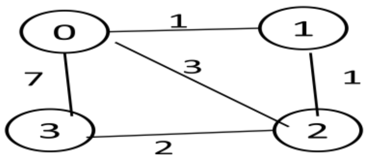
Week #10 – 14th March 2016

Write program to illustrate Distance Vector Routing algorithm

1	**Title of the experiment**	**Simulation of distance vector routing algorithm**
2	**What needs to be done**	Write and execute the program to simulate and **illustrate effectively** the Distance Vector Routing algorithm for the following network topology. 
3	**Input**	• Network topology along with parameters (Costs) to be entered • Assume that initially every node knows only about its neighbors
4	**Expected output**	A. Routing table of each node, at every iteration, to be displayed. B. If the cost of any link changes, routing table of all the tables has to be updated.
3	**Language / Tool / Platform**	Python or java
4	**Reference material**	PPT –File-DV attached

GUIDELINES :

In this programming assignment, you need to write a “distributed” set of procedures that implements a distributed asynchronous distance-vector routing.

You are to write the following routines that will “execute” asynchronously within the emulated environment provided for this assignment.

***rtinitA().* This routine will be called once at the beginning of the emulation.**

Rtinit0() has no arguments. It should initialize your distance table in node 0 to reflect the direct costs of 1, 3, and 7 to nodes 1, 2, and 3, respectively. In the figure above, all links are bidirectional and the costs in both directions are identical.

After initializing the distance table and any other data structures needed by your node 0 routines, it should then send its directly connected neighbors (in this case, 1, 2, and 3) the cost of its minimum-cost paths to all other network nodes. This minimum-cost information is sent to neighboring nodes in a routing update packet by calling the routine *tolayer2(),* as described in the full assignment.
The format of the routing update packet is also described in the full assignment.

• *rtupdate0(struct rtpkt *rcvdpkt)*. This routine will be called when node 0 receives a routing packet that was sent to it by one of its directly connected neighbors. The parameter **rcvdpkt* is a pointer to the packet that was received.

rtupdate0() is the “heart” of the distance-vector algorithm. The values it receives in a routing update packet from some other node *i* contain *i’*s currentshortest-path costs to all other network nodes.

rtupdate0() uses these receivedvalues to update its own distance table (as specified by the distance-vector algorithm). If its own minimum cost to another node changes as a result of the update, node 0 informs its directly connected neighbors of this change in minimum cost by sending them a routing packet. Recall that in the distance-vector algorithm, only directly connected nodes will exchange routing packets. Thus,nodes 1 and 2 will communicate with each other, but nodes 1 and 3 will not
communicate with each other.

Similar routines are defined for nodes 1, 2, and 3. Thus, you will write eight procedures in all: *rtinit0(), rtinit1(), rtinit2(), rtinit3(), rtupdate0(), rtupdate1(), rtupdate2(),*and *rtupdate3().* These routines will together implement a distributed, asynchronous computation of the distance tables for the topology and costs shown in the figure on the preceding page.