SDN Controller Design Document for CSC 4501

Author: Alyx Whipp

Version: 0.1

Last Updated: 5/9/25

1. Introduction

This document outlines the design and architecture of the SDN (Software-Defined Networking) Controller, which manages network topology, flow routing, and traffic prioritization in a simulated environment.

The SDN Controller provides:

- Dynamic path computation based on traffic priority.
- Link failure detection and rerouting.
- Traffic injection with different priority levels.
- Visualization of network topology and link utilization.
- **SDN**: Software-Defined Networking
- **CLI**: Command-Line Interface
- **Flow Table**: Rules that dictate how traffic is routed between switches
- **Topology**: The arrangement of nodes (switches) and links in the network

2. System Architecture

The system consists of:

1. **SDNController Class** – Core logic for topology management, path computation, and flow handling.
2. **SDNCLI Class** – Command-line interface for user interaction.
3. **NetworkX** – Used for graph-based topology representation and pathfinding.

DataFlow

1. User inputs commands via CLI (e.g., `add_node`, `inject_flow`).
2. CLI calls corresponding methods in `SDNController`.
3. `SDNController` updates topology, computes paths, and manages flows.
4. Results are displayed back to the user.

3. Detailed Design

SDNController Class

- `topology` (NetworkX Graph) – Stores nodes and weighted edges.
- `flow_tables` (dict) – Maps switches to destination paths.
- `traffic` (dict) – Tracks active flows and their priorities.
- `link_utilization` (dict) – Monitors bandwidth usage per link.
- `backup_paths` (dict) – Stores alternative paths for failover.

Methods:

Method	Description
`add_node()`	Adds a node to the topology.
`add_link()`	Adds a bidirectional link with weight and capacity.
`remove_link()`	Removes a link and triggers failover handling.
`compute_paths()`	Recomputes all paths using Dijkstra's algorithm.

`inject_flow()`	Injects traffic with a given priority (critical/important/default).
`_handle_link_failure()`	Reroutes flows when a link fails.
`show_utilization()`	Displays link usage statistics.
`show()`	Visualizes the topology with Matplotlib.

SDNCLI Class

Command	Description
`add_node`	Adds one or more nodes.
`add_link`	Connects two nodes with optional weight/capacity.
`remove_link`	Removes a link between nodes.
`inject_flow`	Starts a flow between nodes with a priority level.
`fail_link`	Simulates a link failure.
`show_util`	Shows link utilization stats.
`show`	Displays the topology graphically.
`watermark`	Generates a cryptographic watermark (SHA-256).

4. Algorithms & Logic

Path Computation

- Uses a built in **Dijkstra’s algorithm** for shortest-path routing.
- **Load balancing** is applied for non-critical traffic by selecting the least congested path.

- **Backup paths** are stored for failover scenarios.

Traffic Prioritization

Priority Level	Path Selection Strategy
Critical (3)	Shortest path (lowest weight).
Important (2)	Load-balanced path (least utilized).
Default (1)	Best-effort routing.

Failure Handling

1. When a link fails, affected flows are identified.
2. Backup paths are used for rerouting.
3. If no backup exists, the flow is dropped.

5. User Interface

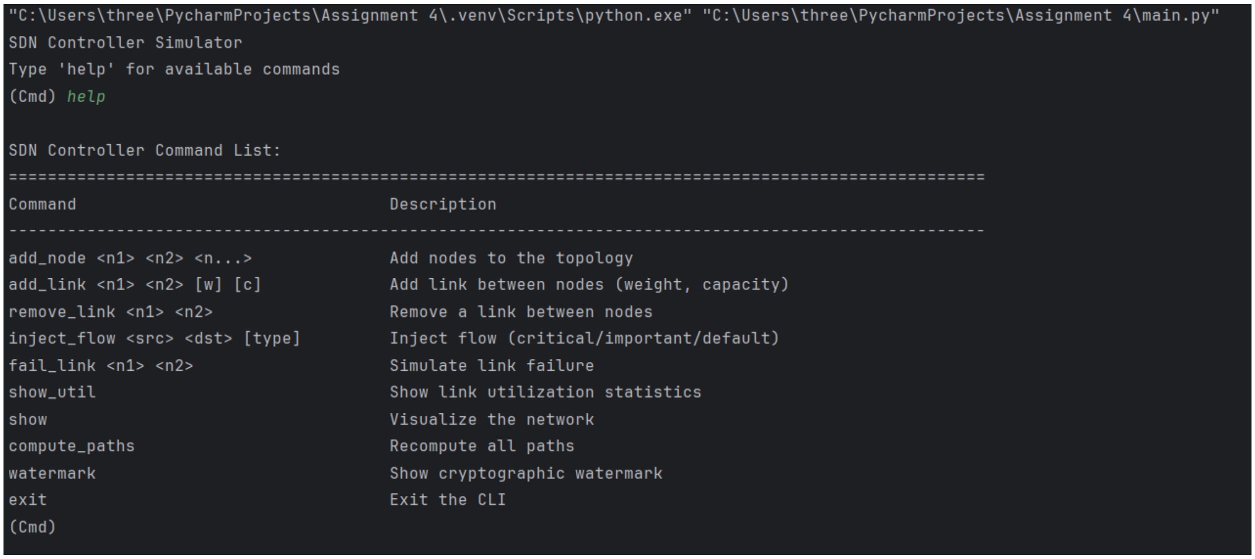
- Interactive command-line interface.
- Help menu lists all available commands.
- Topology visualization via Matplotlib.
- Nodes are represented as circles.
- Links are lines with weights.
- Active flows:
  - **Critical**: Red
  - **Important**: Orange
  - **Default**: Green

6. Limitations & Future Work

Limitations and Potential Changes

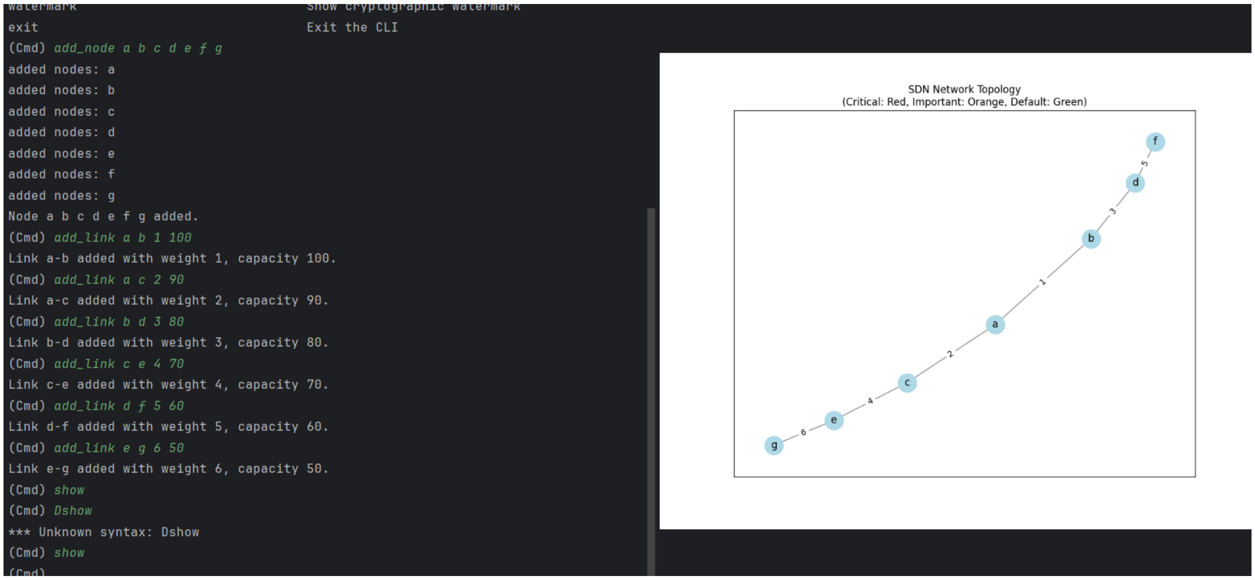
- No Preloaded data makes adding time consuming
- Tends to produce incorrect results as you increase the amount of of nodes and links
- Sorta Annoying to understand
- Using off the wall algorithm for routing which may or may not work at times
- Show commmand doesn’t produce consisent charts and can lead to funky generated diagrams

7. Examples



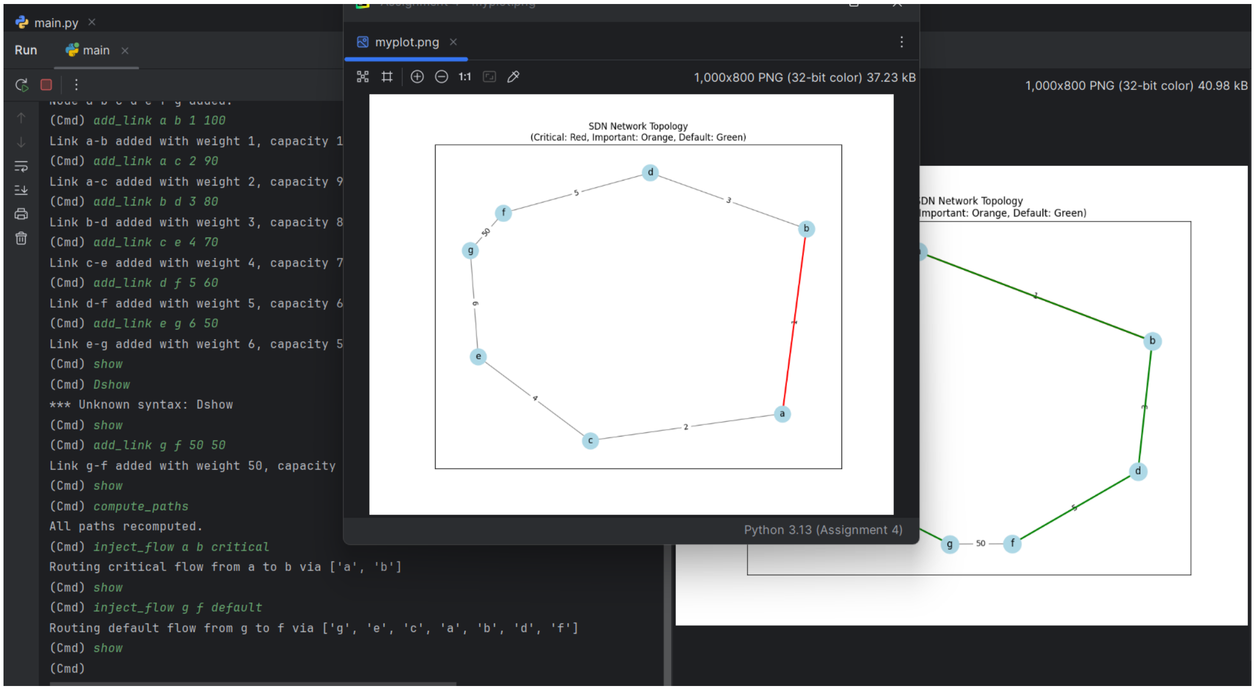
```
"C:\Users\three\PycharmProjects\Assignment 4\.venv\Scripts\python.exe" "C:\Users\three\PycharmProjects\Assignment 4\main.py"
SDN Controller Simulator
Type 'help' for available commands
(Cmd) help

SDN Controller Command List:
====================================================================================================
Command                                Description
----------------------------------------------------------------------------------------------------
add_node <n1> <n2> <n...>              Add nodes to the topology
add_link <n1> <n2> [w] [c]             Add link between nodes (weight, capacity)
remove_link <n1> <n2>                  Remove a link between nodes
inject_flow <src> <dst> [type]         Inject flow (critical/important/default)
fail_link <n1> <n2>                    Simulate link failure
show_util                              Show link utilization statistics
show                                   Visualize the network
compute_paths                          Recompute all paths
watermark                              Show cryptographic watermark
exit                                   Exit the CLI
(Cmd)
```

Overloaded CMD help screen



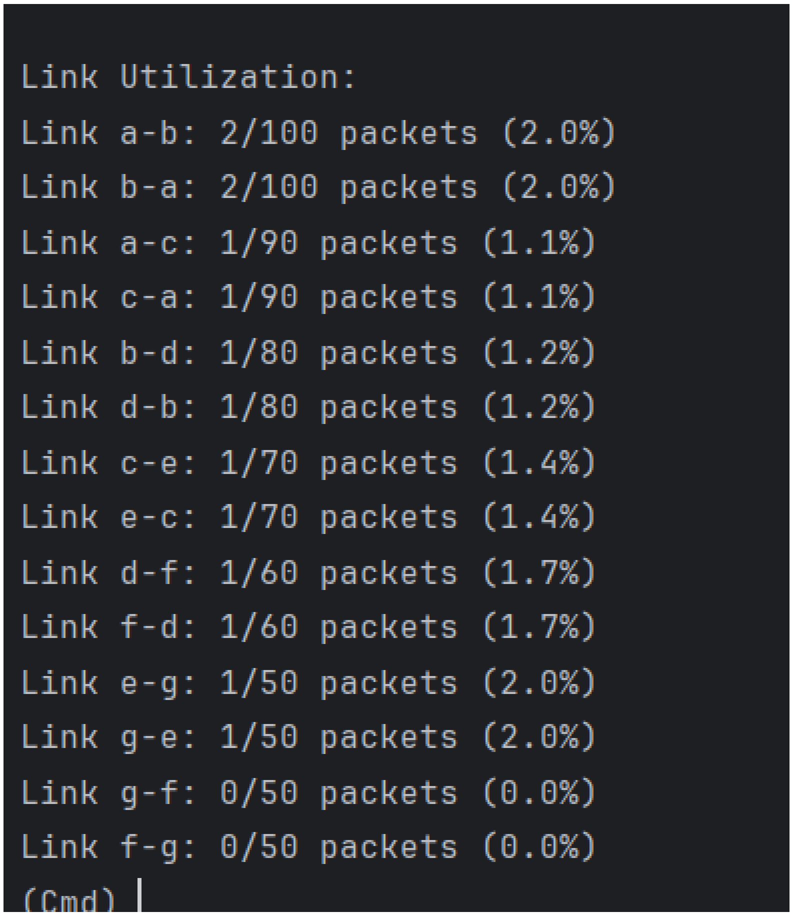
```
watermark                              Show cryptographic watermark
exit                                   Exit the CLI
(Cmd) add_node a b c d e f g
added nodes: a
added nodes: b
added nodes: c
added nodes: d
added nodes: e
added nodes: f
added nodes: g
Node a b c d e f g added.
(Cmd) add_link a b 1 100
Link a-b added with weight 1, capacity 100.
(Cmd) add_link a c 2 90
Link a-c added with weight 2, capacity 90.
(Cmd) add_link b d 3 80
Link b-d added with weight 3, capacity 80.
(Cmd) add_link c e 4 70
Link c-e added with weight 4, capacity 70.
(Cmd) add_link d f 5 60
Link d-f added with weight 5, capacity 60.
(Cmd) add_link e g 6 50
Link e-g added with weight 6, capacity 50.
(Cmd) show
(Cmd) Dshow
*** Unknown syntax: Dshow
(Cmd) show
(Cmd)
```

Adding nodes and links


main.py
Run
main
myplot.png
1:1
1,000x800 PNG (32-bit color) 37.23 kB
1,000x800 PNG (32-bit color) 40.98 kB
SDN Network Topology
(Critical: Red, Important: Orange, Default: Green)
d
f
b
g
e
a
c
5
3
50
6
1
4
2
DN Network Topology
mportant: Orange, Default: Green)
b
d
g
f
50
Python 3.13 (Assignment 4)

```
(Cmd) add_link a b 1 100
Link a-b added with weight 1, capacity 1
(Cmd) add_link a c 2 90
Link a-c added with weight 2, capacity 9
(Cmd) add_link b d 3 80
Link b-d added with weight 3, capacity 8
(Cmd) add_link c e 4 70
Link c-e added with weight 4, capacity 7
(Cmd) add_link d f 5 60
Link d-f added with weight 5, capacity 6
(Cmd) add_link e g 6 50
Link e-g added with weight 6, capacity 5
(Cmd) show
(Cmd) Dshow
*** Unknown syntax: Dshow
(Cmd) show
(Cmd) add_link g f 50 50
Link g-f added with weight 50, capacity
(Cmd) show
(Cmd) compute_paths
All paths recomputed.
(Cmd) inject_flow a b critical
Routing critical flow from a to b via ['a', 'b']
(Cmd) show
(Cmd) inject_flow g f default
Routing default flow from g to f via ['g', 'e', 'c', 'a', 'b', 'd', 'f']
(Cmd) show
(Cmd)
```

Multi injects can cause previous runs to be not shown



```
Link Utilization:
Link a-b: 2/100 packets (2.0%)
Link b-a: 2/100 packets (2.0%)
Link a-c: 1/90 packets (1.1%)
Link c-a: 1/90 packets (1.1%)
Link b-d: 1/80 packets (1.2%)
Link d-b: 1/80 packets (1.2%)
Link c-e: 1/70 packets (1.4%)
Link e-c: 1/70 packets (1.4%)
Link d-f: 1/60 packets (1.7%)
Link f-d: 1/60 packets (1.7%)
Link e-g: 1/50 packets (2.0%)
Link g-e: 1/50 packets (2.0%)
Link g-f: 0/50 packets (0.0%)
Link f-g: 0/50 packets (0.0%)
(Cmd)
```

However the data still exits in here its just not visualized

8. Challenges

I think in the process of making this i relied on out the box solutions a bit too much and it led to certain things not really working as i wanted them too. I am not super happy with the way the the graphing works and I think matplotlib wasn’t the tool i should have used despite knowing it the most. I also faced a challenge with the shortest path algorithm attempting to orginally implement A* as it checks actual distance and not just weight but it became problematic while implementing so i just reverted back to a simple weight system for the built in functions to work. I unfortunately also do not have screenshots of old code as i didn’t read the uniqueness verification until i was almost done with the assignment.

9.Crypt Hash

Yeah I'm not really sure what this means when the worksheet says to reference this in design document but the hash itself is 7e96a087bf275b6a79d3607b7ff01810f5fa9bc9ca1b839410d7fc1dedc4fd2b

The watermark itself is in teh code and you can also type watermark to print it in the command line if you so choose. It doesn’t really relate to anything and tbh its probably somehow wrong because python gives me a different hash than what random websites on teh internet give me so I'm really not quite sure whats up with that.