Assignment 2: Network Emulation with IPv4 Routing

Due date: *Thursday, April 11, 2024 at 22:00*

> *This is an individual assignment. You may discuss it with others, but your code and documentation must be written on your own.*

In a source file called `emulation.py` write a Python program that takes as input the definition of a network topology in YAML format, and creates an emulation of that network using Mininet. The general usage of the program should be the following

```
$ ./emulation.py --help
usage: emulation.py [-h] [-d] definition

A tool to define the emulation of a network.

positional arguments:
  definition   the definition file of the network in YAML

options:
  -h, --help   show this help message and exit
  -d, --draw   output a map of the routers in GraphViz format
```

The topology definition consists of two sections, *routers* and *hosts*, as shown in the example of Figure 1. The routers section (`routers:`) specifies a number of routers, each identified by a unique name (e.g., `r1` in Figure 1). You may assume that no router name matches the regular expression `s[0-9]+` (explained later). A router has zero or more interfaces. Each interface is characterized by an IPv4 address, a network mask, and an optional cost associated with the link. The cost of a shared link can be specified with any of the interfaces that share that link. If no cost is specified, the default should be 1. You may assume that the cost specifications are consistent (same cost for adjacent interfaces).

The hosts section (`hosts:`) specifies a number of hosts, each identified by a unique host name that also does not match the regular expression `s[0-9]+`. A host has one interface, characterized by an IPv4 address and a network mask. The host interface is guaranteed to be in a subnet with a single router interface, which you can therefore use as default gateway for that host.

You are guaranteed that the input topology definition file is valid. For instance, the router and host sections will always be present, the file will be in valid YAML format, there will be no duplicate IP addresses, etc.

To parse the input YAML file, you may use the *PyYAML* python package. The `requirements.txt` file available on the iCorsi system lists all the external packages that you are allowed to use[1]. You may not use any other package or non-standard library function. To install these Python packages into a separate environment you might want to create a virtual environment[2].

The topology definition is such that every interface is connected to at least another interface from some other node (router or host). If multiple hosts share the same subnet address, you must add a layer-2 switch to interconnect them. The switches will not be listed in the configuration file. You can name those switches `s1`, `s2`, etc. This is why the definition guarantees that hosts and router names do not conflict with such switch names.

[1] You may want to use the `pip install -r requirements.txt` command to install all the packages you can use with their versions.

[2] https://docs.python.org/3/library/venv.html

```yaml
---
routers:
  r1:
    eth0:
      address: 192.168.0.1
      mask: 255.255.255.252
      cost: 5
    eth1:
      address: 192.168.1.1
      mask: 255.255.255.248
      cost: 2
    eth2:
      address: 192.168.0.5
      mask: 255.255.255.252
  r2:
    eth0:
      address: 192.168.0.2
      mask: 255.255.255.252
      cost: 5
    eth1:
      address: 192.168.1.2
      mask: 255.255.255.248
    eth2:
      address: 10.0.2.1
      mask: 255.255.255.0
  r3:
    eth0:
      address: 192.168.0.6
      mask: 255.255.255.252
    eth1:
      address: 192.168.1.3
      mask: 255.255.255.248
    eth2:
      address: 10.0.3.1
      mask: 255.255.255.0

hosts:
  h1:
    eth0:
      address: 10.0.2.2
      mask: 255.255.255.0
  h2:
    eth0:
      address: 10.0.2.3
      mask: 255.255.255.0
  h3:
    eth0:
      address: 10.0.3.2
      mask: 255.255.255.0
  h4:
    eth0:
      address: 10.0.3.3
      mask: 255.255.255.0
```

Figure 1: Example topology definition in YAML format.

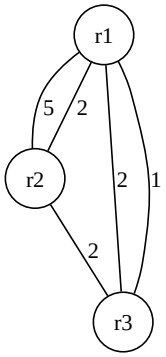

Figure 2: Rendering of GraphViz representation obtained from the output of the `--draw` option.

When called with the `--draw` option, the program must write onto the standard output a GraphViz[3] representation of the input topology. The GraphViz representation should show only the routers in the network and the connections between them with their associated cost. For example, using the topology in the example of Figure 1, the program should print something similar to the following:

```
$ ./emulation.py --draw topology.yaml
graph Network {
    r1 [shape=circle];
    r2 [shape=circle];
    r3 [shape=circle];
    r1 -- r2 [label="5"];
    r1 -- r2 [label="2"];
    r1 -- r3 [label="2"];
    r2 -- r3 [label="2"];
    r1 -- r3 [label="1"];
}
```

Using the `dot` tool, you can render the topology to obtain a picture similar to the one in Figure 2.

With no parameters, the program start a Mininet emulation. The emulation must be set up so as to provide full connectivity. In practice, this means that you must properly set up the forwarding tables for every router and host. The routing must be based on the topology, as well as the cost metrics defined in the YAML configuration file. If multiple best-cost paths are available, you may choose any one of them. To compute the shortest (best cost) paths between every pair of routers in the network, you may want to use the Floyd-Warshall or Dijkstra algorithms.

Submission Instructions

Submit a source file named `emulation.py` through the iCorsi system. Add comments to your code to explain sections of the code that might not be clear. You must also add comments at the beginning of the source file to properly acknowledge any and all external sources of information you may have used, including code, suggestions, and comments from other students. If your implementation has limitations and errors you are aware of (and were unable to fix), then list those as well in the initial comments.

You may use an integrated development environment (IDE) of your choice. However, *do not submit any IDE-specific file*, such as project description files.

[3] https://graphviz.org/doc/info/lang.html