
# INDIAN INSTITUTE OF TECHNOLOGY(ISM), DHANBAD.

**Academic Year:** 2021 -22

**Branch:** Electronics and Communication Engineering

**Subject:** Communication Networks (ECC17103)

**Mentor:** Prof. Jaisingh Thangaraj

# Distance Vector Routing Protocol

**Name**: Ajay Jagannath

**Admission No**: 18JE0049

**Email ID**: ajayjagan2511.18je0049@ece.iitism.ac.in

# Table of Contents

# ABSTRACT

Routing is the process of selecting a path for transporting data packets within a network or across multiple networks. Routers work in the following way: when a router receives a packet, it reads the headers of the packet to see its intended destination and determines where to route the packet based on information in its routing tables.

The decisions made on which path to take is made by forming a routing table based on certain protocols that place emphasis on minimizing the cost and time required to transfer the information. There are mainly two protocols, Distance Vector Routing Protocol and Link State Routing Protocol which define the governing principals of various routing Algorithm.

This project aims to shed light on the Distance Vector Routing Protocol, its rules, algorithm and workings on a randomized network to find the minimum cost path between Source and Destination.

# INTRODUCTION

At the simplest level, routing establishes basic internetwork communications, implements an addressing structure that uniquely identifies each device, and organizes individual devices into a hierarchical network structure. Controlling routing flows within the network is vital to the proper operation and performance of the network. Routing protocols use certain metrics to evaluate what path will be the best for a packet to travel. A **metric** is a criterion of measurement; such as path bandwidth, reliability, delay, current load on that path etc. that is used by routing algorithms to determine the optimal path to a destination.

Desirable properties of a router are as follows:

- **Correctness and simplicity:** The packets are to be correctly delivered. Simpler the process, the better.
- **Robustness:** Ability of the network to deliver packets somehow even in the face of failures.
- **Stability:** The algorithm should converge to equilibrium as fast as possible with changing conditions in the network.
- **Efficiency**: Minimum overhead cost

Routing Protocol Design Parameter are as follows:

- **Performance Criteria:** Metric ->Number of hops, Cost, Delay, Throughput, etc.
- **Decision Time:** When ->Per packet basis (Datagram) or per session (Virtual-circuit) basis
- **Decision Place:** Where ->Each node (distributed), Central node (centralized), Originated node (source)
- **Network Information Source:** None, Local, Adjacent node, Nodes along route, All nodes
- **Network Information Update Timing**: Continuous, Periodic, Major load change, Topology change

Distance Vector Routing also known as **Bellman-Ford routing algorithm and Ford-Fulkerson algorithm** is a dynamic routing algorithm where each node maintains a distance table called a **vector**. Distance vector routing protocol is an intradomain routing protocol. Intra domain routing means routing inside the autonomous system (i.e. cluster of routers and networks authorized by a single administrator).

The Distance vector algorithm is iterative, asynchronous and distributed.

**Decision place is Distributed**: It is distributed in that each node receives information from one or more of its directly attached neighbors, performs calculation and then distributes the result back to its neighbors.

**Network Information Update Timing is Iterative**: It is iterative in that its process continues until no more information is available to be exchanged between neighbors.

**Operation is Asynchronous**: It does not require that all of its nodes operate in sync with each other.

**Key Features**

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. 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 been changed.

Each Node maintains a Distance Vector table containing the distance between itself and all possible destination nodes keeping into account the minimum cost for the transaction and the next hop (node) in the path.

# ALGORITHM

- Each router prepares its routing table. By their local knowledge. each router knows about-
    1. All the routers present in the network
    2. Distance to its neighboring routers
    3. All other routers Distance assigned as infinity
- Each router exchanges its distance vector with its neighboring routers.
- Each router prepares a new routing table using the distance vectors it has obtained from its neighbors.
- This step is repeated for (n-2) times if there are n routers in the network.
- The path calculation is based on minimizing the cost to each destination

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

C(x,v) = Cost to each neighbor v of x.

Dx = [Dx(y): y $\in$ N ] = Node x maintains distance vector

Node x also maintains its neighbors' distance vectors

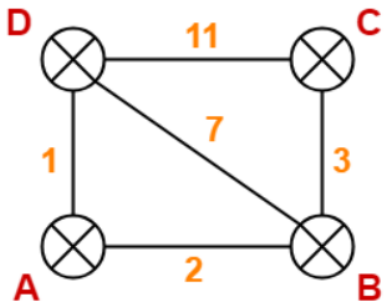
-For each neighbor v, x maintains Dv = [Dv(y): y $\in$ N ]

- From time-to-time, each node sends its own distance vector estimate to neighbors.
- When a node x receives new path estimate from any neighbor v, it saves v's distance vector and it updates its own path using this equation:

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

- Therefore, the path taken from X to Y is the minimum cost path of Y from each neighbor of X + Cost of that Neighbor of X.

# EXAMPLE




Weighted Graph with Four Nodes.

Weight may be any metric like cost or delay.

## Routing Table with Local Knowledge

A

| Destination | Distance | Next Hop |
|---|---|---|
| A | 0 | A |
| B | 2 | B |
| C | ∞ | – |
| D | 1 | D |

B

| Destination | Distance | Next Hop |
|---|---|---|
| A | 2 | A |
| B | 0 | B |
| C | 3 | C |
| D | 7 | D |

C

| Destination | Distance | Next Hop |
|---|---|---|
| A | ∞ | – |
| B | 3 | B |
| C | 0 | C |
| D | 11 | D |

D

| Destination | Distance | Next Hop |
|---|---|---|
| A | 1 | A |
| B | 7 | B |
| C | 11 | C |
| D | 0 | D |

## Final Routing Table

A

| Destination | Distance | Next Hop |
|---|---|---|
| A | 0 | A |
| B | 2 | B |
| C | 5 | B |
| D | 1 | D |

B

| Destination | Distance | Next Hop |
|---|---|---|
| A | 2 | A |
| B | 0 | B |
| C | 3 | C |
| D | 3 | A |

C

| Destination | Distance | Next Hop |
|---|---|---|
| A | 5 | B |
| B | 3 | B |
| C | 0 | C |
| D | 10 | B |

D

| Destination | Distance | Next Hop |
|---|---|---|
| A | 1 | A |
| B | 3 | A |
| C | 10 | B |
| D | 0 | D |

# MATLAB CODE

```
%Distance Vector Routing Protocol

clc;
clear;
close all;
disp('Distance vector routing protocol');
%Get total nodes from the user
node=input('Enter the no. of nodes: ');


%Randomly Generating N nodes forming a weighted Graph
network=zeros(node,node);
for i=1:node    %initializing
    for j=i:node
        if(i==j)
    network(i,j)=0; %diagonal cost is zero
        else
            network(i,j)=rand(1,1); %creates complete matrix
            if(network(i,j)>0.750)%randomly cut off links between nodes
                network(i,j)=0; %no connection between the nodes
            end
            network(j,i)=network(i,j);
        end
    end
end

disp(network);
%0 in matrix implies no connection between nodes

%Assign numbers to the nodes
numbered_nodes=zeros(node,node);
x=1;
 upp_tri=triu(network); %Upper triangle of matrix
 for i=1:node
     for j=1:node
         if(upp_tri(i,j)~=0)
             numbered_nodes(i,j)=x;
             numbered_nodes(j,i)=numbered_nodes(i,j);
             x=x+1;
         end
     end
 end



%Displaying graph
view(biograph(triu(network),[],'showarrows','off','ShowWeights','on','EdgeTextColor',
[0 0 1]));
```

```
% Fill initial matrix with via = to (ie: no connecting nodes in
% between) meaning for all nodes
%routing table based on local Information

routing_table=zeros(node,node,node);
for from=1:node
    for next_hop=1:node
        for to=1:node
            if(from~=next_hop&&from~=to)
                if(next_hop==to&&network(from,to)~=0)
                    routing_table(to,next_hop,from)=network(from,to);
                else
                    routing_table(to,next_hop,from)=100;%depicts non availablity of
route
                end
            else
                    routing_table(to,next_hop,from)=inf;
            end
        end
    end
end

disp('Routing Table with local knowledge');
disp(routing_table);

%Final Routing Table
temp=zeros(node,node);
i=0;
while(i<2)
    for from=1:node
        for to=1:node
            if(from~=to)
                if(network(from,to)~=0)%calculate neighbor node
                    for x=1:node
                        for y=1:node
                            temp(x,y)=network(from,to)+min(routing_table(y,:,to));

if(temp(x,y)<routing_table(y,to,from)&&routing_table(y,to,from)<inf)
                                routing_table(y,to,from)=temp(x,y);
                            end
                        end
                    end
                end
            end
        end
    end
    i=i+1;
end

%For each node: row: Node to Reach ; column: Next Hop
disp('Final Routing Table');
disp(routing_table);
```

```
choice='y';
while(choice=='y')
    source=input('Enter the source node: '); %choosing source node
dest=input('Enter the destination node: ');
%choosing destination node
    path_nodes(1)=source;
    j=2;
    while(source~=dest)
        %minimum cost

[row,col]=find(routing_table(dest,:,source)==min(routing_table(dest,:,source)));
        path_nodes(j)=col;
        source=col;
        j=j+1;
    end

    k=1:j-1;
    disp("Path Travelled");
    disp(path_nodes(k));

bg=biograph(triu(network),[],'showarrows','off','ShowWeights','on','EdgeTextColor',[0
0.5 1]);

    for i=1:j-2
        set(bg.nodes(path_nodes(i)), 'color', [0 0 1]);
   %Path Nodes: BLUE
        set(bg.nodes(path_nodes(1)),'color',[0 1 0]);
        %Source Node: GREEN
        set(bg.nodes(path_nodes(j-1)),'color',[1 0 0]);
        %Destination Node: RED
        if(i<j-1)
         set(bg.edges(numbered_nodes(path_nodes(i+1),path_nodes(i))),'linecolor',[1 0
0]);
    %Connecting Edges: RED
        end
    end

    view(bg);

choice=input('Do you want to try again (y/n):','s');
%Choose different source and destination

end
```

# OUTPUT

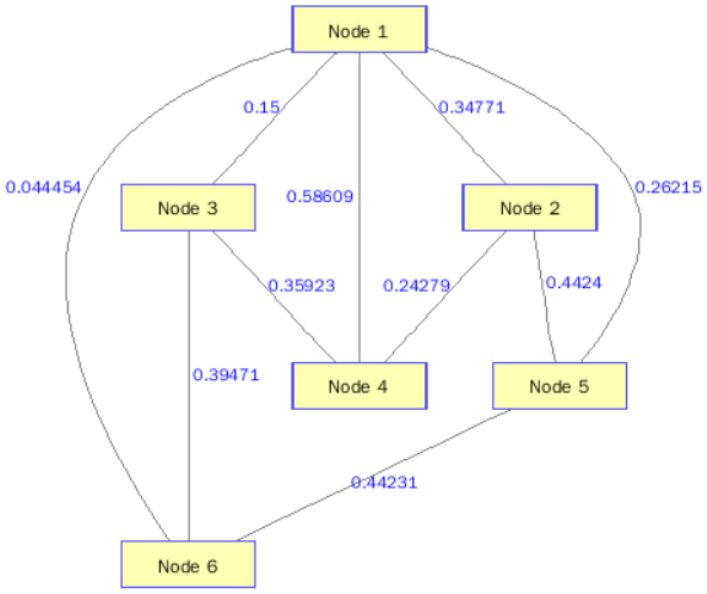
```
Distance vector routing protocol
Enter the no. of nodes:
6
         0    0.3477    0.1500    0.5861    0.2621    0.0445
    0.3477         0         0    0.2428    0.4424         0
    0.1500         0         0    0.3592         0    0.3947
    0.5861    0.2428    0.3592         0         0         0
    0.2621    0.4424         0         0         0    0.4423
    0.0445         0    0.3947         0    0.4423         0

Routing Table with local knowledge

(:,:,1) =
       Inf       Inf       Inf       Inf       Inf       Inf
       Inf    0.3477  100.0000  100.0000  100.0000  100.0000
       Inf  100.0000    0.1500  100.0000  100.0000  100.0000
       Inf  100.0000  100.0000    0.5861  100.0000  100.0000
       Inf  100.0000  100.0000  100.0000    0.2621  100.0000
       Inf  100.0000  100.0000  100.0000  100.0000    0.0445

(:,:,2) =
    0.3477       Inf  100.0000  100.0000  100.0000  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
  100.0000       Inf  100.0000  100.0000  100.0000  100.0000
  100.0000       Inf  100.0000    0.2428  100.0000  100.0000
  100.0000       Inf  100.0000  100.0000    0.4424  100.0000
  100.0000       Inf  100.0000  100.0000  100.0000  100.0000


(:,:,3) =
    0.1500  100.0000       Inf  100.0000  100.0000  100.0000
  100.0000  100.0000       Inf  100.0000  100.0000  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
  100.0000  100.0000       Inf    0.3592  100.0000  100.0000
  100.0000  100.0000       Inf  100.0000  100.0000  100.0000
  100.0000  100.0000       Inf  100.0000  100.0000    0.3947
```

```
(:,:,4) =
    0.5861  100.0000  100.0000       Inf  100.0000  100.0000
  100.0000    0.2428  100.0000       Inf  100.0000  100.0000
  100.0000  100.0000    0.3592       Inf  100.0000  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
  100.0000  100.0000  100.0000       Inf  100.0000  100.0000
  100.0000  100.0000  100.0000       Inf  100.0000  100.0000

(:,:,5) =
    0.2621  100.0000  100.0000  100.0000       Inf  100.0000
  100.0000    0.4424  100.0000  100.0000       Inf  100.0000
  100.0000  100.0000  100.0000  100.0000       Inf  100.0000
  100.0000  100.0000  100.0000  100.0000       Inf  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
  100.0000  100.0000  100.0000  100.0000       Inf    0.4423

(:,:,6) =
    0.0445  100.0000  100.0000  100.0000  100.0000       Inf
  100.0000  100.0000  100.0000  100.0000  100.0000       Inf
  100.0000  100.0000    0.3947  100.0000  100.0000       Inf
  100.0000  100.0000  100.0000  100.0000  100.0000       Inf
  100.0000  100.0000  100.0000  100.0000    0.4423       Inf
       Inf       Inf       Inf       Inf       Inf       Inf

Final Routing Table

(:,:,1) =
       Inf       Inf       Inf       Inf       Inf       Inf
       Inf    0.3477    0.6477    0.8289    0.7045    0.4366
       Inf    0.8454    0.1500    0.9453    0.6743    0.2389
       Inf    0.5905    0.5092    0.5861    0.9473    0.5981
       Inf    0.7901    0.5621    1.2713    0.2621    0.3511
       Inf    0.7399    0.3444    1.1398    0.5687    0.0445

(:,:,2) =
    0.3477       Inf  100.0000    0.7520    0.7045  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
    0.4977       Inf  100.0000    0.6020    0.8545  100.0000
    0.8569       Inf  100.0000    0.2428    1.1276  100.0000
    0.6099       Inf  100.0000    0.9280    0.4424  100.0000
    0.3922       Inf  100.0000    0.7965    0.7490  100.0000
```

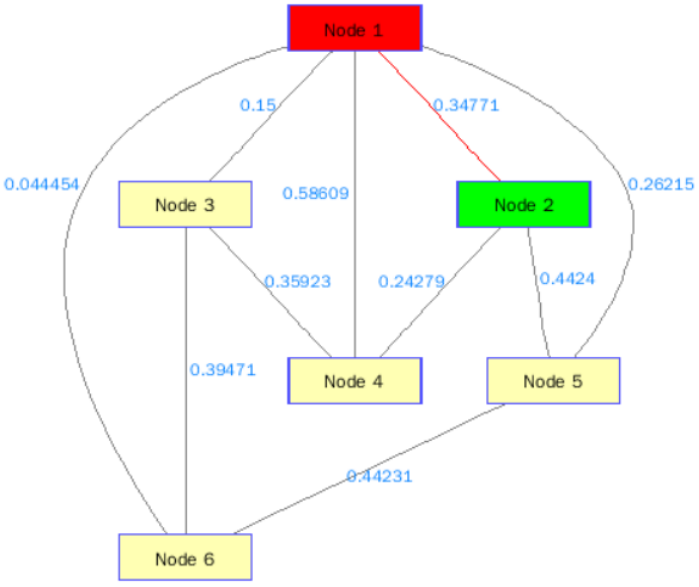
```
(:,:,3) =
    0.1500  100.0000       Inf    0.8685  100.0000    0.4392
    0.4977  100.0000       Inf    0.6020  100.0000    0.7869
       Inf       Inf       Inf       Inf       Inf       Inf
    0.6592  100.0000       Inf    0.3592  100.0000    0.9484
    0.4121  100.0000       Inf    1.0444  100.0000    0.7013
    0.1945  100.0000       Inf    0.9129  100.0000    0.3947

(:,:,4) =
    0.5861    0.5905    0.5092       Inf  100.0000  100.0000
    0.9338    0.2428    0.8569       Inf  100.0000  100.0000
    0.7361    0.7405    0.3592       Inf  100.0000  100.0000
       Inf       Inf       Inf       Inf       Inf       Inf
    0.8482    0.6852    0.7714       Inf  100.0000  100.0000
    0.6305    0.6350    0.5537       Inf  100.0000  100.0000

(:,:,5) =
    0.2621    0.7901  100.0000  100.0000       Inf    0.4868
    0.6099    0.4424  100.0000  100.0000       Inf    0.8345
    0.4121    0.9401  100.0000  100.0000       Inf    0.6368
    0.7714    0.6852  100.0000  100.0000       Inf    0.9960
       Inf       Inf       Inf       Inf       Inf       Inf
    0.3066    0.8346  100.0000  100.0000       Inf    0.4423


(:,:,6) =
    0.0445  100.0000    0.5447  100.0000    0.7045       Inf
    0.3922  100.0000    0.8924  100.0000    0.8847       Inf
    0.1945  100.0000    0.3947  100.0000    0.8544       Inf
    0.5537  100.0000    0.7539  100.0000    1.1275       Inf
    0.3066  100.0000    0.8069  100.0000    0.4423       Inf
       Inf       Inf       Inf       Inf       Inf       Inf
```

```
Enter the source node:
2
Enter the destination node:
1
Path Travelled
     2     1


Do you want to try again (y/n):
y

Enter the source node:
6
Enter the destination node:
5
Path Travelled
     6     1     5


Do you want to try again (y/n):
y

Enter the source node:
6
Enter the destination node:
4
Path Travelled
     6     1     3     4


Do you want to try again (y/n):
n
```

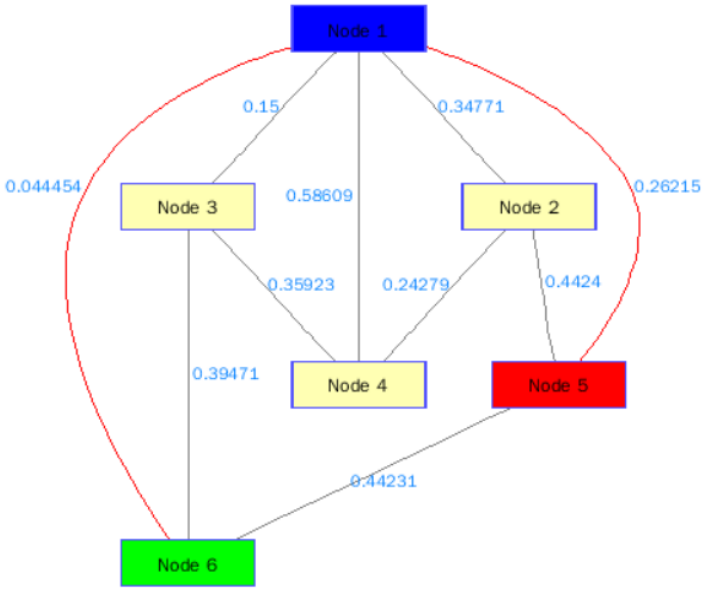
# RESULTS



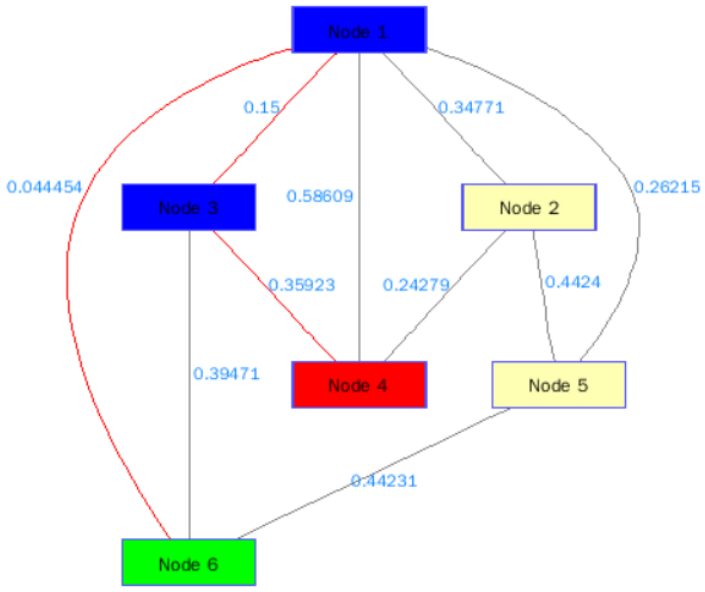

1. Randomized Graph formed for number of nodes=6




2. Next hop=Destination
(Node 2 to Node 1)




3. Destination is neighbor of Next Hop
(Node 6 to Node 5)




4. Multiple Path Nodes between Source and Destination
(Node 6 to Node 4)

MATLAB code takes number of nodes as input from user and creates a randomized weighted graph and displays the connections **-> Graph 1**

Following the Distance Vector Routing Protocol, a routing table is made for each mode consisting of minimum cost to reach every node in the graph and the immediate next hop to do the same (value=100 implies the path does not exist). This Routing Table is Displayed.

User is asked to select source and destination node. Based on the routing table the path is taken is displayed**.**

**Source Node:** Green

**Connecting Nodes:** Blue

**Destination Node:** Red

**Connecting Edges:** Red

# SPECIAL CASES

(A) **Source node = Destination Node**

There are no edges to be traversed when data has to be sent from a router back to itself, so there will **NO CHANGES** in the Graph.

(B) **No Path from Source Node to Destination Node**

If there is no path that exists between source and destination then the routing table will denote it by cost=100 for every possible Next Hop. Hence an exception will throw up stating that **"No Routes exist"**.

# INFERENCE

Distance vector routing protocol is a simple routing technique and is easy to implement in small networks. It has very little redundancy and lower bandwidth due to much of the information is being passed on from the neighbors than being directly calculated.

However due to whole routing tables being shared from one router to another, this protocol is susceptible to security threats, since the whole topology can be mapped from a single node. Also, complete updates occur from time to time, which is not necessary unless the Network is changed drastically.

To overcome these issues, shorted path first protocol such as link state routing protocol or the combination of the two called Advanced Distance vector routing protocol can be explored.

Hence, Distance Vector Routing Protocol is best suited for Simple and small networks that are frequently changed, to take the best advantage of all its features.

# REFERENCES


1) https://nptel.ac.in/content/storage2/courses/106105080/pdf/M7L1.pdf

2) https://www.tutorialspoint.com/what-is-distance-vector-routing-algorithm

3) https://www.javatpoint.com/distance-vector-routing-algorithm

4) https://www.mathworks.com/help/matlab/

5) https://www.geeksforgeeks.org/classes-of-routing-protocols/?ref=lbp

6) https://www.youtube.com/watch?v=dmS1t2twFrI&ab_channel=CSETechnicalVideos

7) https://binaryterms.com/distance-vector-routing-protocol.html