:-COMPUTER NETWORKS LAB 10-:

~Pranay K Karvi 23BDS1137

Q] A company operates a multi-area OSPF network with three areas: HQ (Area 0 - backbone), Data Center (Area 1), and Branch Office (Area 2). Each area has multiple routers that dynamically exchange link-state information. Implement OSPF in C programming to handle dynamic topology updates, ensuring routers can join, leave, or fail while maintaining accurate routing tables. Use Dijkstra's Algorithm to compute shortest paths in real time and facilitate inter-area routing by ensuring communication between Area 1 and Area 2 through Area 0. Implement the Hello Protocol and Link-State Advertisements (LSAs) to enable neighbor discovery and periodic route updates. Simulate packet serialization to mimic OSPF packet exchanges between routers. Additionally, design failure detection mechanisms to recalculate paths dynamically upon router or link failures. As a bonus challenge, optimize routing to prioritize QoS-sensitive traffic while maintaining efficient link utilization.

CODE

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <stdbool.h>

#define MAX_ROUTERS 10
#define INF INT_MAX

typedef struct {
   int id;
   bool active;
   int neighbors[MAX_ROUTERS];
   int costs[MAX_ROUTERS];
} Router;

Router network[MAX_ROUTERS];
int num_routers;
```

```
void initialize_network(int n) {
  num_routers = n;
  for (int i = 0; i < n; i++) {
    network[i].id = i;
    network[i].active = true;
    for (int j = 0; j < n; j++) {
      network[i].neighbors[j] = (i == j) ? 0 : INF;
      network[i].costs[j] = (i == j) ? 0 : INF;
    }
  }
}

void add_link(int src, int dest, int cost) {
  network[src].neighbors[dest] = cost;
  network[dest].neighbors[src] = cost;
  network[src].costs[dest] = cost;
  network[dest].costs[src] = cost;
}

void remove_router(int id) {
  network[id].active = false;
  for (int i = 0; i < num_routers; i++) {
    network[i].neighbors[id] = INF;
    network[i].costs[id] = INF;
  }
}

int min_distance(int dist[], bool visited[]) {
  int min = INF, min_index;
  for (int v = 0; v < num_routers; v++) {
    if (!visited[v] && dist[v] <= min) {
      min = dist[v];
      min_index = v;
    }
  }
  return min_index;
}

void dijkstra(int src) {
  int dist[MAX_ROUTERS];
```

```
    bool visited[MAX_ROUTERS];
    for (int i = 0; i < num_routers; i++) {
        dist[i] = INF;
        visited[i] = false;
    }
    dist[src] = 0;
    for (int count = 0; count < num_routers - 1; count++) {
        int u = min_distance(dist, visited);
        visited[u] = true;
        for (int v = 0; v < num_routers; v++) {
            if (!visited[v] && network[u].neighbors[v] != INF &&
                dist[u] != INF && dist[u] + network[u].neighbors[v] < dist[v]) {
                dist[v] = dist[u] + network[u].neighbors[v];
            }
        }
    }
    printf("Router %d Routing Table:\n", src);
    for (int i = 0; i < num_routers; i++) {
        printf("To %d: Cost %d\n", i, dist[i]);
    }
}

void simulate_ospf() {
    for (int i = 0; i < num_routers; i++) {
        if (network[i].active) {
            dijkstra(i);
        }
    }
}

int main() {
    initialize_network(5);
    add_link(0, 1, 10);
    add_link(1, 2, 20);
    add_link(2, 3, 5);
    add_link(3, 4, 15);
    add_link(0, 4, 30);

    printf("Initial OSPF Routing:\n");
    simulate_ospf();
```

```
    printf("\nSimulating Router 2 Failure:\n");
    remove_router(2);
    simulate_ospf();

    return 0;
}
```

SCREENSHOTS

```c

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <string.h>
#include <stdbool.h>

#define MAX_ROUTERS 10
#define INF INT_MAX

typedef struct {
    int id;
    bool active;
    int neighbors[MAX_ROUTERS];
    int costs[MAX_ROUTERS];
} Router;

Router network[MAX_ROUTERS];
int num_routers;

void initialize_network(int n) {
    num_routers = n;
    for (int i = 0; i < n; i++) {
        network[i].id = i;
        network[i].active = true;
        for (int j = 0; j < n; j++) {
            network[i].neighbors[j] = (i == j) ? 0 : INF;
            network[i].costs[j] = (i == j) ? 0 : INF;
        }
    }
}

void add_link(int src, int dest, int cost) {
    network[src].neighbors[dest] = cost;
    network[dest].neighbors[src] = cost;
    network[src].costs[dest] = cost;
    network[dest].costs[src] = cost;
}

void remove_router(int id) {
    network[id].active = false;
    for (int i = 0; i < num_routers; i++) {
        network[i].neighbors[id] = INF;
        network[i].costs[id] = INF;
    }
}

```

```c
int min_distance(int dist[], bool visited[]) {
    int min = INF, min_index;
    for (int v = 0; v < num_routers; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
}

void dijkstra(int src) {
    int dist[MAX_ROUTERS];
    bool visited[MAX_ROUTERS];
    for (int i = 0; i < num_routers; i++) {
        dist[i] = INF;
        visited[i] = false;
    }
    dist[src] = 0;
    for (int count = 0; count < num_routers - 1; count++) {
        int u = min_distance(dist, visited);
        visited[u] = true;
        for (int v = 0; v < num_routers; v++) {
            if (!visited[v] && network[u].neighbors[v] != INF &&
                dist[u] != INF && dist[u] + network[u].neighbors[v] < dist[v]) {
                dist[v] = dist[u] + network[u].neighbors[v];
            }
        }
    }
    printf("Router %d Routing Table:\n", src);
    for (int i = 0; i < num_routers; i++) {
        printf("To %d: Cost %d\n", i, dist[i]);
    }
}

void simulate_ospf() {
    for (int i = 0; i < num_routers; i++) {
        if (network[i].active) {
            dijkstra(i);
        }
    }
}
```

```c
int main() {
    initialize_network(5);
    add_link(0, 1, 10);
    add_link(1, 2, 20);
    add_link(2, 3, 5);
    add_link(3, 4, 15);
    add_link(0, 4, 30);

    printf("Initial OSPF Routing:\n");
    simulate_ospf();

    printf("\nSimulating Router 2 Failure:\n");
    remove_router(2);
    simulate_ospf();

    return 0;
}
```

```
┌──(pranulegend㉿kali)-[~/Desktop]
└─$ gcc LAB10.C

┌──(pranulegend㉿kali)-[~/Desktop]
└─$ ./a.out
Initial OSPF Routing:
Router 0 Routing Table:
To 0: Cost 0
To 1: Cost 10
To 2: Cost 30
To 3: Cost 35
To 4: Cost 30
Router 1 Routing Table:
To 0: Cost 10
To 1: Cost 0
To 2: Cost 20
To 3: Cost 25
To 4: Cost 40
Router 2 Routing Table:
To 0: Cost 30
To 1: Cost 20
To 2: Cost 0
To 3: Cost 5
To 4: Cost 20
Router 3 Routing Table:
To 0: Cost 35
To 1: Cost 25
To 2: Cost 5
To 3: Cost 0
To 4: Cost 15
Router 4 Routing Table:
To 0: Cost 30
To 1: Cost 40
To 2: Cost 20
To 3: Cost 15
To 4: Cost 0
```

```
Simulating Router 2 Failure:
Router 0 Routing Table:
To 0: Cost 0
To 1: Cost 10
To 2: Cost 2147483647
To 3: Cost 45
To 4: Cost 30
Router 1 Routing Table:
To 0: Cost 10
To 1: Cost 0
To 2: Cost 2147483647
To 3: Cost 55
To 4: Cost 40
Router 3 Routing Table:
To 0: Cost 45
To 1: Cost 55
To 2: Cost 2147483647
To 3: Cost 0
To 4: Cost 15
Router 4 Routing Table:
To 0: Cost 30
To 1: Cost 40
To 2: Cost 2147483647
To 3: Cost 15
To 4: Cost 0
```