Middle East Technical University

Department of Computer Engineering

CENG 435

Data Communications and Networking

Fall 2022–2023

THE - 3

Due date: 2022-12-22 23:59

1 Introduction

In this assignment, we are going to implement distance-vector routing protocol without dynamic cost changes for arbitary networks. The assignment will cover *routing algorithms*. You will use `Python` to implement it.

Before starting the implementation, please read Section 5.2 – *Routing Algorithms* from *Kurose & Ross* to understand the concept.

You can discuss the homework and ask your questions on the discussion forum thread on our ODTUClass page. I am also available at fyavuz@ceng.metu.edu.tr.

2 Implementation Details

For this assignment, you will implement a single `Node.py` that will be run separately for every node in the network. The nodes will then start communicating asynchronously using TCP sockets. When there are no more updates to be made in any of the distance vectors of the nodes, the algorithm will terminate.

2.1 Creating the nodes and initializing costs

Every node will start by reading it's assigned `<port>.costs` file from the disk to populate it's initial distance vector. If you read the Section 5.2 mentioned above, you know that the initial distances for nodes that are *not* immediate neighbours of the current node are set to infinite. Every node will then start listening to it's assigned port number on `localhost` (`127.0.0.1`). This port is used for receiving distance vector updates from the other neighbouring nodes.

2.2 Distance-Vector Routing

Every node will start by sending it's initial distance vector to it's immediate neighbouring nodes. Whenever a node x receives a distance vector, it will do the following check for every node in the *entire network*.

For instance, during node x's distance vector update for node y, distance to node y should be updated such as;

$$\texttt{dist}_x(y) = \min\{\texttt{dist}(x, v) + \texttt{dist}(v, y)\}$$

For every $v \in$ immediate neighbours of x.

If this update causes any change in the distance vector, that node should advertise it's new distance vector to it's immediate neighbours. This distributed and asynchronous communication should continue until there are no more updates on any of the nodes of the network. For this homework, we will settle on 5 seconds of inactivity as the indicator of the algorithm terminating. After the algorithm is done, every node should close their connections and print out the distance vector.

2.3 Stages

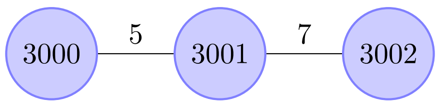
1. Read the neighborhood information from the `.costs` file
2. Send the node's distance vector to every immediate neighbor
3. Listen for updates from the neighbors
4. Attempt to update the distance vector by using received distance vectors
5. If no update happens for 5 seconds, close all connections and print the distance vector

3 I/O Specifications

distance vector should be printed in the following format;

```
<current node's port> -<neighbor's port> | distance
```

So, in an hypothetical network with 3 nodes in the following configuration;



The node at port 3000 will have and print the following distance vector at the end;

```
3000 -3000 | 0
3000 -3001 | 5
3000 -3002 | 12
```

3.1 .costs file format

A `<port>.costs` file has the following format, where the every line except the first are separated by a single space;

```
Total number of nodes in the network
Port of immediate neighbour Distance to that neighbour
Port of immediate neighbour Distance to that neighbour
...
```

For instance, given the network in the diagram above, the node on port `3001` has the following `3001.costs` file;

```
3
3000 5
3002 7
```

You will receive `n` different `<port>.costs` files for every test case.

4 Usage

Please download the test cases and the `runner.sh` from our ODTUClass page. Since we are spawning a new process for each node, it would get cumbersome without the runner script. `runner.sh`, on the other hand, handles that for you and allows you to pick test cases. During grading, we might introduce additional test cases.

5 Submission

This is an individual assignment. Discussing high level ideas regarding your implementation is encouraged. However, using implementation specific code that is not your own is strictly forbidden and constitutes as cheating. This includes but not limited to friends, previous homework, CENG homework repositories on GitHub, or the Internet in general. The violators will get no grade from this assignment and will be punished according to the department regulations.

Upload your `Node.py` (do not pick an alternative name) on the THE3 submission on our ODTUClass page.