ASSIGNMENT - 05

GROUP - 25 :
220123052 - Hardhika Ramineni
220101008 - Adloori Chandana
220101010 - Alampally Khushi
220101009 - Aetukuri Sri Durga

Drive link: CN_ASSIGNMENT05
REPORT Link : CN_ASSIGNMENT05

PART-1

DVR Algorithm Procedure

1. **Initialization**:
Each node initializes its routing table. The cost to itself is 0, the cost to direct neighbors is the edge weight, and the cost to all other nodes is set to infinity (∞), indicating an initially unknown path.

2. **Distance Vector Exchange**:

   ○ Each node periodically sends its distance vector (its routing table) to its immediate neighbors.

   ○ Upon receiving distance vectors from neighbors, each node updates its routing table based on the Bellman-Ford equation: Distance(D)=min(Distance(D),Cost(A→B)+Distance(B→D)) where D is the destination, A is the current node, and B is the neighbor node from whom the update was received.

3. **Convergence**:

   ○ This exchange continues iteratively, with each node updating its routing table upon receiving new information.

   ○ Eventually, each node's routing table stabilizes (converges), meaning no further updates occur, and all nodes have computed the shortest paths to every other node.

4. **Final Output (Routing Tables)**:

   ○ After convergence, each node has a **routing table** showing the shortest path (cost and next hop) to all other nodes.

   ○ The DVR algorithm outputs each node's routing table as a summary of the shortest paths in the network.

Count-to-Infinity Problem

1. **Definition**:

   ○ The count-to-infinity problem occurs in distance vector routing algorithms when nodes continually update their routing tables based on outdated or incorrect information, leading to increasing hop counts to unreachable destinations.

2. **Cause**:

   ○ Arises when a network topology change (e.g., link failure) occurs, causing nodes to misinterpret the state of the network and rely on stale routing information.

3. **Behavior**:

   ○ Nodes may keep incrementing their distance values to a particular destination, failing to recognize that the destination is unreachable due to a broken link.

4. **Threshold**:

   ○ A maximum distance value (often set at 100) is used to indicate unreachable routes. If a node's distance exceeds this threshold, it is flagged as exhibiting count-to-infinity behavior.

Detection in Simulation

1. **Simulation Setup**:

   ○ Removed a critical link (e.g., between nodes 5 and 4) to simulate a network failure.

2. **Re-running the Algorithm**:

   ○ The Distance Vector Routing (DVR) algorithm was re-executed to observe the changes in routing tables after the link removal.

3. **Monitoring Changes**:

   ○ Observed and recorded the distance values in each node's routing table for signs of increasing hop counts.

4. **Flagging Count-to-Infinity**:

   ○ If any node's distance to a destination exceeded 100, it was flagged as experiencing the count-to-infinity problem.

5. **Results**:

   ○ Detected nodes with continually increasing distances, illustrating the limitations of the basic DVR algorithm in adapting to network changes.

```
PS C:\Users\sridu\OneDrive\Desktop\cn> cd "c:\Users\sridu\OneDrive\Desktop\cn\" ; if ($?
q1 } ; if ($?) { .\wfq1 }
5 6
1 2 3
1 3 5
2 3 2
2 4 4
3 4 1
4 5 6

Routing tables after running DVR algorithm:
Routing table for Node 1:
Destination     Cost    Next Hop
1               0       1
2               3       2
3               5       2
4               6       2
5               12      2

Routing table for Node 2:
Destination     Cost    Next Hop
1               3       1
2               0       2
3               2       3
4               3       3
5               9       3
```

```
Routing table for Node 3:
Destination     Cost    Next Hop
1               5       1
2               2       2
3               0       3
4               1       4
5               7       4

Routing table for Node 4:
Destination     Cost    Next Hop
1               6       2
2               3       2
3               1       3
4               0       4
5               6       5

Routing table for Node 5:
Destination     Cost    Next Hop
1               12      4
2               9       4
3               7       4
4               6       4
5               0       5
```

```
Simulate Link Failure between
Node A:



4
Node B: 5
Node 1 has distance >100 to Node 5.
Count-to-infinity problem detected.

Routing tables after link failure:
Routing table for Node 1:
Destination     Cost    Next Hop
1               0       1
2               3       2
3               5       2
4               6       2
5               102     2

Routing table for Node 2:
Destination     Cost    Next Hop
1               3       1
2               0       2
3               2       3
4               3       3
5               99      3
```

```
Routing table for Node 3:
Destination     Cost    Next Hop
1               5       1
2               2       2
3               0       3
4               1       4
5               99      4

Routing table for Node 4:
Destination     Cost    Next Hop
1               6       2
2               3       2
3               1       3
4               0       4
5               98      3

Routing table for Node 5:
Destination     Cost    Next Hop
1               12      4
2               9       4
3               7       4
4               INF     -
5               0       5
```

PART - 2 :

Poisoned Reverse is an enhancement to the Distance Vector Routing (DVR) protocol that helps to prevent the **count-to-infinity** problem. In DVR, routers share their distance vectors (routing information) with each other, allowing each router to know the shortest path to every other router. However, when a link breaks or becomes unavailable, routers can encounter the count-to-infinity problem.

Count-to-Infinity Problem

The count-to-infinity issue occurs when a router mistakenly assumes it can reach a destination through a neighbor, which leads to a loop where each router continually increments the "hop count" to the destination. This results in routes that continue to "count up" indefinitely until they reach a predefined limit, creating inefficiency and incorrect routing.

Poisoned Reverse Mechanism

The **Poisoned Reverse** mechanism prevents a router from advertising a shorter path to a destination back to the router it learned that route from. When using poisoned reverse, instead of not sending any information, the router advertises an artificially large cost (often labeled as "infinity") for the route back to the original sender. This effectively prevents the originating router from choosing this route, as the artificially high cost makes it undesirable.

For example, if router 5 relies on router 4 to reach router 1, and router 4's link to router 5 breaks, router 4 will inform router 5 that the path to router 1 has an "infinite" cost. This way, router 5 will not attempt to route through router 4, avoiding the count-to-infinity cycle.

Steps to Implement Poisoned Reverse

1. When a router sends its routing table to a neighboring router, it will mark the cost as "infinity" for any destination it originally learned from that neighbor.

2. This ensures that the neighbor router won't mistakenly believe it can use this route to reach the destination, avoiding the count-to-infinity loop.

Significance of Poisoned Reverse

Prevents Routing Loops: By signaling "infinity" on routes back to the source, routers avoid creating loops.

Reduces Convergence Time: The network stabilizes faster after a topology change, like a link failure, as routers quickly avoid invalid paths.

Efficient Network Performance: Minimizing routing loops and infinite paths improves network efficiency, keeping routing tables more accurate.

After implementing Poisoned Reverse and re-running the simulation, observe that the routing tables no longer show misleading path costs after a link failure (like the broken link between nodes 4 and 5). This approach helps ensure that routers quickly identify invalid routes, achieving a faster, more reliable convergence of routing information.

Implementing Poisoned Reverse in This Problem

1. **Link Failure Detection**:

   When the link between routers 4 and 5 fails, router 4 identifies the loss of its direct route to router 5, and vice versa.

2. **Propagation of "Poisoned" Routes**:

   Router 4 applies Poisoned Reverse by advertising its route to router 5 with an infinitely high cost to its neighbors (routers 3 and 2), "poisoning" the route from router 4 to router 5.

   This prevents neighboring routers, like router 3, from mistakenly believing they can reach router 5 through router 4.

3. **Updates in Neighboring Routers**:

   As routers exchange updates, routers 1, 2, and 3 adjust their tables to reflect high costs for reaching router 5 via router 4, preventing looped paths through router 4.

   Router 4 maintains this high cost to router 5 until all routers converge on this view of the network.

4. **Convergence**:

   After several updates, each router stabilizes its routing table, effectively treating router 5 as unreachable from router 4. This ensures that routing loops and count-to-infinity issues are avoided by preventing routers from relying on invalid paths.

PART - 3 :

Split Horizon Mechanism is a routing optimization technique used in **distance-vector routing** protocols to prevent routing loops and improve convergence. It works by prohibiting a router from advertising routes back to the neighbor from which it learned them. This practice helps eliminate potential routing loops and allows for faster convergence by stabilizing routing paths more quickly after topology changes. Split Horizon also reduces the amount of routing traffic in the network. Split Horizon plays a crucial role in maintaining a stable and efficient routing environment.

Split Horizon Mechanism :

The **Split Horizon** mechanism improves routing stability in distance vector routing protocols by preventing a router from advertising a route back in the direction it originally learned it from. Unlike Poisoned Reverse, which advertises an artificially high cost, Split Horizon simply withholds updates about a route in specific directions to prevent the spread of potentially misleading information. This approach helps prevent routing loops and the count-to-infinity issue.

For example, suppose router 5 learns about a route to router 1 through router 4. With Split Horizon enabled, router 5 will not advertise its route to router 1 back to router 4. This prevents router 4 from incorrectly considering router 5 as a path to router 1, avoiding a possible loop.

Steps to Implement Split Horizon

1. **Restrict Route Advertisements**: When a router advertises its routing table to a neighboring router, it withholds any route information that it originally learned from that neighbor.

2. **Prevent Loop Creation**: This restriction ensures that routers do not accidentally consider the advertising router as a path to the destination, thus preventing loops.

3. **Repeat for All Neighbors**: The rule is applied across all routers and their neighbors in the network, maintaining stability in dynamic network conditions.

Significance of Split Horizon

- **Prevents Routing Loops**: By withholding routes in the direction they were learned from, routers avoid loops that could arise from cycling path dependencies.

- **Improves Network Convergence**: Split Horizon limits the spread of outdated routing information, reducing the time it takes for the network to converge to accurate route states after changes.

- **Enhances Network Efficiency**: By reducing the risk of count-to-infinity problems and limiting unnecessary route updates, Split Horizon helps keep routing tables precise and improves overall network efficiency.

After implementing Split Horizon, routers quickly update their tables without relying on paths that could otherwise lead to loops. This reduces the need for correction updates after a topology change, such as a link failure, making network convergence faster and more reliable.

Implementing Poisoned Reverse in This Problem:

1. Link Failure Detection:
When the link between routers 4 and 5 fails, both routers recognize the direct route to each other as broken.

2. Route Advertisement Control with Split Horizon:
Using Split Horizon, router 4 stops advertising routes to router 5 back to its neighbors. This prevents router 3 (and subsequently other routers) from learning a route to router 5 through router 4, effectively blocking any potential loop.This means router 4 no longer sends updates indicating a route to router 5 to its neighboring routers, which ensures no misinformation propagates back into the network.

3. Propagation of Updated Routing Information:
As routers exchange updates, router 3 learns that router 5 is no longer reachable through router 4. Each neighboring router updates its own routing tables, marking router 5 as unreachable. Routers continue to exchange information, confirming that router 5 cannot be reached through router 4, which reinforces this new view across the network.

4. Convergence:
Through successive updates, each router stabilizes its routing table. Routers 1, 2, 3, and 4 now all mark router 5 as unreachable, with router 4 effectively removing any invalid paths to router 5.
This controlled propagation ensures that all routers converge on a stable network view without relying on "poisoned" routes, and it effectively prevents count-to-infinity issues by blocking potentially invalid paths from circulating.

Advantages of Split Horizon:
Split Horizon enhances Distance Vector Routing (DVR) by preventing routing loops, as it stops routers from advertising routes back to the neighbors they were learned from. This reduces the risk of incorrect routing information circulating, speeding up network convergence after changes like link failures. It also minimizes unnecessary updates, improving bandwidth efficiency and overall network stability. Its simple implementation can be combined with techniques like Poisoned Reverse to further reinforce accurate route advertisement.