|  |  |
| --- | --- |
| **Ex. No: 7** | **Implementation of Distance Vector Routing** |
| **04.03.25** |

**Pre-Lab Exercise for Distance Vector Routing**

1. What is the primary function of a routing protocol in a computer network?
2. Explain the difference between static and dynamic routing protocols.
3. What is the Distance Vector Routing (DVR) protocol, and how does it work?
4. What are the potential limitations or disadvantages of using Distance Vector Routing?

**In Lab Exercise for Distance Vector Routing**

**Aim:**

To understand and implement **Distance Vector Routing (DVR)**, a dynamic routing algorithm that determines the best path for packet forwarding using the **Bellman-Ford Algorithm**. The goal is to observe how routers update their routing tables and exchange distance vector information to maintain efficient network connectivity.

**Algorithm:**

**Step 1: Input the Network Topology**

1. Start the program.
2. Accept the number of nodes (**n**) from the user.
3. Input the **cost matrix** (distance between nodes).
4. Set **diagonal elements to 0** (distance to itself is always 0).

**Step 2: Initialize the Routing Tables**

1. For each router *i*, initialize:
   * Distance to **all destinations** based on the cost matrix.
   * Next-hop for each destination as **directly connected nodes**.

**Step 3: Apply the Distance Vector Algorithm**

1. **Repeat Until No More Updates:**
   * Set **count = 0** (indicates no change).
   * For each router *i*:
     + For each destination *j*:
       - Check all possible intermediate routers *k*:
         * If the path through *k* offers a **shorter route**, update:

**Distance**: rt[i].dist[j] = dmat[i][k] + rt[k].dist[j]

**Next-hop**: rt[i].from[j] = k

**Increment count** (indicating a change).

* + Repeat the process until **count = 0** (i.e., no further updates).

**Step 4: Display the Final Routing Tables**

1. Print the final **routing table for each router**, showing: Destination node, Shortest distance and Next-hop router

**Step 5: End the Program**

**Program:**

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int dmat[20][20];

int n,i,j,k,count=0;

printf("\nEnter the number of nodes : ");

scanf("%d",&n);

printf("\nEnter the cost matrix :\n");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

{

scanf("%d",&dmat[i][j]);

dmat[i][i]=0;

rt[i].dist[j]=dmat[i][j];

rt[i].from[j]=j;

}

do

{

count=0;

for(i=0;i<n;i++)

for(j=0;j<n;j++)

for(k=0;k<n;k++)

if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])

{

rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];

rt[i].from[j]=k;

count++;

}

}while(count!=0);

for(i=0;i<n;i++)

{

printf("\n\nFinal Routing Table for Router %d is \n",i+1);

printf("Destination \t Distance \t Next Hop\n");

printf("-------------------------------------------\n");

for(j=0;j<n;j++)

{

printf(" %d\t\t %d\t\t %d\n",j+1,rt[i].dist[j],rt[i].from[j]+1);

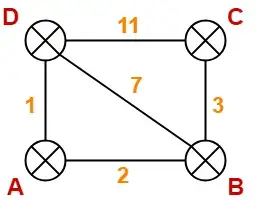
}

}

printf("\n\n");

}

**Output:**

****

A screenshot of a computer

AI-generated content may be incorrect.

**Result**

The implementation of Distance Vector Routing (DVR) using the Bellman-Ford Algorithm demonstrated how routers dynamically update routing tables by exchanging distance vector information, ensuring efficient packet forwarding.

**Post-Lab Exercise for Distance Vector Routing**

1. How did routers update their routing tables during the experiment?
2. What challenges were observed when routers attempted to find the best path to destinations?
3. How did changes in network topology (e.g., a link failure) impact the routing tables during the lab?
4. What would happen if a router receives an incorrect distance vector from a neighboring router? How would it affect the overall network performance?

**Assessment:**

|  |  |  |
| --- | --- | --- |
| **Description** | **Max. Marks** | **Marks Awarded** |
| **Pre Lab Exercise** | 5 |  |
| **In Lab Exercise** | 20 |  |
| **Post Lab Exercise** | 5 |  |
| **Viva** | 10 |  |
| **Total** | 40 |  |
| **Faculty Signature with Date** | |  |