Project requirements

Name: **Muhammad Hamza Shahab, Syed Haider Abbas Naqvi**

Community & UN SDG(s): **SaskTel network engineers and architects**

UN SDG(s):

- SDG#7: Affordable and clean energy
- SDG#11: Sustainable cities and communities
- SDG#12: Responsible consumption and production
- SDG#13: Climate action

Date: **February 8th 2025**

Project Name	Eco-Resilient Networks: Smart Deployment for the Future

Functional Requirements

1. **Network Simulation:** Users can create and test various network setups, such as standard fat-tree and custom topologies, by defining nodes, links, capacities, and energy use.
2. **Service Function Chains:** The user will be allowed to define SFCs by selecting, among others, firewall, load balancer, and IDS VNFs, and specify resource requirements and availability. Support for custom VNFs will also be provided.
3. **SFC Embedding Policies:** The system will provide a variety of SFC placement strategies, including:
    a. **Availability-Aware Embedding (AAE) -** Focuses on maximizing **availability** of the system.
    b. **Carbon-Aware Embedding (CAE) –** Focuses on reducing **carbon footprint** of the system.
    c. **Trade-off-Aware Embedding (TAE) –** Focuses on **achieving balance** between availability and carbon footprint of system.
4. **Redundancy Optimization Algorithms:** User will be able to choose between different optimization methods to achieve availability with a minimum carbon footprint such as:
    a. Simulated Annealing – SA
    b. Particle Swarm Optimization – PSO
    c. Genetic Algorithm – GA
5. **Key Performance Metrics:** Following key performance metrics will be used to evaluate the performance of the algorithms:
    a. **Availability –** The percentage of uptime.
    b. **Carbon Footprint -** Energy consumption based on node/link utilization.
    c. **Latency -** End-to-end delay in the network.
    d. **Execution Time –** The time taken for each algorithm to execute.
6. **Running Simulations:** Users can simulate with diverse network setups, SFC configurations, embedding strategies, and optimization techniques to test performance.
7. **Results Visualization:** Results will be clearly represented and interactive, including:
    a. Network topology and VNF placement.
    b. Metrics such as availability, carbon footprint, and latency.
    c. Resource utilization and policy comparisons.
8. **Export & Customization:** The user will export the simulation results in **CSV** and **JSON** formats and customize parameters: SFC types, node counts, redundancy settings.

Technical/Performance Requirements

TECHNOLOGY REQUIREMENTS

- **Programming Language:** The system will be built primarily in **Python**, using libraries like **NetworkX**, **NumPy**, **Pandas**, and **DEAP**.

- **Simulation Environment:** It will run on a discrete-event simulation framework, based on **CloudSim**.
- **Data Handling:** The system will handle data primarily in **CSV** and **JSON** formats.

PERFORMANCE REQUIREMENTS

- **Scalability & Performance:** Initial tests should handle **100 nodes** and **20 SFCs**, with future expansion in mind. Optimization algorithms should find solutions within a reasonable time (e.g., PSO solving **50 nodes** & **10 SFCs** in under **5 minutes**).
- **Service Level Agreement (SLA) Metrics & Constraints:** The simulation environment should be the compliant to the following constraints:
    - **Availability (%) –** The availability achieved by the algorithms should be above **99.9%** as per the ETSI standards.
    - **Latency (ms) –** The latency achieved by the algorithms should not exceed more than **100ms** for each SFC.
    - **Number of redundant instances –** The number of replica instances for each VNF should not exceed **3 instances**.
    - **Execution time (seconds) –** The time taken for an algorithm's execution should not be more than **5 minutes**.
- **Accuracy:** Simulations should maintain a **5% margin of error** when compared to analytical models, requiring proper validation.
- **Modularity & Compatibility:** The system should be modular for easy updates and Linux-compatible.
- **Deployment:**
    - **MVP:** A standalone Python application running on a single machine.
    - **Future:** Cloud/edge deployment with Docker & Kubernetes.