Develop a Basic Network Traffic Simulator for a Telecom Company

Overview:

You are tasked with designing, building, and deploying a basic network traffic simulator within **3 hours**. This simplified project focuses on simulating network traffic in a telecommunications network with minimal parameters. It will assess your problem-solving abilities, analytical thinking, troubleshooting skills, capacity for simple yet effective design solutions, proficiency in web development, and basic database modelling and design capabilities.

Requirements:

1. **Simulation Engine (Backend):**
    - Use **Node.js** and **Express.js** to build the server-side application.
    - Simulate a simple network consisting of multiple interconnected nodes (e.g., routers or switches).
    - Implement logic to simulate data packets being transmitted through the network.
    - Simulate variable network traffic loads based on provided traffic generation rates (**optional**).

2. **Visualization Dashboard (Frontend):**
    - Develop a responsive web interface using React, Angular, Vue or anything that you prefer.
    - Display a graphical representation of the network topology (Use any charting library or a simple div structure).
    - Show real-time statistics for each node and link, such as:
        - Current traffic generation load (e.g., packets generated per second).
        - Link Load (e.g. packets passing through the link per second)
        - Packets in queue (e.g. number of packets waiting at node)
    - Provide controls to (**This is optional**):
        - Adjust traffic generation rates for different nodes or links.
        - Modify network parameters like link capacities or node processing speeds.
        - Start, pause, and reset the simulation.

3. **Data Management:**
    - Use a simple data storage solution (e.g., in-memory data structures or JSON files).
    - No complex database modeling is required.
    - Store simulation parameters and state data.

4. **Algorithm Implementation:**
    - Implement a basic routing algorithm (e.g., shortest path routing).
    - Calculate network load and adjust packet flow based on link capacities.
    - Update node and link statuses at each simulation step.
    - Handle simple congestion control by queuing packets when necessary.

5. **Code Commit and Deployment:**
    - Commit the code to GitLab or GitHub as public repository. Share the repo link.
    - Deploy the application on a cloud platform like Heroku, Netlify, or GitHub Pages (for frontend) and Heroku or Railway (for backend) or any other choice of platform that you prefer.
    - Share the public deploy application URL
    - Set up any necessary environment variables and configurations for deployment.
    - Repo Link and Application URL shall be shared on HR@DigiPlusIT.com with your name, roll number and contact information.

Dataset Overview

- **Network Topology:** A simple network with 5 nodes labeled A, B, C, D, and E.
- **Links:** Connections between nodes with specified capacities.
- **Parameters:**
    - **Traffic Generation Rate (packets per second):** Number of packets generated at each node.
    - **Link Capacity (packets per second):** Maximum number of packets that can be transmitted through a link per second

Sample Data

Nodes Traffic Generation Rates

```
const trafficRates = {
  '08:00': { A: 50, B: 30, C: 40, D: 20, E: 60 },
  '08:15': { A: 55, B: 35, C: 45, D: 25, E: 65 },
  '08:30': { A: 60, B: 40, C: 50, D: 30, E: 70 },
  '08:45': { A: 55, B: 35, C: 45, D: 25, E: 65 }
};
```

Network Links and Capacities

```
const links = [
  { from: 'A', to: 'B', capacity: 100 },
  { from: 'A', to: 'C', capacity: 80 },
  { from: 'B', to: 'D', capacity: 70 },
  { from: 'C', to: 'D', capacity: 90 },
  { from: 'C', to: 'E', capacity: 100 },
  { from: 'D', to: 'E', capacity: 60 }
];
```

Hints:

Algorithm Input

- **Traffic Generation:** At each time step (e.g., every simulated second), each node generates packets based on its traffic generation rate.
- **Packet Routing:** Packets are routed through the network from their source node to a randomly assigned destination node.
- **Link Utilization:** Packets consume capacity on each link they traverse.

Routing Algorithm

- **Shortest Path Routing:** Use a basic shortest path algorithm to determine the route from source to destination.
- **Capacity Checking:** Before transmitting packets over a link, check if the link has available capacity.
    - If the link is at capacity, packets are queued at the source node or previous node.
    - Implement a simple queuing mechanism with a maximum queue size (optional).