Negative Test Cases

This document outlines the negative test cases considered for testing the packet forwarding functionality in the provided network simulation framework.

Test Case 1: Invalid Topology File

This test case has been precluded from consideration as issues arising from an invalid topology file are already handled within the framework code. The framework includes robust checks to ensure the topology file is well-formed and valid.

Test Case 2: Sending Data to a Non-Existent Destination Router

In this test case, the simulation attempts to forward data to a destination router that does not exist in the forwarding information base (FIB). The following approaches are employed to simulate this scenario:

Automated Negative Test

The code automatically generates a random router name and uses it as the destination for sending data. This serves as the default negative test.

The relevant code snippet is as follows:

```
# Random router name generation for negative test
random_router_name = generate_random_router_name()
sendData(random_router_name, data)
```

Manual Negative Test

Users have the option to manually input a non-existent router name while running the program. This can be done by setting the third command-line argument to a router name that does not exist in the network. For example, to test this scenario, you can run the program as follows:

```
python main.py net.json R01 R_non_exist
```

Test Case 3: Disconnected Graphs

Considering the default `net.json` represents a connected graph, the scenario of a disconnected graph has also been contemplated. To account for this, a modification was made to the provided `net.json` file, introducing an isolated router to the topology:

```
{ "Router": "R21", "Links": {} }
```

This isolated router (`R21`) is not connected to any other routers, which tests the system's ability to handle disjoint sections within the network topology.