PUNE INSTITUTE OF COMPUTER TECHNOLOGY, PUNE

ACADEMIC YEAR: 2022-23

DEPARTMENT of COMPUTER ENGINEERING DEPARTMENT

CLASS: T.E. **SEMESTER: I**

SUBJECT: CNSL

ASSINGMENT NO.	B2
TITLE	Write a program to implement Link state / Distance vector routing protocol to find suitable path for transmission
PROBLEM STATEMENT /DEFINITION	1. Write a program to implement Link state routing protocol to find suitable path for transmission 2. Write a program to implement Distance vector state routing protocol to find suitable path for transmission
OBJECTIVE	Learn and understand behavior of link state routing algorithm Implementation of Link state strategy to find the appropriate path from source to destination
OUTCOME	Demonstrate the Study Link state and Distance vector method of routing
S/W PACKAGES AND HARDWARE APPARATUS USED	PC with the configuration as Pentium IV 1.7 GHz. 128M.B RAM, 40 G.B HDD, 15''Color Monitor, Keyboard, Mouse. Operating Systems (64-Bit)64-BIT Fedora 20 or latest 64-BIT Update of Equivalent Open source OS Programming Tools (64-Bit) GCC/G++
REFERENCES	Mansfield K., Antonakos J., "An Introduction to Computer Networking", Perason Education, 2002, ISBN 81 - 7808 - 828 – 2 Fourauzan B., "Data Communications and Networking", 5 th Edition, Tata McGraw- Hill, Publications, ISBN: 0 – 07 – 058408 – 7
STEPS	Refer to student activity flowchart
INSTRUCTIONS FOR WRITING JOURNAL	1. Date 2. Assignment no. 3. Problem definition 4. Learning objective 5. Learning Outcome 6. Concepts related Theory 7. Algorithm 8. Test cases 10. Conclusion/Analysis

Prerequisites: Network layer functions, Routing and forwarding

Concepts related Theory:

1. Distance vector and Link State Routing Protocol

Link-state routing protocols are one of the two main classes of routing protocols used in packet switching networks for computer communications, the other being distance-vector routing protocols. Examples of link-state routing protocols include Open Shortest Path First (OSPF) and intermediate system to intermediate system (IS-IS).

The link-state protocol is performed by every switching node in the network (i.e., nodes that are prepared to forward packets; in the Internet, these are called routers). The basic concept of link-state routing is that every node constructs a map of the connectivity to the network, in the form of a graph, showing which nodes are connected to which other nodes. Each node then independently calculates the next best logical path from it to every possible destination in the network. Each collection of best paths will then form each node's routing table.

2. Calculating the shortest paths

Each node independently runs an algorithm over the map to determine the shortest path from itself to every other node in the network; generally some variant of Dijkstra's algorithmis used. This is based around a link cost across each path which includes available bandwidth among other things. A node maintains two data structures: a tree containing nodes which are "done", and a list of candidates. The algorithm starts with both structures empty; it then adds to the first one the node itself. The variant of a Greedy Algorithm then repetitively does the following:

All neighbour nodes which are directly connected to the node are just added to the tree (exaccepting any nodes which are already in either the tree or the candidate list). The rest areadded to the second (candidate) list.

Each node in the candidate list is compared to each of the nodes already in the tree. The candidate node which is closest to any of the nodes already in the tree is itself moved into the tree and attached to the appropriate neighbor node. When a node is moved from the candidate list into the tree, it is removed from the candidate list and is not considered in subsequent iterations of the algorithm.

The above two steps are repeated as long as there are any nodes left in the candidate list. (When there are none, all the nodes in the network will have been added to the tree.) This procedure ends with the tree containing all the nodes in the network, with the node on which the algorithm is running as the root of the tree. The shortest path from that node to any other node is indicated by the list of nodes one traverses to get from the root of the tree to the desired node in the tree.

3. Filling the routing table

With the shortest paths in hand, the next step is to fill in the routing table. For any given

destination node, the best path for that destination is the node which is the first step from the root node, down the branch in the shortest-path tree which leads toward the desired destination node. To create the routing table, it is only necessary to walk the tree, remembering the identity of the node at the head of each branch, and filling in the routing table entry for each node one comes across with that identity.

Algorithm

Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has minimum distance from source.

1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.

2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.

3) While sptSet doesn't include all vertices

a) Pick a vertex u which is not there in sptSetand has minimum distance value.

b) Include u to sptSet.

c) Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if sum of distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

Distance Vector Routing (DVR) Protocol

A distance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Historically known as the old ARPANET routing algorithm (or known as

Bellman-Ford algorithm).

Bellman Ford Basics – Each router maintains a Distance Vector table containing the distance between itself and ALL possible destination nodes. Distances,based on a chosen metric, are computed using information from the neighbors' distance vectors.

Information kept by DV router -

Each router has an ID Associated with each link connected to a router, there is a link cost (static or dynamic). Intermediate hops

Distance Vector Table Initialization -

Distance to itself = 0

Distance to ALL other routers = infinity number.

Distance Vector Algorithm –

A router transmits its distance vector to each of its neighbors in a routing packet.

Each router receives and saves the most recently received distance vector from each of its neighbors.

A router recalculates its distance vector when:

It receives a distance vector from a neighbor containing different information than before.

It discovers that a link to a neighbor has gone down.

The DV calculation is based on minimizing the cost to each destination

Dx(y) = Estimate of least cost from x to y

C(x,v) = Node x knows cost to each neighbor v

Dx = [Dx(y): y ∈ N] = Node x maintains distance vector

Node x also maintains its neighbors' distance vectors

– For each neighbor v, x maintains Dv = [Dv(y): y ∈ N]

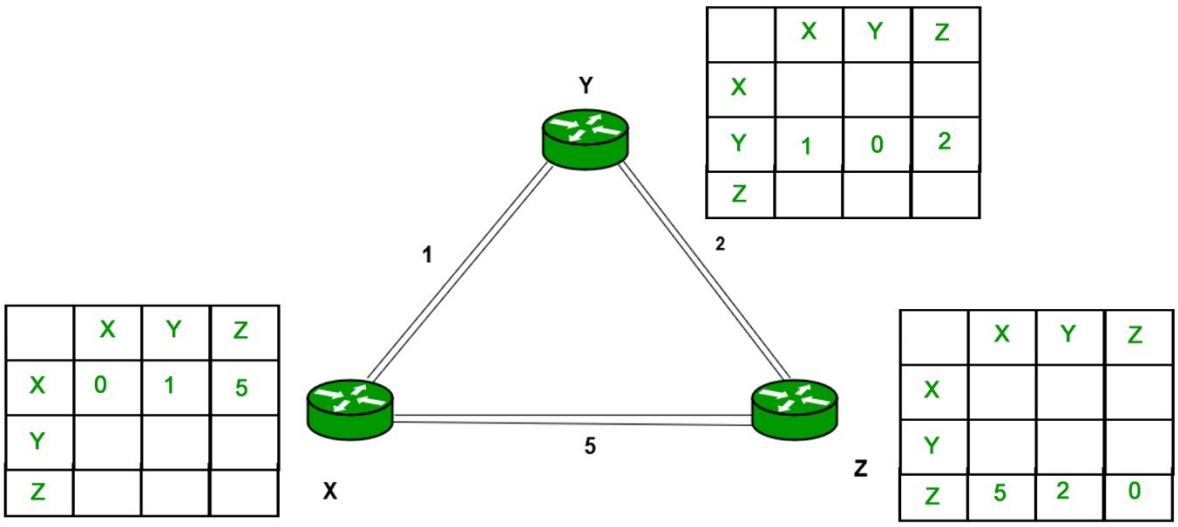
Note –From time-to-time, each node sends its own distance vector estimate to neighbors.

When a node x receives new DV estimate from any neighbor v, it saves v's distance vector and it updates its own DV using B-F equation:

Dx(y) = min { C(x,v) + Dv(y), Dx(y) } for each node y ∈ N

Example – Consider 3-routers X, Y and Z as shown in figure. Each router have their routing

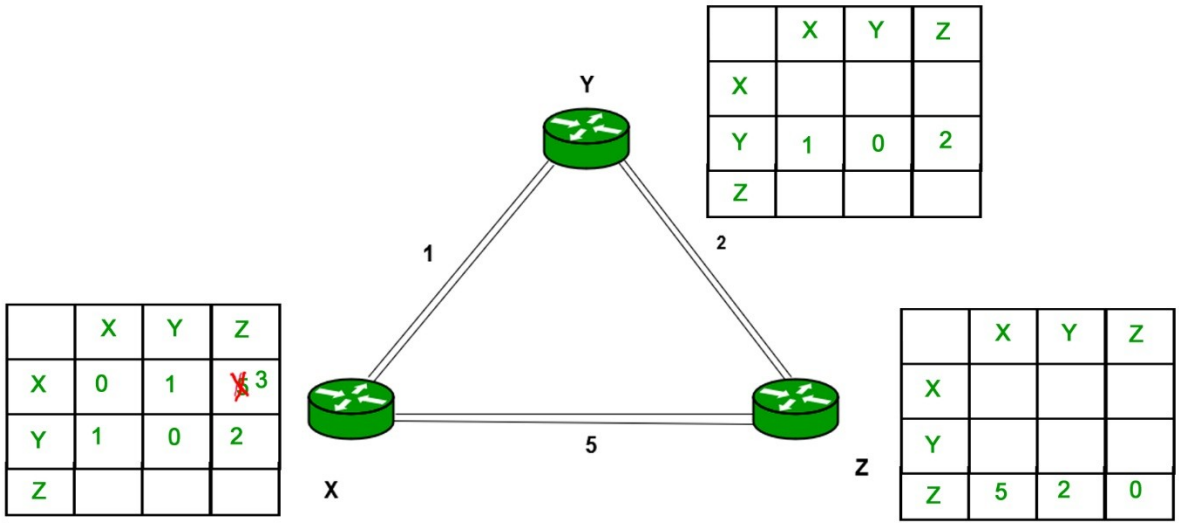
table. Every routing table will contain distance to the destination nodes.



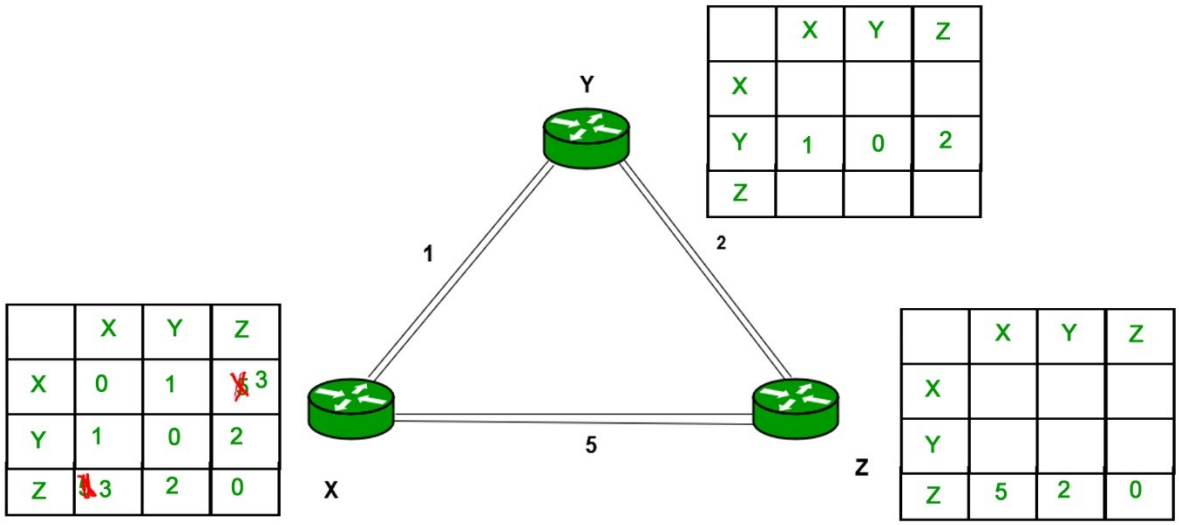
Consider router X , X will share it routing table to neighbors and neighbors will share it routing table to it to X and distance from node X to destination will be calculated using bellmen- ford equation.

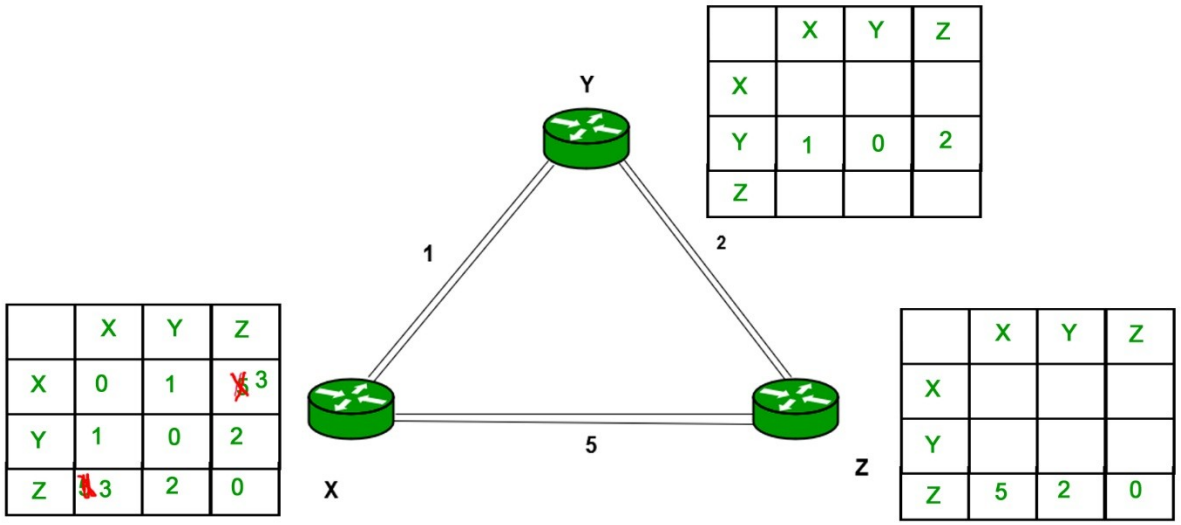
$D_x(y) = \min \{ C(x,v) + D_v(y)\}$ for each node $y \in N$

As we can see that distance will be less going from X to Z when Y is intermediate node(hop) so it will be update in routing table X.

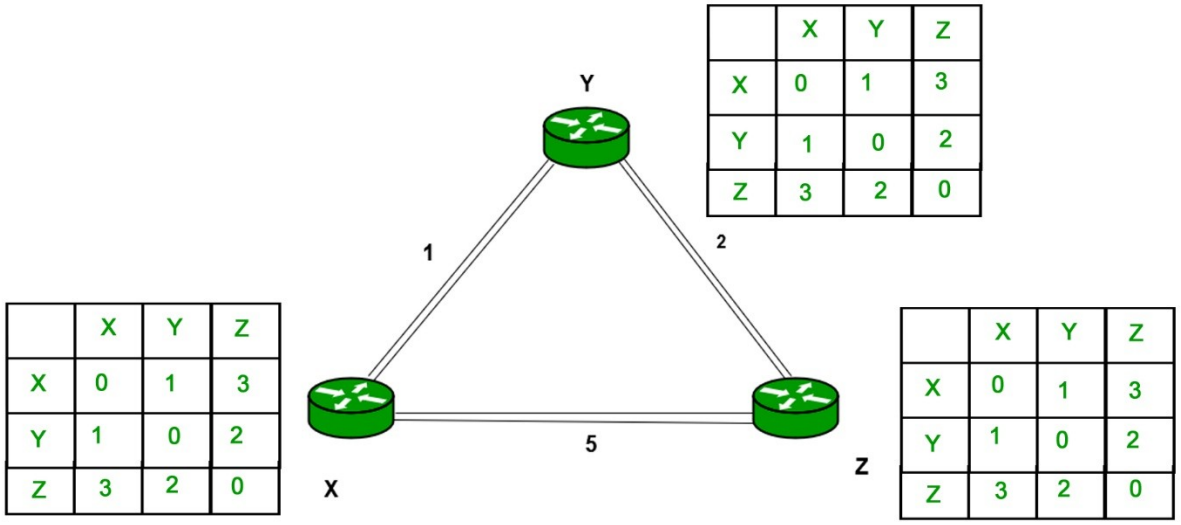


Similarly for Z also –





Finally the routing table for all –



Advantages of Distance Vector routing –

- It is simpler to configure and maintain than link state routing.

Disadvantages of Distance Vector routing –

- It is slower to converge than link state.
- It is at risk from the count-to-infinity problem.

- It creates more traffic than link state since a hop count change must be propagated to all routers and processed on each router. Hop count updates take place on a periodic basis, even if there are no changes in the network topology, so bandwidth-wasting broadcasts still occur.
- For larger networks, distance vector routing results in larger routing tables than link state since each router must know about all other routers. This can also lead to congestion on WAN links.

Note – Distance Vector routing uses UDP(User datagram protocol) for transportation.

Review Questions:

1. What is routing?
2. What is distance vector and link state routing? Differentiate.
3. What is RIP?
4. What is OSPF?