# CS 3251 - Spring 2017

# 3rd programming assignment

DUE DATE: Friday, April 21, 5pm

# *Distance vector routing*

## 1. Introduction

You are asked to write a simulator of distance vector routing. Your program will simulate the convergence of a distance vector routing protocol in a network with N routers. It will be a (single-threaded) process that will be simulating the exchange of routing messages (distance vectors) between N network nodes (routers).

The exchange of routing messages will be synchronous, meaning that time will be measured in "rounds" (round 1, 2, ..). In each round, every router will "move" once. During that move, the router will process the latest distance vectors it has received from other routers in the previous round, it will compute its new distance vector, and then it will send that distance vector to its neighboring routers.

To be consistent with the distance vector routing protocol, routers should only exchange distance vectors with their neighbors – communication with other routers is not allowed. Also, you should ***not*** compute the routing table of each router by running the shortest paths algorithm on the entire network topology. Instead, each router needs to compute its own routing table by only exchanging distance vectors with its neighbors.

How will you know whether the routing protocol has converged to a stable set of routing tables? The answer is that the routing table has converged at a round k if *none* of the N routers has made *any* changes in its distance vector during that round.

Your program will need to have three command-line arguments, described in the following sections.

## 2. Specification of the initial topology

Initially, your program should read a text file that will be provided as the **first command-line argument.** This file includes the number of routers N (in the first line), and a list of all network links with their costs (one link per line). The routers will be identified with the numbers 1, 2, .. N.

For example, the following is a description of a network with three routers (1, 2, 3), and with two links: one link of cost 10 between routers 1 and 2, and another link of cost 20 between routers 2 and 3:

```
3
1 2 10
2 3 20
```

This text file will not include any other data (no comments, no redundant numbers or characters, no formatting errors, no empty lines, etc).

## 3. Specification of topological events

The **second command-line argument** will be another text file that will describe certain topological changes ("events") in the given network. These events can be the removal or addition of a link, or a change in the cost of an existing link. The removal of a link will be represented by setting the link cost to -1.

Each line of this file represents a different topological event: the first number represents the time of the event (simulation round), the second and third numbers identify the link, and the fourth number represents the (potentially new) cost.

The events will be given in increasing temporal order.

For example, consider the following list of three events:

```
10 1 3 5
20 2 3 -1
30 2 3 4
```

The first line means that a link between routers 1 and 3 will be added at the (start of the) 10th round of the simulation; the cost of this link will be equal to 5.
The second line means that a link between routers 2 and 3 will be removed at the 20th round of the simulation.
The third line means that the link between routers 2 and 3 will be added back to the network with a cost of 4 at the 30th round.

**PLEASE NOTE:** You need to process all topological events, even if the routing protocol has converged before the execution of the next event. Similarly, a new topological event may occur even before the routing protocol has converged.

## 4. Three variations of distance vector routing

As you know, the basic distance vector routing algorithm is subject to “slow convergence” and “count-to-infinity” problems when the cost of a link increases or when a link is removed.

To experiment with these issues, you are asked to implement three slightly different variations of the distance vector protocol:

a) The “basic protocol” in which each router sends its complete distance vector to all its neighbors.
b) The “split horizon” variation in which router X does not send to router Y its distance to destination Z, if Y is the next-hop that X uses to reach Z.
c) The “split horizon with poison reverse” variation: as in “split horizon” but router X sends a cost of “inifinity” to router Y.

## 5. Experiments

Your program should output the following results for each of the previous three variations:

a) The final routing table for each router. The routing tables should be formatted as a matrix with N rows (one row for each router) and N column (one column for each destination). In each column, you need to show the next-hop (e.g., router 2) and the distance to that destination (e.g., 3 hops).
b) The number of rounds it took for the protocol to converge after the last topological event. For instance, if the last topological event took place at round-30, and the protocol converged at round-40, then the program should output “Convergence delay: 10 rounds”. **Note:** If one of the distances becomes larger than 100, your program should exit reporting that it has encountered a “count-to-infinity” instability.

The **third command-line argument** of your program should be a binary flag. If it is 0, it means that your program should only report the previous results (routing tables and convergence delay). If it is 1, it means that your program should produce a more detailed output, showing the routing table of each router at each round (in the previous format).

Please try to present these results in a nicely structured format so that the TAs (and you!) can easily understand the execution of the protocol.

## 6. What to submit

Please turn in well-documented source code, a README file, and a sample output file called sample.txt.

The README file must contain:

- Your name (or names for group projects), email addresses, date and assignment title
- Names and descriptions of all files submitted
- Detailed instructions for compiling and running your programs
- Any known bugs or limitations of your program

You must submit your program files online. Create a ZIP/tar archive of your entire submission. Use T-Square to submit your complete submission package as an attachment.

Only one member of each group needs to submit the actual code and documentation. The other group members can submit a simple text file in which they mention their partner's name that has submitted the actual assignment.