**Assignment No. :**

**Title:** Distance Vector Routing Algorithm

**Aim**: Implement Distance Vector Routing algorithm using C++

**Objective:**To study and understand network routing algorithm

**Theory:**

In computer communication theory relating to packet-switched networks, a distance-vector routing protocol is one of the two major classes of routing protocols, the other major class being the link-state protocol. A distance-vector routing protocol uses the Bellman-Ford algorithm to calculate paths. A distance-vector routing protocol requires that a router informs its neighbours of topology changes periodically and, in some cases, when a change is detected in the topology of a network. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. Distance Vector means that Routers are advertised as vector of distance and direction. 'Direction' is represented by next hop address and exit interface, whereas 'Distance' uses metrics such as hop count. Routers using distance vector protocol do not have knowledge of the entire path to a destination. Instead DV uses two methods:

1. Direction in which or interface to which a packet should be forwarded.
2. Distance from its destination.

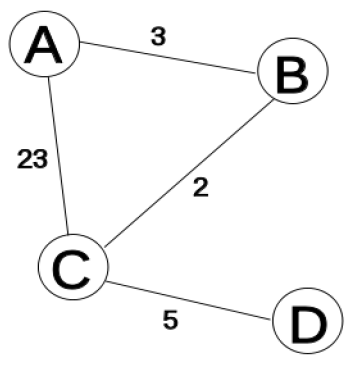
Examples of distance-vector routing protocols include Routing Information Protocol Version 1 & 2, RIPv1 and RIPv2 and IGRP. EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.

**Method**

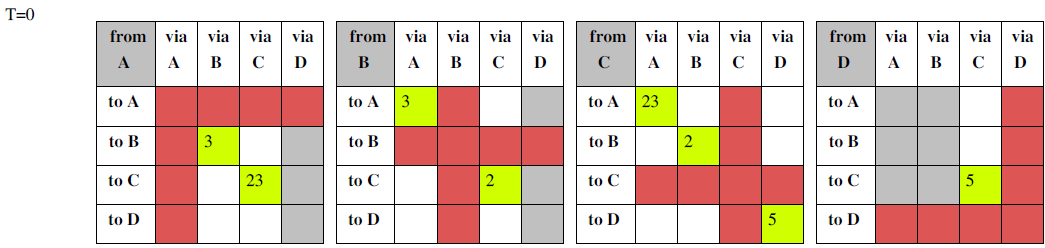
The methods used to calculate the best path for a network are different between different routing protocols but the fundamental features of distance-vector algorithms are the same across all DV based protocols. As the name suggests the DV protocol is based on calculating the direction and distance to any link in a network. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth. Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbours that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbours of the changes. This process has been described as ‘routing by rumour’ because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

**Example**

In this network we have 4 routers A, B, C, and D:

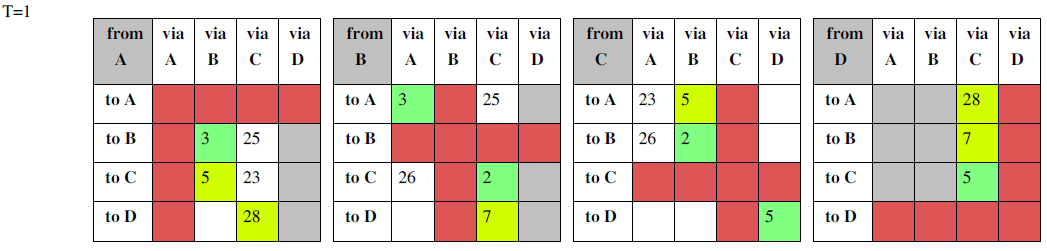


We shall mark the current time (or iteration) in the algorithm with T, and shall begin (at time 0, or T=0) by creating distance matrices for each router to its immediate neighbors. As we build the routing tables below, the shortest path is highlighted with the colorgreen, a new shortest path is highlighted with the color yellow.

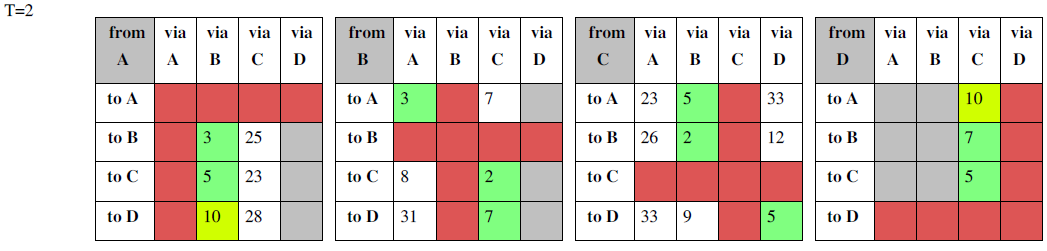


At this point, all the routers (A,B,C,D) have new "shortest-paths" for their DV (the list of distances that are from them to another router via a neighbour). They each broadcast this new DV to all their neighbours: A to B and C, B to C and A, C to A, B, and D, and D to C. As each of these neighbours receives this information, they now recalculate the shortest path using it.

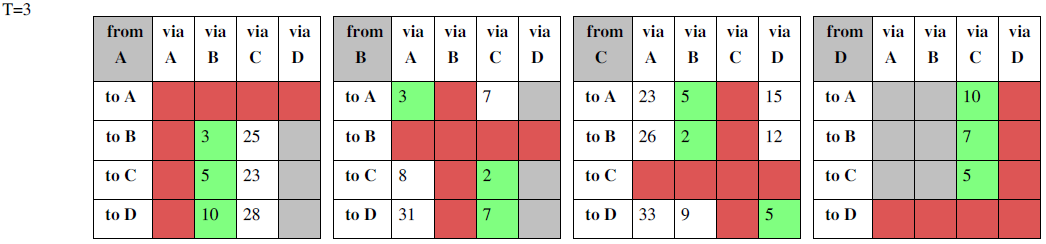
For example: A receives a DV from C that tells A there is a path via C to D, with a distance (or cost) of 5. Since the current "shortest-path" to C is 23, then A knows it has a path to D that costs 23+5=28. As there are no other shorter paths that A knows about, it puts this as its current estimate for the shortest-path from itself (A) to D, via C.



Again, all the routers have gained in the last iteration (at T=1) new "shortest-paths", so they all broadcast their DVs to their neighbours; This prompts each neighbour to re-calculate their shortest distances again. For instance: A receives a DV from B that tells A there is a path via B to D, with a distance (or cost) of 7. Since the current "shortest-path" to B is 3, then A knows it has a path to D that costs 7+3=10. This path to D of length 10 (via B) is shorter than the existing "shortest-path" to D of length 28 (via C), so it becomes the new "shortest-path" to D.



This time, only routers A and D have new shortest-paths for their DVs. So they broadcast their new DVs to their neighbors: A broadcasts to B and C, and D broadcasts to C. This causes each of the neighbors receiving the new DVs to re-calculate their shortest paths. However, since the information from the DVs doesn't yield any shorter paths than they already have in their routing tables, then there are no changes to the routing tables.



None of the routers have any new shortest-paths to broadcast. Therefore, none of the routers receive any new information that might change their routing tables. So the algorithm comes to a stop.

**Limitations**

The Bellman-Ford algorithm does not prevent routing loops from happening and suffers from the count-to-infinity problem. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like as A-B-C-D-E-F, and let the metric between the routers be "number of jumps". Now suppose that A goes down (out of order). In the vector-update-process B notices that the route to A, which was distance 1, is down - B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down - so it tells B that A is only two jumps from C (C to B to A) , which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the "Relax property" of Bellman Ford).

**Partial solutions**

RIP uses Split Horizon with Poison Reverse technique to reduce the chance of forming loops and uses a maximum number of hops to counter the 'count-to-infinity' problem. These measures avoid the formation of routing loops in some, but not all, cases. The addition of a hold time (refusing route updates for a few minutes after a route retraction) avoids loop formation in virtually all cases, but causes a significant increase in convergence times. A number of loop-free distance vector protocols, such as EIGRP and DSDV, have been developed. These avoid loop formation in all cases, but suffer from increased complexity, and their deployment has been slowed down by the success of link-state protocols such as OSPF.

**Algorithm:**

1. Enter number of nodes as a input from user

2. Enter cost matrix

3. Initialize distance equal to cost matrix

4. Choose arbitrary vertex k and calculate the direct distance from the node i to k using the cost matrix and add the distance from k to node j

5. Calculate minimum distance

**Conclusion:** Successfully studied and implemented Distance Vector Routing Algorithm.

**Faqs:**

1**.** What are the drawbacks of Distance Vector Routing Algorithm?

2. State and explain Count-to-infinity problem.

**Mathematical Model:**

1.Initializing routing table2. Sharing distance3. Updating routing table

OUTPUT

PROCESSING

INPUT

* **Input**: Number of nodes, Distance between nodes

* **Process**: 1.Initializing routing table

2. Sharing distance

3. Updating routing table

* **Output**: Minimum distance from one node to every other node.
* **Successive cases**:

i.If we get the shortest distance of each node

ii. Program executed correctly

* **Failure cases:**

i.If link between any two nodes is broken which leads to Count- to-infinity problem