CSE-589 MODERN NETOWKRING CONCEPTS PROGRAMING ASSIGNEMENT-2

(PROJECT)

KEYUR JOSHI

keyurjos@buffalo.edu

UB PERSON # - 36508572

1. The project implements a simple distance vector routing algorithm using the Bellman-Ford equation and UDP sockets.

2. The server/router program is started by passing topology file and timeout/update interval as command line input.

3. **./server –t <topology file> –i <timeout interval>**

4. The server when started reads the initial configuration from a topology file that contains no of servers in the network, the number of neighbors, a table consisting of server ID followed by IP address and port number. The others entries are the neighbors directly connected to the router (server) and their cost.

5. These data entries for servers are read from file and stored in an **array** of objects of **server class**. The number of elements in array is decided by number of servers. The **neighbor link's cost** is stored in a **separate array (neighbor table)**. The IP address is stored in string form locally in an array and the corresponding IP address is stores as 32-bit unsigned integer. All others are 16-bit (unsigned short). This is the initial routing table.

```
 class server
{
 public:
 uint32_t IP;
 short int port;
 short int id;
 short int cost;
 short int disabled;
 }; //readfile.h [5-13]
```

6. The router the sends this table to its neighbors. The update has header indicating number of fields and id of sending node. This header followed by the routing table (array of server objects) is copied onto a buffer using **'memcpy'** and this buffer is send across the network via UDP sockets.(Fig 1 [page 4]) /keyurjos_proj2.cpp [393]

7. The router then enters a listening mode where it periodically sends updates to neighbors as well as listen to updates from neighbors and accepts user commands. In case of a command input it reads the input data and performs the corresponding action and returns a status message. If a routing update was received it checks if the message belongs to a

neighbor in its table and if the node is disabled if true then it ignores else it copies the cost from table into cost matrix and runs the DV algorithm

8. The DV is implemented using the Bellman-Ford equation

$$D_x(y) = \min_v \{c(x, v) + D_v(y)\}$$

9. The user can modify the link cost which will write that value to neighbor table and recalculate paths and send updates. Similarly a link can be disabled meaning server will no longer send update to that neighbor and ignore updates from it. This is done by setting the **'disabled =1' .**The **disable flag is used locally** and the receiver simple ignore this flag in the update it receives.

10. The routing table is displayed using the server class members and the corresponding IP address – print()[421]

```
Cout<<hst[k].id<<"\t"<<IPad[k]<<"\t"<<hst[k].port<<"\t"<<hst[k].cost<<"\t"<<nexthop[k];
//keyurjos_proj2.cpp
```

11. If a server does not receive updates in 3 timeout intervals it sets the neighbor cost to infinity. Depending on topology it can access the router through alternate path.

12. A node once disabled /crashed cannot be enabled again during the run-time of the program. The crash command causes all open file descriptors to be closed, exit listening state and wait for exit command.

Fig 1.Routing Message

*Disabled flag is Ignored by receiver and not processed

Number of Fields	Server ID
IP Address Server 1	
Server 1 Port	Server Id 1
Server 1 Cost	Server Disabled(*)
IP Address Server 2	
Server 2 Port	Server Id 2
Server 2 Cost	Server Disabled(*)
IP Address Server 3	
Server 3 Port	Server Id 2
Server 3 Cost	
IP Address Server 4	
Server 4 Port	Server Id 4
Server 4 Cost	Server Disabled(*)
IP Address Server 5	
Server 5 Port	Server Id 2
Server 5 Cost	Server Disabled(*)