**Name:** Anastcaia Muhamamd
**LSU ID:** 890607568
**Course:** CSC 4501
**Assignment:** Assignment 4
**Date:** 5/9/2025

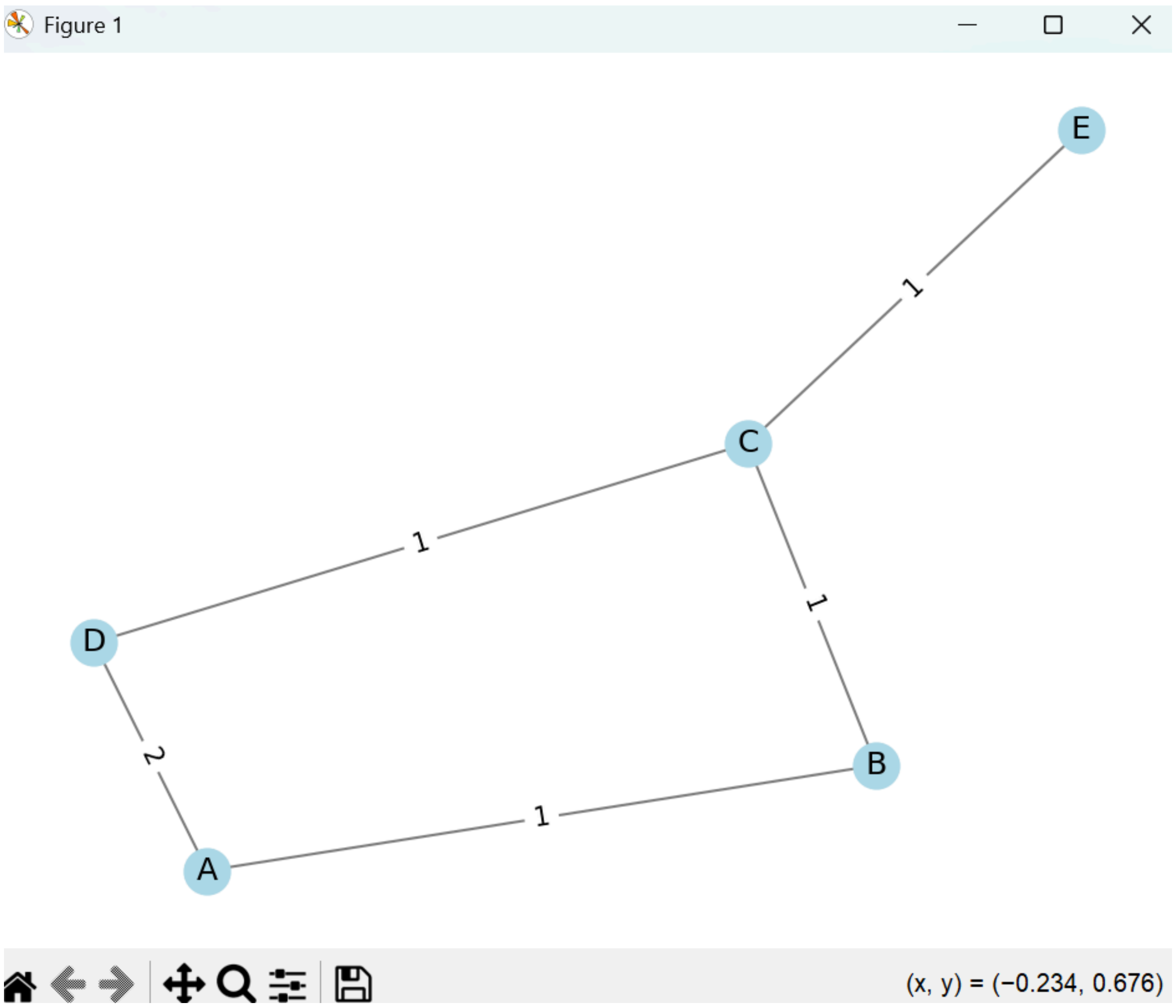
Architecture:

- The SDN controller is implemented in Python using networkx for the topology and matplotlib for visualization. It computes shortest paths, generates flow tables, handles link failures, and updates paths dynamically
-

Policies:

- I followed load-balancing by using random.choice () over the shortest paths. Along with handling NORMA and HIGH flows, with backup support after link failures.

Visuals:

- 

CLI Commands:

- flow A E [HIGH]
- fail B C
- table A
- show
- add F
- link A F 2
- exit

Challenges:

- I originally implemented nx.shortest_path(),  which returned only one path. I replaced it with: list(nx.all_shortest_paths()) that selects a random path to balance traffic, as stated in the requirements.
- Original:
  ```
  path = nx.shortest_path(network, source=src, target=dst, weight='weight')
  ```
- Fixed:
  ```
  paths = list(nx.all_shortest_paths(network, source=src, target=dst, weight='weight'))
  chosen_path = random.choice(paths)
  ```

Watermark: 4973e767e3f4694d82fc203ae62cd4a8f5fa0268a3c4fb0a0fa7ae7b99f300b1

Reflection:
This project taught me how to combine topology management, traffic prioritization, and CLI control into a single SDN controller. Implementing shortest path routing was straightforward, but adding support for multiple equal-cost paths and priority-based traffic injection showed me how real networks handle fairness and congestion control. I gained experience with failure recovery logic and the importance of visualizing network behavior to catch flow issues. The CLI added a practical way to test different network scenarios, I learned how small scheduling and routing decisions can significantly affect overall network performance. This experience deepened my appreciation for scalable, transparent network design and the challenges behind traffic engineering in software-defined networking.