Project 2

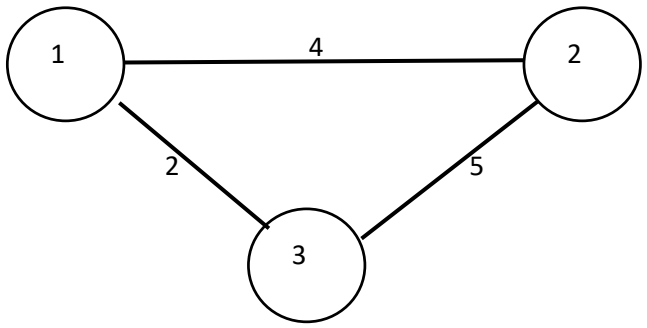
Due date: December 17, 11:59pm

The goal of this project is to Implement link state and distance vector routing algorithms. To make this simpler, it is broken up into several parts. These algorithms work on weighted graphs, so as input to your project, you will be given a file describing the graph with the following format:

Node <X>
<connected node> <cost>
…
For example,

Node 1
2 4
3 2
Node 2
3 5

Is the graph:



Note that because edges are undirected, and the costs are bidirectional, the links are only specified once.

1. (10 points) Read in the graph configuration file.
2. (30 points) Implement the Dijkstra's algorithm (link state). Because this algorithm requires global knowledge, you can implement it just for one node. You should output the solution to a file. This file can be any format you like, but document it in your report.
3. (50 points) Implement the Bellman-Ford distance vector algorithm. Because this algorithm is distributed, you need to do the following:
    a. For each node in the graph, you need to spawn a separate process. (programmatically)

b. Each process needs to make a TCP connection to the neighbors specified in the graph file. For example, Node 1 would connect to 2 and 3, and Node 2 would connect to Node 3. Since TCP is bidirectional, only one connection is required.
c. Each process should start with *only* local information, meaning the costs to send to each neighbor node. It's OK to read in the whole file, but you can't use information from the file that isn't for your node.
d. You should create a protocol to exchange distance vector information, as required by Bellman-Ford.
e. Nodes should asynchronously exchange distance vector information as needed until the algorithm converges.
f. Since costs are fixed, you don't have to worry about poison reverse or anything other than the basic algorithm.
g. After convergence, each node should output their local DV table to a file (once file per node). Again this can have any format, but you should document this in your report.

4. (10 points) A report detailing what you implemented, how to build and run your code, any known bugs or limitations and any internet sources you used more than 10 lines of code from.

You must write this in Python.

This is an individual project so collaboration or sharing of code is not allowed. The University policy on plagiarism and collaboration will be strictly enforced. Limited internet references may be used (you can't submit a project you download from the internet), but any source you use more than 10 lines of code from should be documented in your report. Some examples of acceptable code snippets include: code for how to use TCP sockets in Python, code for how to make containers thread-safe, code for how to create threads, etc.

All students are expected to be able to explain their code. Students who can't explain their code will be subject to a penalty, up to 100% of the marks awarded.