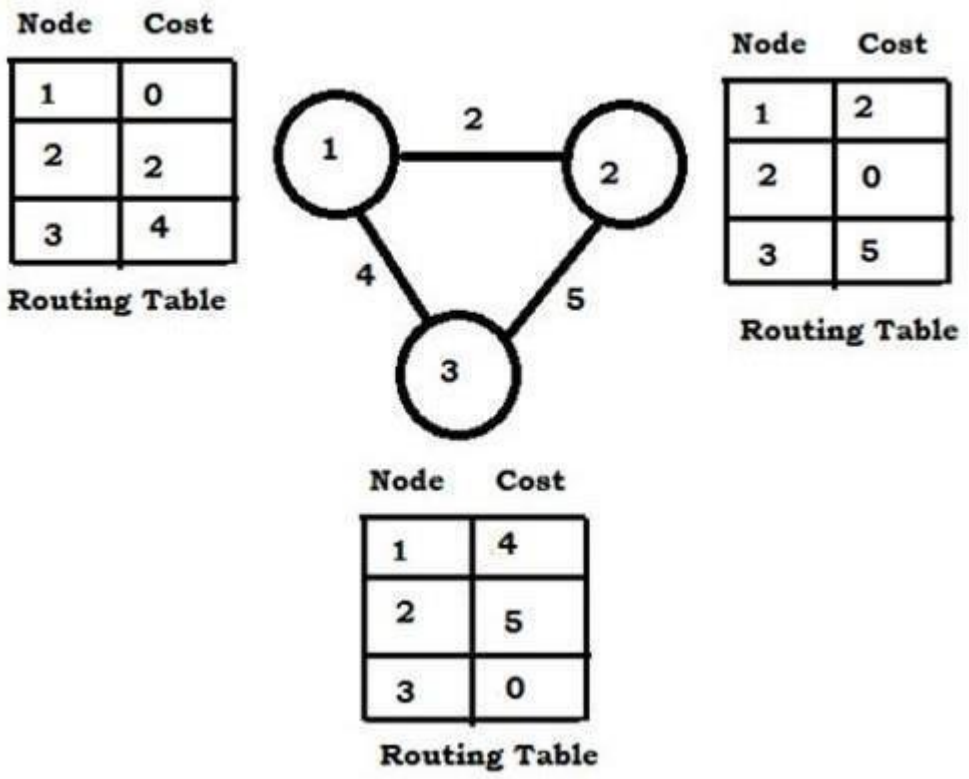
EXPERIMENT NO: 5

Implement distance vector routing algorithm for obtaining routing tables at each node.

AIM: Obtain Routing table at each node using distance vector routing algorithm for a given subnet.



HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination. Distance vector routing algorithms are not preferable for complex networks and take longer to converge

ALGORITHM:

Step-by-Step Distance Vector Algorithm

Step 1: Initialization

1. **Set Up Routing Table**: Each router initializes its routing table. For each destination:
    - The distance to itself is set to 0.
    - The distance to directly connected neighbors is set to 1 (or the appropriate cost).
    - The distance to all other destinations is set to infinity (∞).

Step 2: Exchange Distance Vectors

2. **Periodic Updates**: Each router periodically sends its entire distance vector (routing table) to its immediate neighbors.

Step 2: Update Routing Tables

3. **Receive Distance Vectors**: When a router receives a distance vector from a neighbor:
    - For each destination in the neighbor's vector, calculate the cost to reach that destination via the neighbor.
    - Use the formula: New Cost(D)=Cost(Neighbor)+Cost(D from Neighbo
    - Update the routing table if the new calculated cost is less than the current cost:
        - . If New Cost(D)<Current Cost(D)
        - Update Current Cost(D)to New Cost(D)
        - Update the next hop for DDD to this neighbor.

Step 3: Repeat Process

4. **Iterate**: Steps 2 and 3 are repeated periodically or until there are no further updates:
    - Each router continues to share its updated routing table with its neighbors.
    - Each router keeps updating its table based on received vectors.

Step 4: Convergence

5. **Convergence**: The algorithm converges when all routers have consistent distance vectors, meaning that no router can find a shorter path than what is already in its table

Example

Let’s consider a simple example with three routers: A, B, and C.

1. **Initialization**:
    - Router A: {A: 0, B: 1, C: ∞}
    - Router B: {A: 1, B: 0, C: 1}
    - Router C: {A: ∞, B: 1, C: 0}
2. **Exchange Distance Vectors**:
    - A sends its table to B and C.
    - B sends its table to A and C.
    - C sends its table to A and B.
3. **Update Tables**:
    - For example, Router A receives from B:
        - New Cost(C) via B = Cost(B) + Cost(C from B) = 1 + 1 = 2.
        - Update C's entry in A’s table: A: 0, B: 1, C: 2.
    - This process continues for all routers.
4. **Iterate**: Each router continues to share and update until no further changes occur.
5. **Convergence**: Once all routers have stable routing tables, the process stops.

Conclusion

The Distance Vector Algorithm is a fundamental routing technique that emphasizes simplicity and ease of implementation. Its effectiveness is often seen in smaller networks, but it has limitations, particularly in larger, more dynamic environments. If you have any specific scenarios or questions, feel free to ask!

4o mini

SOURCE CODE

OUTPUT:

Enter the number of nodes : 2 Enter the cost matrix

: 1 2

1 2

State value for router 1 is node 1 via 1 Distance0

node 2 via 2 Distance2 State value for router 2 is node 1 via 1 Distance1

node 2 via 2 Distance0