CSE 589 PROJECT 2

REPORT

ROUTING PROTOCOL

HEMA MADHAV JAKKAMPUDI
UB ID: 50206563

CODE IMPLEMENTATION:

Entire program has been written in two files .c and .h files

".h" Files are regarded as header files in C which contains declaration of structures and .c files in the application contains main code which keeps the program running.

By executing the "make file" creates the UNIX executable server.

In the program the **MAX_INT** parameter which is considered **"infinity"** in the program. The value for MAX_INT is considered to be around 32600.This value is considered for the convenience and for the ease in computations.

A) Structures Inside "commands.h" Header file.

There three main structures that keeps the program running.

1) (Line Number 12)

```
typedef struct {
  int16_t id;
  int32_t IP;
  int16_t port;
  int16_t cost;
  int isNeighbour;
  int16_t updatecount
  int isDisabled;
 } NODE;
```

2) (Line Number 23)

```
typedef struct {
  int16_t id;
  int16_t cost;
  int16_t nexthop;
 }routing_table;
```

```
3) typedef struct{          (Line Number 30)
    char ip[32];
    int port;
    }ip_port;
```

Details about Structures

1) The first structure NODE gives the details about the node (Server) id, neighboring server cost, IP address and port details. These details are parsed and stored in the structure during the parsing of the topology files. Most of the fields are in either 16 bit or 32 bit data type to send routing updates in the form of proper bytes.

2) The second structure deals with routing table updates. The routing table is calculated using Bellman Ford Algorithm and the resultant routing vectors are stored in this structure.

3) The third structure is used to temporarily store node IP's that helps in determination of self id and port number.

B) Functions that keep the server Running.

a) Void sendUpdate() (Line Number 195)
   This server sends DV to the neighbors in the specified intervals.

b) void update_link(int16_t id,int16_t cost) (Line Number 225)
   This function handles the server link updates by sending the updated cost to the neighbours.
   Example: Update 1 2 9

c) void reset_routingtable() (Line Number 252)
   This function makes the routing table costs to "inf" that is MAX_INT before calculating the updated routing table from the received Distance Vectors.

d) void distance_vector() (Line Number 267)
   This function Calculates the routing table from the received distance vectors.This implements the Bellman Fold Algorithm.

e) int deconcatinate_msg(char* packet) (Line Number 305)
   This function parses the packet.ie; Convert the received packet of bits into readable form.
   This function also extracts the required fields form the packet.

f) int Index(int16_t neigh) (Line Number 406)

This function returns the index of neighbor in the structure when the ID of the server is given.

g) void get_ipaddr()   (Line Number 420)

This function returns the IP address and port of Ifaddress() ie; address of interfaces.

Beyond this main () is the C inbuilt function from where the program starts execution. This function contains logic to parse the topology and store the details in the appropriate structures.
Select() function is also implemented in the same method, by setting proper time out values.

I have used 2D array which stores the incoming DV's from the neighboring servers. These DV's are stored in the list w.r.t to their assigned server ID's. This is created at the beginning of the program.

REPORT

The distance Vector in the program is calculated using Bellman-Ford Algorithm. The following functions are core for the implementation of this algorithm.

1) **char* concatenate()** –This function concatenates given fields in the form of packet as prescribed in the project description.
2) **distance_vector()** – This function imports the received distance vectors from the neighboring servers and updates the routing table.
3) **disable_link()**- This function disables the particular neighbor-link, ie; By making the link cost as infinity.
4) **Update_link()** – Updates the link between two neighbors to the particular value.

When the program is started firstly initial routing tables are updated with the values given from the topology file. Servers exchange their Distance vectors using UDP connections between them. The link costs of all the servers will be updated by exchanging few packets among servers. At certain time, stability will be reached where there will be no change in the routing table. Servers has determined their shortest path to their neighbors. This happens with the series of exchanges between servers. This emulates real world routing in the routers.

Converges Slow

When the links are being updates it takes a while for the entire to stabilize. During this small intervals, the server may mislead its neighbors. During this state

1) This mislead cost of links will prorogate all through the system.
2) Routing tables of the severs will show the incorrect information until the system reaches stabilized state.
3) This also causes the packets to direct through the misleading paths, which sometimes causes timeout as the algorithms calculates wrong paths.
4) This will continue through out the system until the system stabilizes. The stabilization time is proportional to the cost that pushed the system into this state.

Infinity due to timeout

When the server has no communication with the particular server for the specific amount of time. (time=3*timeout intervals). The cost associated with this server is made to that corresponding neighboring server is made as “Inf”, i.e. The maximum value, MAX_INT, and next hop=-1. This is updated back again when the server determines alternative path to the destination. The path determination time may be determined by the distance hops to the reach the destination and availability of the neighboring nodes.

Count to Infinity Problem:

Count to infinity problem is caused when the one of the links in the server has grown large. It takes series of update packet to stabilize the network. It has caused due to crash in the link or disabling of the particular link. Server propagates this updated information down the network. This information is quickly propagated along the network in no time, since there will be condition where the server propagates error information to their neighbors, which eventually misleads the neighboring servers ending up routing packets through the misleading links. This generally happens as the servers only share cost to their neighbors but not the next hop information. Thus the program may end up in dilemma state, where it exchanges large number of packets in order to determine the shortest path to the destination.