**2251923O**

**Advanced Networks and Communications (H)**

**Part 1:**

**INFORMAL OVERVIEW**

The simulator which has been created involves 2 networks of nodes (nodes.csv and splithorizon.csv) An example of the normal convergence can be seen in nodes.csv whilst splithorizon.csv is an example of split horizon.

A menu is displayed showing the valid commands. Given the users input (exchange, trace all) the shortest path from each node to all other nodes is found, using the Bellman Ford algorithm as the distance-vector algorithm. The Bellman Ford algorithm only computes the graph if no negative weight cycles are found.

Text

Description automatically generatedIf the user’s input is not a valid command, they will be asked for another input. Links between nodes can be adjusted to increase or decrease the distance or can set to fail.

On request, split horizon can be enabled to overcome the count to infinity problem and combat slow convergence. The user is able to see the different paths with and without split horizon enabled.

**User Manual**

Before the program can be ran, it is essential that some requirements are met:

1. Python (2.7 or above) must be installed – instructions on how to do so is available at <https://www.python.org/downloads/>
2. Install pip – pip is pre-installed with Python 2.7.9 and higher
3. In the command line type:

pip install argparse, networkx

To run the program for normal convergence:

1. In the command line, type:

git clone <https://github.com/EveOHagan/ADN_Assignment.git>

1. In the command line, type:

cd ADN\_Assignment

1. When in the ADN\_Assignment folder, in the command line type:

python network.py -f nodes.csv

1. In the command line, to see the command options type:

Help

1. Adjusting or changing the network (add/remove nodes, or change distance between nodes) can be done through the nodes.csv file

To run the program for slow convergence:

1. In the command line, type:

git clone <https://github.com/EveOHagan/ADN_Assignment.git>

1. In the command line, type:

cd ADN\_Assignment

1. When in the ADN\_Assignment folder, in the command line type:

python network.py -f splithorizon.csv

1. In the command line, to see the command options type:

Help

1. To enable split horizon, in command line:

split-horizon

1. Adjusting or changing the network (add/remove nodes, or change distance between nodes) can be done through the splithorizon.csv file

**Part 4**

**Diagram

Description automatically generated with medium confidence**

***Routing Table showing shortest path:***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **N1** | **N2** | **N3** | **N4** | **N5** | **N6** | **N7** |
| **N1** | 0 | 1 | 4 | 8 | 7 | 9 | inf |
| **N2** | 1 | 0 | 3 | 7 | 6 | 8 | inf |
| **N3** | 4 | 3 | 0 | 4 | 7 | 9 | inf |
| **N4** | 8 | 7 | 4 | 0 | 3 | 5 | inf |
| **N5** | 7 | 6 | 7 | 3 | 0 | 2 | inf |
| **N6** | 9 | 8 | 9 | 5 | 2 | 0 | inf |
| **N7** | inf | inf | inf | inf | inf | inf | inf |

The table and graph above display the first network that is used. It consists of 7 Nodes (N1-N7) with the distance between each node shown in the graph.

The routing table shows the shortest route to each node.

**NORMAL CONVERGENCE**

Running the normal convergence can be done with instructions in part one.

Initially before any exchanges have been made, and the input is “trace all”, only the nodes which are directly connected to others show a distance in the routing table. This is because no exchanges have been made between nodes and their routing tables. To explain further, N1 is directly connected to N2 and N3 as seen above in the graph, so the distance between N1 and N2/N3 is printed. However, as N1 is not directly connected to N4, N5, N6, N7, the cost between the nodes appears as infinity as it is unknown.

Text

Description automatically generated

Once an exchange 3 has been made, and “trace all” is recalled, the nodes have exchanged routing tables which allows them to make connects to those nodes they are not adjacent too. The nodes can find the shortest path to the nodes which they are not directly connected to via other nodes, using the Bellman Ford algorithm. The Bellman Ford algorithms has the advantages of working with graphs which contain negative weights as well as works well with distributed systems. Due to the distance vector algorithm being dynamic, Bellman Ford was the most appropriate algorithm to use to find the shortest path.

Table

Description automatically generated

­­­­However, although N7 is present, the distance from all nodes to N7 will always be infinity despite how many exchanges are complete. This is due to N7 not being connected to any node, so the shortest path can never be found. Infinity represents connections that are unknown or unreachable.

A screenshot of a video game

Description automatically generated with medium confidence

The shortest route can also be found from one node to another, even when a route fails, it is redirected through other nodes. In the example below, it demonstrates how the shortest path between N1 and N2 is found (route N1 N2) From the graph above, it can be seen that N1 and N2 are directly connected. However, when the route between N1 and N2 fails, the shortest path is found using other nodes that are directly connected (via N3)

**Text

Description automatically generated**

**SLOW CONVERGENCE – splithorizon.csv**

**3**

N1

N2

**2**

N3

To run the network, the instructions are found in part 1.

The graph above displays the second network used.

When trace all is the input by the user, it displays the nodes which are directly connected to another:

* N1, N2 with a cost of 3
* N2, N3 with a cost of 2
* N1, N3 with a cost of infinity

As N1 and N3 are not connect directly, and no exchanges have been made, the distance between (N1 and N3) is unknown, so infinity is shown.

Text

Description automatically generated

When an exchange 1 occurs, the routing tables update. This allows for a connection to be made between nodes that are not directly connect, for example N1 with N3.

Text

Description automatically generated

* N1, N3 with a cost of 5 (via N2)

**WITHOUT SPLIT HORIZON:**

With the input cost N1 N2 fail it causes the link between N1 and N2 to fail. By doing so, N1 can no longer have a connection to N3 via N2.

Although N3 has not been notified of the failed cost between N1 and N2, so still is of the assumption that it can reach N1.

When the routing table is updated, the distance between N1 and N2 is infinity due to the link failing. However, all others remain as before due to the unawareness of the failed link.

Text

Description automatically generated

When an input of exchange 1 is entered, the routing tables updates with information gained from the other nodes.

Due to N3 being unaware that the link between N1 and N2 has failed, there is the assumption that:

* N1, N3 distance is 5 (via N2)
* N1, N2 distance of infinity (due to the link failing)

For Routing table N2, the cost of N2, N1 is 7. This is because due to the N2 direct link to N1 failing. N1 reminds N2, that it has a connection distance of 5 to N3 via N2 (N1 -> N3 = 5). In Routing tables, the node does not store the route which node went to get the shortest distance, instead just the shortest distance is stored. N2 also has a direct connection to N3 of distance 2. Thus, resulting in a distance of 7. (N1 -> N3 = 5) + (N2 -> N3 = 2)

However, as N1 is not aware that to get a connection too N3 it has to go via N2, which overlaps with the direct connection of N2 to N3. This is an example of slow convergence as eventually it will lead to a distance of infinity. (Count to infinity problem)

Text

Description automatically generated

**WITH SPLIT HORIZON**

Split horizon prevents the count to infinity from occurring, by allowing a node to only advertise its links between adjacent nodes (nodes they are directly connected to)

In the example above, N1 advertised to N2, that it had a link of 5 to N3. However, when split horizon is enable, this cannot occur.

When the N1, N2 link fails in split horizon, N1 can only advertise its connections of adjacent nodes. N1 and N3 are not adjacent nodes, as they are only connected through N2. N1 can only advertise its connection to N2 as it is its only direction connection (adjacent node). Instead, the cost between (N1 and N2) and (N1 and N3) is now infinity.

N2 and N3 are adjacent nodes, due to being directly connected, thus the cost of 2 remains.

Text

Description automatically generated

**LIMITATIONS**

One limitation of the simulator due to how split horizon is ran. When the program is ran without split horizon enabled, it works as intended. However, in order for split horizon to be enabled, the program has to stop running (exit program) and be re-ran to enable it. Although this would only add a number of seconds to the time the user would spend doing this, it is inconvenient. This would be the first improvement of the simulator.

Another limitation came from calculating the shortest path from one node to another. Although this work as intended, the shortest path was always computed as if an exchange between all nodes had been completed, even when it had not. An improvement on this would be, only to compute the shortest task with the knowledge the nodes have.

With the current method that the simulator takes, without split horizon the node distance will never reach infinity, due to the loops not operating to their full potential. This could be improved on to show the count to infinity problem.