and informing neighbors of changed costs of the least cost path to a destination continues until no update messages are sent.
- At this point, since no update messages are sent, no further routing table calculations will occur and the algorithm will converge and nodes will be performing the waiting until a link cost changes.
- Routing tables for each node will be:

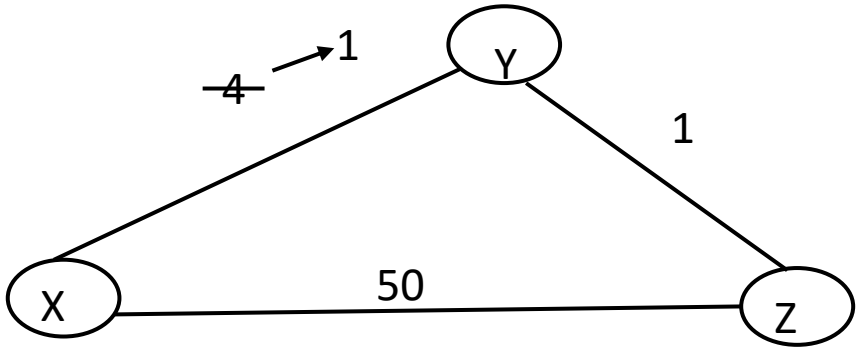
Node X		
Dst	Cost	Outgoing Link
X	0	-
Y	2	X-Y
Z	3	X-Y

Node Y		
Dst	Cost	Outgoing Link
X	2	Y-X
Y	0	-
Z	1	Y-Z

Node Z		
Dst	Cost	Outgoing Link
X	3	Z-Y
Y	1	Z-Y
Z	0	-

Distance Vector Algorithm – Link Cost Change

- When a node running the DV algorithm detects a change in the link cost from itself to a neighbor , it updates its distance vector and, if there's a change in the cost of the least cost path, informs its neighbors of its new distance vector.
- Before the link cost changes, the routing tables at each node will be:
- Now link cost changes from 4 → 1



Cost to

From	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

Distance Vector Algorithm – Link Cost Change

- At time t_0 , y detects the link cost change and updates its distance vector, and informs its neighbors of this change.

- At time t_1 , z and x receive the update from y and update their tables.

$$D_x(x) = 0$$
$$D_x(y) = \min\{c(x,y) + D_y(y)\,, c(x,z) + D_z(y)\} = \min\{1+0\,, 50+1\} = 1$$
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{1+1, 50+0\} = 2$$

$$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\} = \{50+0\,, 1+1\} = 2$$
$$D_z(y) = \min\{c(z,y) + D_y(y), c(z,x) + D_x(y)\} = \{1+0\,, 50+1\} = 1$$
$$D_z(z) = 0$$

Cost to

From

D_y	X	Y	Z
X	0	4	5
Y	1	0	1
Z	5	1	0

Cost to

From

D_x	X	Y	Z
X	0	1	2
Y	1	0	1
Z	5	1	0

Cost to

From

D_z	X	Y	Z
X	0	4	5
Y	1	0	1
Z	2	1	0

Distance Vector Algorithm – Link Cost Change

- At time t_2 , x and z send their updated distance vectors to neighbors and they recompute their routing table entries.

$$D_x(x) = 0$$
$$D_x(y) = \min\{c(x,y) + D_y(y), c(x,z) + D_z(y)\} = \min\{1+0, 50+1\} = 1$$
$$D_x(z) = \min\{c(x,y) + D_y(z), c(x,z) + D_z(z)\} = \min\{1+1, 50+0\} = 2$$

Cost to

From D_x	X	Y	Z
X	0	1	2
Y	1	0	1
Z	2	1	0

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{1+0, 1+2\} = 1$$
$$D_y(y) = 0$$
$$D_y(z) = \min\{c(y,z) + D_z(z), c(y,x) + D_x(z)\} = \min\{1+0, 1+2\} = 1$$

Cost to

From D_y	X	Y	Z
X	0	1	2
Y	1	0	1
Z	2	1	0

$$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\} = \{50+0, 1+1\} = 2$$
$$D_z(y) = \min\{c(z,y) + D_y(y), c(z,x) + D_x(y)\} = \{1+0, 50+1\} = 1$$
$$D_z(z) = 0$$

Cost to

From D_z	X	Y	Z
X	0	1	2
Y	1	0	1
Z	2	1	0

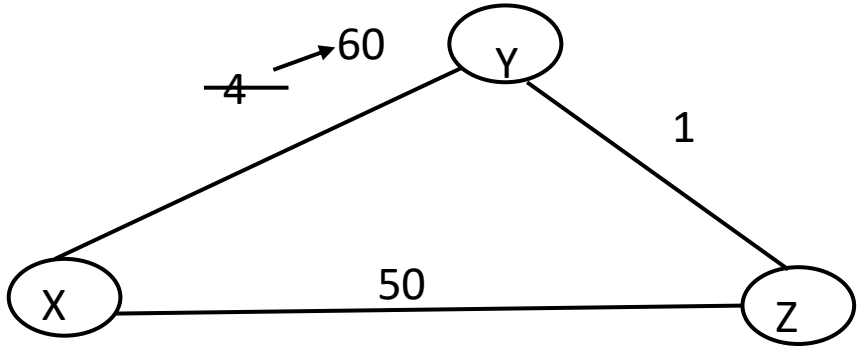
Distance Vector Algorithm – Link Cost Change

- None of the routing entries have been updated and updates will not be sent to neighbors. Thus two iterations are required for the DV algorithm to converge. The good news about the decreased cost between x and y has propagated quickly through the network.
- Before the link cost changes, the routing tables at each node will be:

Cost to

From	X	Y	Z
X	0	4	5
Y	4	0	1
Z	5	1	0

- Now consider that link cost between x and y has increased.



Distance Vector Algorithm – Link Cost Change

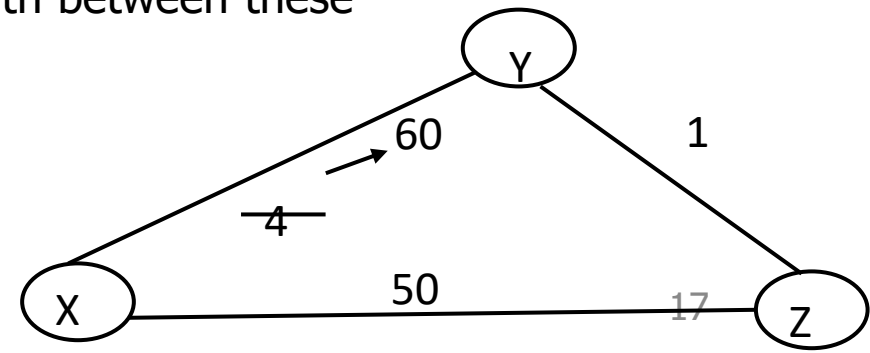
- At time t_0 , y detects the link cost change and updates its distance vector,

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60 + 0, 1 + 5\} = 6$$
$$D_y(y) = 0$$
$$D_y(z) = \min\{c(y,z) + D_z(z), c(y,x) + D_x(z)\} = \min\{1 + 0, 60 + 5\} = 1$$

Cost to

From D_y	X	Y	Z
X	0	4	5
Y	6	0	1
Z	5	1	0

- With the global view of the network, we can see that this new cost via z is wrong.
- But the only information node y has is that its direct cost to x is 60 and that z has last told y that z could get to x with a cost of 5.
- So in order to get to x, y would now route through z, fully expecting that z will be able to get to x with a cost of 5.
- As of t_1 we have a routing loop—in order to get to x, y routes through z, and z routes through y.
- A packet destined for x arriving at y or z at t_1 will bounce back and forth between these two nodes forever (or until the forwarding tables are changed).



Distance Vector Algorithm – Link Cost Change

- At time t_1 , y informs its neighbors of its new distance vector.
- Sometime after t_1, z receives y's new distance vector, which indicates that y's minimum cost to x is 6.
- z knows it can get to y with a cost of 1 and hence computes a new least cost to x:

$$D_z(x) = \min\{c(z,x) + D_x(x), c(z,y) + D_y(x)\} = \{50 + 0\ , 1 + 6\} = 7$$

- Since z's least cost to x has increased, it then informs y of its new distance vector at t_2.
- In a similar manner, after receiving z's new distance vector, y determines:

$$D_y(x) = \min\{c(y,x) + D_x(x), c(y,z) + D_z(x)\} = \min\{60+, 1 + 7\} = 8$$

and sends z its distance vector. z then determines $D_z(x) = 9$ and sends y its distance vector, and so on.

Cost to

From D_z	X	Y	Z
X	0	4	5
Y	6	0	1
Z	7	1	0

Cost to

From D_y	X	Y	Z
X	0	4	5
Y	8	0	1
Z	7	1	0

Distance Vector Algorithm – Link Cost Change

- The loop will persist i.e. message exchanges between y and z until z eventually computes the cost of its path to x via y to be greater than 50.
- At this point, z will determine that its least cost path to x is via its direct connection to x. y will then route to x via z.
- The result of the bad news about the increase in link cost has indeed traveled slowly. This problem is referred as count to infinity problem.

Link State Algorithm

- Link-state routing is designed to overcome the drawbacks of distance vector routing.
- When a router is initialized, it determines the link cost on each of its network interfaces. The router then advertises this set of link costs(Link state advertisements) to all other routers in the internet topology, not just neighboring routers.
- Every router monitors its link costs and whenever there is a significant change (link cost increases or decreases substantially, a new link is created, an existing link becomes unavailable), the router again advertises its set of link costs to all other routers in the configuration.
- Each router receives the link costs of all routers in the configuration and thus can construct the topology of the entire configuration and then calculate the shortest path to each destination network. In practice, Dijkstra's algorithm is used to calculate the shortest path. The router's routing table lists the first hop to each destination.
- Because the router has a representation of the entire network, it does not use a distributed version of a routing algorithm, as is done in distance vector routing.
- Link state algorithms adapt dynamically to changing internetwork conditions, and also allow routes to be selected based on more realistic metrics of cost than simply the number of hops between networks.

LS Algorithm VS DV Algorithm

- In the DV algorithm, each node talks to only its directly connected neighbors, and provides its neighbors with least cost estimates from itself to all the nodes in the network.
- In the LS algorithm, each node talks with all other nodes , but it tells them only the costs of its directly connected links.
- LS requires each node to know the cost of each link in the network. Also, whenever a link cost changes, the new link cost must be sent to all nodes. The DV algorithm requires message exchanges between directly connected neighbors at each iteration.
- LS is an $O(N)^2$ algorithm whereas DV algorithm can converge slowly and can have routing loops while the algorithm is converging. DV also suffers from the count to infinity problem.