Members
Benjamin Cheung
Michael McBreen
Quang Nguyen

Project 2 DV Routing Algorithm

November 18, 2015
CptS 455

Describe the routing problem

The routing problem is how to make sure that each router know the shortest path through the network. This is typically solved in one of two different approaches. The first is Dijkstra's algorithm which is run synchronously. The second approach is the Bellman-Ford or Distance vector algorithm.

Solution Approach:

Our solution to the routing problem is to utilize the distance vector routing algorithm on each router in our simulated network.

Each router in the simulator network is run as its own program. After being started with the command line arguments the router reads the router and config file to initialize. The initialization process involves creating connections with its neighbors and the starting DV table.

The routers then send out scheduled updates based on their routing table and if poison reverse was provided as an option. The router also read in updates from their neighbors which they then use to update their own DV table. This process continues with the end state being each router knowing the shortest paths.

In the following tables, the rows represent the destination while the columns represent the first hop router. The table entries represent the total cost to get from the current router to the destination router depending on the first hop router chosen.

Test1:

In test1, what happens when link cost BE changes to 2, the algorithm is allowed to converge, and then BE changes back to 8?

Test1 Case 1 Router A: Without Poison Reverse

Original Table:

```
from A :  via A via B via C via D via E
to    A : '      0"   13"  INF"  INF"    2'
to    B : '   INF"     7"  INF"  INF"    6'
to    C : '   INF"    8"   INF"  INF"    5'
to    D : '   INF"   10"  INF"  INF"    3'
to    E : '   INF"   12"  INF"  INF"    1'
```

Change to cost 2:

```
from A :  via A via B via C via D via E
to    A : '     0"    10"  INF"  INF"    2'
to    B : '  INF"     7"  INF"  INF"    3'
to    C : '  INF"     8"  INF"  INF"    4'
to    D : '  INF"    10"  INF"  INF"    3'
to    E : '  INF"     9"  INF"  INF"    1'
```

Table after we changed the cost back to 8:

```
from A :  via A via B via C via D via E
to    A : '     0"    13"  INF"  INF"    2'
to    B : '   INF"     7"  INF"  INF"    6'
to    C : '   INF"     8"  INF"  INF"    5'
to    D : '   INF"    10"  INF"  INF"    3'
to    E : '   INF"    12"  INF"  INF"    1'
```

Under normal condition after router B get an update with the cost of 2 to E, it updates its routing table for the most optimal path. Then it convert back to it original routing table when the cost get change back to 8. Shown in the tables above.

Test1 Case 2 Router A: With Poison Reverse

Is the behavior different if the -p flag is used (poisoned reverse)?

Original Table:

```
from A :  via A via B via C via D via E
to    A : '     0"    13"  INF"  INF"    2'
to    B : '  INF"     7"  INF"  INF"    6'
to    C : '  INF"     8"  INF"  INF"    5'
to    D : '  INF"    10"  INF"  INF"    3'
to    E : '  INF"    12"  INF"  INF"    1'
```

Changed to 2 -p condition:

```
from A :  via A via B via C via D via E
to    A : '     0"    10"  INF"  INF"     2'
to    B : '  INF"     7"  INF"  INF"     3'
to    C : '  INF"     8"  INF"  INF"     4'
to    D : '  INF"    10"  INF"  INF"     3'
to    E : '  INF"     9"  INF"  INF"     1'
```

Table after we changed the cost back to 8:

```
from A :  via A via B via C via D via E
to    A : '     0"    13"  INF"  INF"    2'
to    B : '  INF"     7"  INF"  INF"    6'
to    C : '  INF"     8"  INF"  INF"    5'
to    D : '  INF"    10"  INF"  INF"     3'
to    E : '  INF"    12"  INF"  INF"     1'
```

In both cases the table stabilize at the original values after the link cost change was reverted. From the perspective of router A there is no difference whether poison reverse is used or not. The largest difference we noticed is the time it takes to stabilize after a change. This is inline with the behavior of poison reverse.

Addition experiments:
Included in the OUTPUT.txt are the DV and Routing Tables for Test2. Since the tables would be the same with or without the -p flag, only the run output for Test2 with poison reverse is shown. The result of the Test2 output has no discrepancies.

Decisions:
We initially based our Distance Vector Algorithm on its wikipedia entry for distance vector routing. The largest decision we made about implementing the routing algorithm was assuming that no routers could be added or removed from the network. We assume that each U message we get will only describe routers that we already have in our table. If routers could be dynamically added or removed we would have to handle this behavior.

Discrepancies:
We were unable to find any discrepancies. However given the nature of software and especially networking software this is no guarantee that there are no discrepancies from expected behavior.

Conclusion:
In some respect, this program can be considered as a real Internet router because it has a routing table which contain the most optimal cost to send to it destination. The update interval also makes it more realistic as things on the Internet get change constantly and we can't rely on one link to work forever. If we would to rely on one link for all then every time when a link we're connected to get changed, it would destroy the whole network. However, in some aspect, the program doesn't represent the Internet because the host we are using a local host. Therefore, we won't experience any packet loss or latency issue as a real Internet router would. Thus, it limit our understanding of a real network functionalities. Also, the program is implemented using a buffer where a real Internet router would have something such as a crossbar matrix implemented with much higher throughput .

Hours for team members:

Benjamin Cheung	: 19	hours individual | 11 hours group
Michael McBreen	: 3	hours individual | 10 hours group
Quang Nguyen	: 8.5	hours individual | 6 hours group