REPORT

(Computer Networks)

Distance Vectors using Bellman-Ford Algorithm

7/4/2021

A **distance-vector routing protocol** in data Networks determines the best route for data packets based on distance. Distance-vector routing protocols measure the distance by the number of routers a packet has to pass, one router counts as one hop. Some distance-vector protocols also take into account network Latency and other factors that influence traffic on a given route. To determine the best route across a network, routers, on which a distance-vector protocol is implemented, exchange information with one another, usually routing tables plus hop counts for destination networks and possibly other traffic information. Distance-vector routing protocols also require that a router informs its neighbours of network Topology changes periodically.

Distance-vector routing protocols use the Bellman-Ford Algorithm to calculate the best route. Another way of calculating the best route across a network is based on link cost, and is implemented through link-state Routing Protocols.

Work-Around Classes:

Class Buffer used for storing tables and stuff

```
class Buffer():
```

Network Class to form a network (forming neighbour lists)

```
class Network():
```

Router Class to initializing a router and DVs

```
class Router():
```

Threading:

Function takes network, buffer and router as parameter, and run a router as thread.

```
def thread_target(network, buffer, r):
    for i in range(4):

        with LOCK:
            print(f"Itreation : {i + 1}")
            r.Show()
            r.initialize_mod()

        DV_2_Neighbour(network, buffer, r)

        # sleep thread for 2 sec
        time.sleep(2)

        # proceed only when all neighbours received
        while buffer.all_neighbours_received(r) == False:
            pass

        get_tables_from_buffer(buffer, r)
```

Threads runs independently.

Below functions execute Bellman-Ford Algorithm and updates Distance Vector Lists.

```
def BellmanFord(router, dv_list):
    num_routers = len(router.dv)

    for i in range(num_routers):

        for x in dv_list:

            for r_dv in router.dv:

                if r_dv[0] == x[0]:
                    val = r_dv[1]

            val = val + x[1][i][1]
```

```
            if val < (router.dv[i][1]):
                router.dv[i][1] = val
                router.modified[i] = 1


def get_tables_from_buffer(buffer, router):
    with LOCK:
        for x in buffer.queue:
            if x[0] == router.name:

                dv_list = []
                values = len(x[1])
                for i in range(values):
                    dv_list.append(x[1].pop(0))

    BellmanFord(router, dv_list)
```

Output:

Run Command -> python dvr.py input.txt

```
Router Name - B
Distance Vector -

Dest      Cost
A         5
B         0
C         4
Itreation : 4

Router Name - A
Distance Vector -

Dest      Cost
A         0
B         5
C         9
Itreation : 4

Router Name - C
Distance Vector -

Dest      Cost
A         9
B         4
C         0
```

Image above shows last iteration for each router indicating Destination router and its DV Cost respectively.