Programming Assignment 6

Goal

- Understand and implement distance vector protocol

Description

In this assignment, we will simulate multiple routers in a single machine using multiple processes, and each process (aka “router”) will listen to a specified port for update messages from other neighboring routers. The start code for this assignment can be found on the course website.

How to start router

The code to start a router is in `start_router.py`. For example, run the following commands in 3 separate terminals will start 3 routers with topology defined here.

```
python start_router.py config/small-1
python start_router.py config/small-2
python start_router.py config/small-3
```

Router code is in `router.py`, where you need to add functionalities like sending update messages to neighboring routers, compute forwarding table, etc.

Router config file format

Configuration file format for each router is defined below. (Note: we restrict link costs to be positive integers)

```
router_id
neighbor_1_id,cost
neighbor_2_id,cost
…
neighbor_n_id,cost
```

router_id is an integer from 1 to n (where n is the number of routers in the given network). And for each of its directly connected neighbors, add one line with neighbor's router_id and link cost separated by a comma.

Inside `router.py`, there is a PeriodicClosure running to read the given config file every 5 seconds (configurable through `_CONFIG_UPDATE_INTERVAL_SEC` variable). We will simulate link cost update by editing config files, and router process will pick up the changes.

If router_id is i, then this router process will be listening update messages from neighbors in UDP port $8000 + i$ (configurable through `_BASE_ID` variable).

For example, if you start a router with the following config file

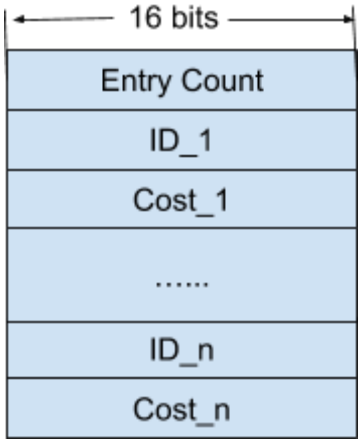
```
1
2,10
3,11
4,12
```

it means

- router_id is 1, and this router processing is listening at port 8001 for update messages from its neighbors.
- This router has 3 directly connected neighbors with router_id 2, 3, and 4 respectively.
- Link costs to router 2 is 10, to router 3 is 11, and to router 4 is 12.
- This router needs to send update message to port 8002 (for router 2), port 8003 (for router 3), and port 8004 (for router 4).

Update message format

The update message between routers to communicate forwarding information is defined below. To simplify the logic of distance vector, each router sends periodic update messages to all neighbors (e.g. every 5 seconds).



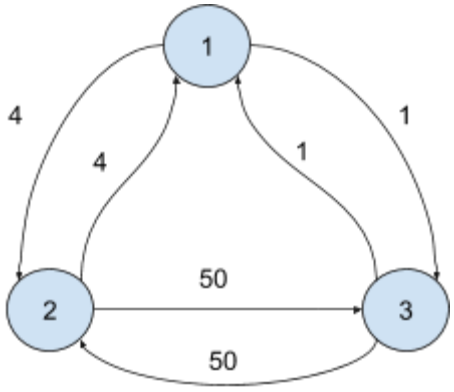
Forwarding table design

Forwarding table is defined in `table.py`. It has 3 columns: destination (router_id), next_hop (router_id), and cost. Your routing algorithm should process neighbors' update messages and update this forwarding table accordingly.

Testing your code

Case 1: small network

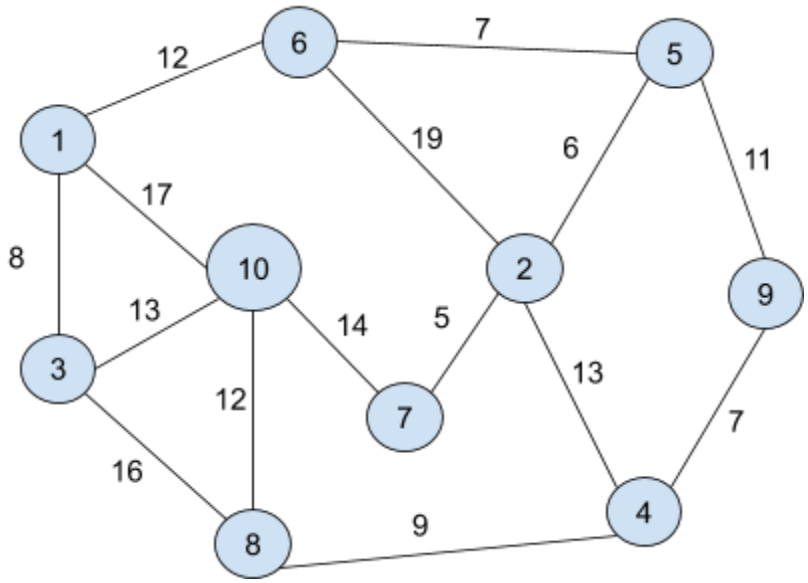
Below is a graph topology we covered in class. Configuration files are provided in `config/small-{1,2,3}`. Start 3 router processes with these config files and make screenshots of your forwarding table information to show me they computed the correct shortest path and next hop information.



We will also take a look at the 'Count-To-Infinity' problem. After all routers converge to the right shortest path, update link cost of (1,2) from 4 to 60 in `config/small-1`, and show me screenshots of your forwarding table changes that illustrate this problem. And after all routers converge to the right shortest path, update link cost of (1,2) from 60 back to 4, and show me screenshots of your forwarding table changes. Do you see convergent time difference between "good information" and "bad information"?

Case 2: large network

Below is a graph topology with 10 vertices. Configuration files are provided in `config/large-{1..10}`. Start 10 router processes with these config files and make screenshots of your forwarding table information to show me all routers converges to the correct shortest path and next hop information.



Feel free to update any link cost, and see how that information if propagated throughout, and check all routers can converge to the new right shortest paths.

Grading policy

100 points in total.

- [20 pts] Correctly send update messages using UDP socket. Follow the message format exactly.
- [50 pts] Demonstrate your code work in the small test case (with 3 routers).
    - [10 pts] Show all routers converge to shortest distance.
    - [20 pts] Show “bad information travels slow” by increasing the link cost.
    - [20 pts] Show “good information travels fast” by decreasing the link cost.
- [30 pts] Demonstrate your code work in the large test case (with 10 routers).

Submission instruction

Please your code and report to all TAs and cc z.sun@northeastern.edu by the deadline.