Assignment-5

Distance Vector Routing Algorithm with Edge Break and its Solution

Instructions:

- This is a group assignment.
- The programming questions can be written in C/C++/Java.
- Your code should have a readme file, a makefile, and it should be well commented. These will carry separate marks for each question.
- Submit a soft copy of the report, preferably in PDF format, together with your code in a zip file on teams. The name of the zip file should be like "Your_Groupno.zip" (example: "Group11.zip").
- Files submitted without proper naming format will not be evaluated.
- If your trace file size is larger than 2 MB, you are advised to provide the OneDrive/Google Drive/Dropbox link of the traces in your report.
- Assignments submitted before the deadline will only be considered for evaluation.
- No extensions in submission are allowed. Delay in submission will lead to penalty in marks.
- Please do not email your assignments separately to the TAs, it will not be considered for evaluation.
- Your code will be checked for plagiarism. Any kind of academic dishonesty, plagiarism, etc. will lead to penalties.
- No sharing of code between students, submission of downloaded code is allowed.

Introduction

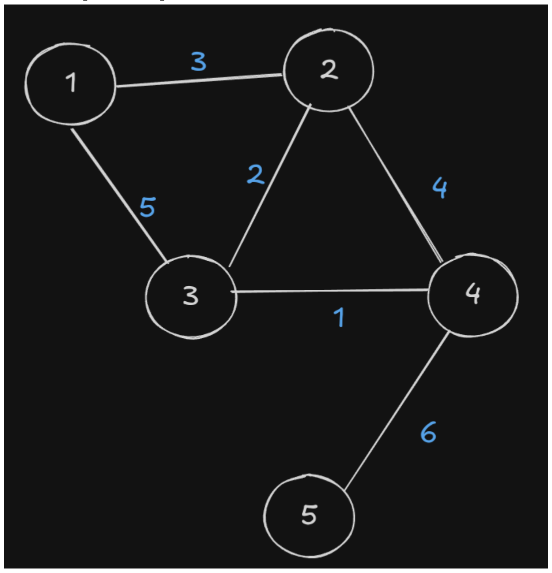
Imagine you are working as a network engineer for a company that has designed a set of routers connected by network links. Each router exchanges information with its neighbors to calculate the shortest paths to every other router in the network. One day, an important network link fails, and your task is to observe how this failure affects the routing tables. You will simulate this scenario and then implement a strategy to prevent routing loops and incorrect information propagation using the **Split Horizon** technique.

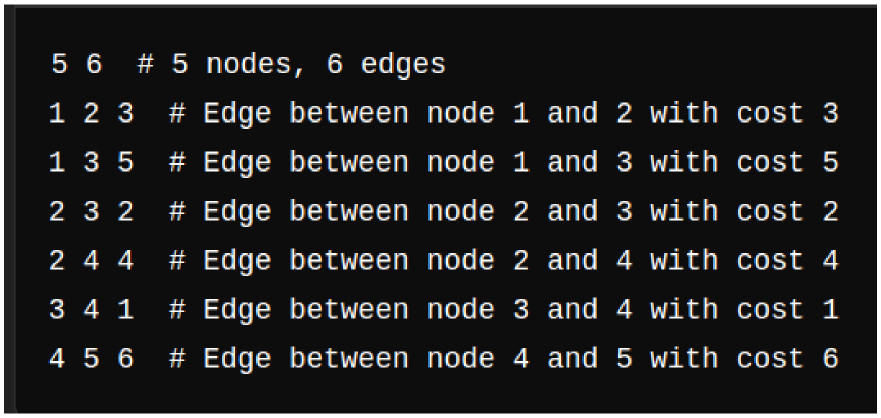
Part 1: Implementing the Distance Vector Routing Algorithm

1. **Graph Setup:**
    - You are provided with a graph of `N` nodes (routers) and `M` edges (network links). Each edge has a weight or cost associated with it. Your task is to implement a DVR algorithm to find the shortest path from each node to every other node.

- Input: A list of edges with their associated costs, formatted as `source, destination, cost`.
- Output: The routing table for each node, showing the shortest path to all other nodes.

Example Input:





```
5 6  # 5 nodes, 6 edges
1 2 3  # Edge between node 1 and 2 with cost 3
1 3 5  # Edge between node 1 and 3 with cost 5
2 3 2  # Edge between node 2 and 3 with cost 2
2 4 4  # Edge between node 2 and 4 with cost 4
3 4 1  # Edge between node 3 and 4 with cost 1
4 5 6  # Edge between node 4 and 5 with cost 6
```

2. **Task:**
    - Implement the DVR algorithm for each node.
    - Print the routing table for each node after running the algorithm.
3. **Simulating Link Failure:**
    - Now, simulate the failure of one of the network links by removing an edge from the graph (e.g., edge between nodes 5 and 4). Re-run the DVR algorithm and observe how the routing tables change. Specifically, you need to check if any nodes enter the "count-to-infinity" problem where their routing tables keep incrementing the hop count. You can terminate if the distance for any node exceeds 100.
4. **Task:**
    - Simulate the removal of an edge.
    - Re-run the DVR algorithm.

    - Observe and print any nodes that exhibit the count-to-infinity problem.

Part 2: Implementing the Poisoned Reverse Mechanism

Poisoned Reverse is a technique that instead of just preventing a router from sending routing information back to the source, the router sends back the information but with an "infinity" value for the distance. This further ensures that the source will never consider using that route.

Task:

- Implement the Poisoned Reverse technique in your DVR algorithm.
- Re-run the simulation where the edge between nodes 5 and 4 is broken.
- Observe how Poisoned Reverse affects the routing tables and whether it successfully prevents the count-to-infinity problem.
- Print the updated routing tables after applying Poisoned Reverse.

Part 3: Implementing the Split Horizon Mechanism

The **Split Horizon** technique is used to prevent a router from sending information about a route back in the direction from which it came, helping to avoid routing loops and the count-to-infinity problem.

Task:

- Implement the Split Horizon technique in your DVR algorithm.
- Re-run the simulation where the edge between nodes 5 and 4 is broken.
- Observe if the Split Horizon technique successfully prevents the count-to-infinity problem.
- Print the updated routing tables for all nodes after applying the Split Horizon technique.

Deliverables

1. **Code Implementation**:
    - A working implementation of the Distance Vector Routing algorithm.
    - Simulation of link failure and observation of the count-to-infinity problem.
    - A 100-distance limit to prevent routing loops.
    - Implementation of the Split Horizon technique to address routing issues.
2. **Results**:
    - Output the initial routing tables after running the DVR algorithm.
    - Output the updated routing tables after the link failure.

    - Output the routing tables after implementing the Poisoned Reverse.
    - Output the routing tables after applying the Split Horizon technique.

Submission Instructions

- Submit your code as a `.zip` file containing all necessary files for running the simulation.
- Include a `README` file that explains how to run the program.

<u>Grading Criteria</u>

Criteria	**Points**
Correct implementation of DVR	25
Handling of link failure and count-to-infinity problem	15
Correct implementation of Poisoned Reverse	20
Correct implementation of Split Horizon	20
Viva	20

Good luck, and may your network always find the shortest path!