**Assignment 11**

**Name: Hitesh Tolani**

**Roll no: 73**

**Class: SY-AIDS-A**

**Title: Write a program to implement a. Network Routing: Shortest path routing, AODV**

**Theory:**

**Shortest path routing**

Dijkstra's algorithm is a popular method for finding the shortest path between nodes in a graph with non-negative edge weights. It starts from a source node and iteratively explores the neighbouring nodes, updating their tentative distances from the source. At each step, it selects the node with the smallest tentative distance as the next one to explore, ensuring that the path to that node is currently the shortest known. This process continues until all reachable nodes have been visited, guaranteeing the shortest path from the source to every other node in the graph.

**AODV**

Ad-hoc On-demand Distance Vector (AODV) routing protocol is designed for wireless ad-hoc networks where nodes may dynamically join or leave the network. AODV establishes routes between nodes only when necessary, in response to specific data packets needing delivery. It employs a distributed algorithm where each node maintains routing tables containing information about available routes to other nodes. When a node needs to send a packet to a destination for which it has no route information, it initiates a route discovery process. During route discovery, control packets are broadcasted through the network, and nodes update their routing tables to establish a route. AODV is efficient for dynamic and mobile networks, as it adapts to changes in network topology without requiring periodic updates, conserving bandwidth and resources.

**Code**

class NetworkRouting:

    def \_\_init\_\_(self, graph):

        self.graph = graph

    def shortest\_path(self, source, destination):

        visited = set()

        distances = {node: float('inf') for node in self.graph}

        distances[source] = 0

        predecessors = {}

        while len(visited) < len(self.graph):

            current\_node = None

            min\_distance = float('inf')

            for node in self.graph:

                if distances[node] < min\_distance and node not in visited:

                    min\_distance = distances[node]

                    current\_node = node

            visited.add(current\_node)

            for neighbor, weight in self.graph[current\_node].items():

                if distances[current\_node] + weight < distances[neighbor]:

                    distances[neighbor] = distances[current\_node] + weight

                    predecessors[neighbor] = current\_node

        path = []

        current\_node = destination

        while current\_node != source:

            path.insert(0, current\_node)

            current\_node = predecessors[current\_node]

        path.insert(0, source)

        return path

class AODV(NetworkRouting):

    def \_\_init\_\_(self, graph):

        super().\_\_init\_\_(graph)

        self.route\_table = {}  # route table for AODV

    def discover\_route(self, source, destination):

        # AODV Route Discovery

        if source not in self.route\_table:

            self.route\_table[source] = {}

        if destination not in self.route\_table[source]:

            shortest\_path = self.shortest\_path(source, destination)

            if shortest\_path:

                self.route\_table[source][destination] = shortest\_path

            else:

                return None

        return self.route\_table[source][destination]

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

    # Creating a sample network graph

    graph = {

        "A": {"B": 1, "C": 2},

        "B": {"A": 1, "C": 1, "D": 3},

        "C": {"A": 2, "B": 1, "D": 1, "E": 4},

        "D": {"B": 3, "C": 1, "E": 1},

        "E": {"C": 4, "D": 1}

    }

    # Initialize routing protocols

    shortest\_path\_routing = NetworkRouting(graph)

    aodv\_routing = AODV(graph)

    # Example usage of shortest path routing

    print("Shortest path from A to E:", shortest\_path\_routing.shortest\_path("A", "E"))

    # Example usage of AODV routing

    print("Route discovered by AODV from A to E:", aodv\_routing.discover\_route("A", "E"))

**Output:**

****

**Conclusion:**

In this assignment, we have learned about network routing and its algorithms such as shortest path and AODV. We have implemented a python program for the same.