Wesleyan University, Spring 2023, COMP 332

Homework 9: Distance Vector Routing

Due by 11:59pm on May 10, 2023

---

1. Coding and hands-on problems (20 points)

In this problem, you will implement the distance vector routing algorithm. Recall that in the distance vector algorithm a node keeps track of the link costs it has with each of its neighbors. A node also keeps track of its own distance to every node in the network in what is called a distance vector. Nodes exchange their distance vectors with their neighbors. Each node x, updates its distance vector using its link costs and neighbor distance vectors with the following dynamic programming update, where N is the set of nodes in the network:

$$D_x(y) \quad = \quad \min_v c(x,v) + D_v(y) \text{ for each node } y \in N$$

You should implement your code in a `router.py` file that will serve as an abstraction of a router node in a graph exchanging distance vectors with other router nodes. You should be able to start up your `router.py` programs independently and asynchronously from each other. You should use the example graph shown in Figure 1 to test your code. Each node in the graph will correspond to a different instantiation of `router.py` running. Thus, you will run the same `router.py` program multiple times simultaneously on different command-line inputs as follows:

```
python3 router.py PORT NUM_ROUTERS NUM_NBRS NBR1 COST1 NBR2 COST2...
```

- `PORT` refers to the port number on which this particular router listens for distance vectors.
- `NUM_ROUTERS` refers to the total number of routers in the network (i.e., graph).
- `NUM_NBRS` is the number of neighbors that this particular router has in the network.
- `NBR_i` is the port number associated with neighbor i.
- `COST_i` is the cost associated with the link to neighbor i.

For instance, for the graph in Figure 1 you could run, in separate windows, the following commands. Sample output from doing this is shown in Figure 2

```
python3 router.py 50001 3 2 50002 2 50003 7
python3 router.py 50002 3 2 50001 2 50003 1
python3 router.py 50003 3 2 50001 7 50002 1
```

Notes:

- You will want to think carefully about how to best structure the data (e.g., link costs, distance vectors) that each router must store.
- Nodes should send their distance vectors to their neighbors once every second. A useful method to pause for 1 second is `time.sleep(1)`.

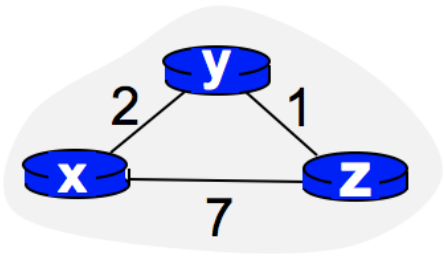

FIGURE 1. Graph topology on which to test your code for Problem 3.

- You will likely find it easier to work with UDP sockets rather than TCP sockets, but the choice of which to use is yours. I will not require you to add reliability to your UDP socket.
- You will need a message format to send a distance vector message to a node.
- You are not required to handle routers arriving and leaving. In practice, you would also need some mechanism to time out the distance vector state that a router has maintained for its neighbors, if the router has not heard from its neighbors within an acceptable period of time.
- Your code should be well-structured and use good programming practice.

2. SUBMISSION

Submit your coding as `router.py` to the Google Drive directory I have created for you named `comp332-s23-USERNAME/hw9/`. You should replace `USERNAME` with your Wesleyan username.

And make sure that at the top of each file you have put your name! Do not, however, change the names of the files.

```
>  python3 router.py 50001 3 2 50002 2 50003 7
----------------------------------------
Distance vector for node 50001:
{50001: 0, 50002: 2, 50003: 7}
----------------------------------------
Distance vector for node 50001:
{50001: 0, 50002: 2, 50003: 7}
----------------------------------------
Distance vector for node 50001:
{50001: 0, 50002: 2, 50003: 3}
----------------------------------------
Distance vector for node 50001:
{50001: 0, 50002: 2, 50003: 3}



> python3 router.py 50002 3 2 50001 2 50003 1
----------------------------------------
Distance vector for node 50002:
{50002: 0, 50001: 2, 50003: 1}
----------------------------------------
Distance vector for node 50002:
{50002: 0, 50001: 2, 50003: 1}
----------------------------------------
Distance vector for node 50002:
{50002: 0, 50001: 2, 50003: 1}
----------------------------------------
Distance vector for node 50002:
{50002: 0, 50001: 2, 50003: 1}
----------------------------------------



> python3 router.py 50003 3 2 50001 7 50002 1
----------------------------------------
Distance vector for node 50003:
{50003: 0, 50001: 7, 50002: 1}
----------------------------------------
Distance vector for node 50003:
{50003: 0, 50001: 3, 50002: 1}
----------------------------------------
Distance vector for node 50003:
{50003: 0, 50001: 3, 50002: 1}
----------------------------------------
Distance vector for node 50003:
{50003: 0, 50001: 3, 50002: 1}
----------------------------------------
Distance vector for node 50003:
{50003: 0, 50001: 3, 50002: 1}
----------------------------------------
```

FIGURE 2. Output traces for Problem 3.