PROJECT 2 : DISTANCE VECTOR SIMULATION

Name: Sarang Dev
Person#50170384

Description

The program runs in two modes :
1.DEBUG
2.NON DEBUG

To run the program in Debug mode, the debug definition macro must be uncommented. In debug mode, all the packets and distance vector is displayed in console after receive of each packet.

//#define DEBUG in Constants.h

To compile the program, use make. make in the program's root directory will make the executable named server.

To run the program :

./server -i 30 -t topology.txt
where i takes time interval and t takes topology file name.
Also the order of t and i doesn't matter.

Distance Vector :

The algorithm is maintains a routing table which contains the next hop and the value of the shortest path and a Distance Vector containing the Distance vector of all the other servers. The structure of DV is :

For Server 1

From Server 1	Via server 1	Via Server 2	Via Server 3	Via Server 4	Via Server 5
To Server 1	NA	NA	NA	NA	NA
To Server 2	NA	2	3	6	4
To Server 3	NA	5	7	8	9
To Server 4	NA	7	8	8	2
To Server 5	NA	8	5	4	3

Here distanceVector[i][j] means path to i via j.

The shortest of these ie For server 2 it will be the shortest j in disatnceVector[2][j].
It is defined in ServerRouter.h as
std::vector<std::vector<unsigned short> > distanceVector.
ServerRouter.h:45

Routing table :

It is the structure which maintains the shortest route to a particular server. It contains the next hop as well.
It is defined in ServerRouter.h as a map of a int and struct object. The key is the server id and value is the structure of the ServerInfo.
ServerRouter.h:29

```
typedef struct
{
   std::string servIp;
   unsigned short port;
   unsigned short nextId;
   std::string nextIp;
   unsigned short cost;
}ServerInfo;
std::map<unsigned short,ServerInfo> serverTable;
```

Packet Structure:

Packet strcuture is defined in RouterPacket.h as a struct with a dynamic array at the end.
RouterPacket.h :5

```
typedef struct
{

   unsigned int serverIp;
   unsigned short serverPort;
   unsigned short serverId;
   unsigned short linkCost;            //automatic padding of 2 bytes will be added
}ServerRouterInfo;

typedef struct
{
   unsigned short numFields;
   unsigned short serverPort;
   unsigned int serverIP;
   ServerRouterInfo List[];
}ServerRouterPacket;                   //PACKET
```

The size of List[] is defined at runtime with malloc based on the no. of servers in the topology file.
space allocation : ServerRouter.cpp : int ServerRouter::updatePacketInit()

Distributed Bellman Ford Algorithm :

It is implemented in ServerRouter.cpp : 280 - 318 in the funtion:
int ServerRouter::recvProcessUpdatePacket()

The algorithms calculates the cost to the servers according to the incoming packet. And it updates the distanceVector[][] for itself.
Then the shortest route for a server is the minimum value correspoding to the row for that particular server in distanceVector.

UPDATE :
It handles all the conditions specified in the description and also checks on the neighbors.
It also sends immediate updates to the other server.

CRASH HANDLING :

Whenever there is a crash either by a CRASH command or forced exit from any server program. The implementation removes it from the list of neighbors.
But the entry remains in the distanceVetcor and Routing Table. Now as whenever there is a crash, the Algoritm goes for COUNT TO INFINITY.
But the infinity has been limited to a certain cost in Constants.h:

const int MAX_COST=30; // please change this value according to topology

It should be set according to the topology. Based on the topology we can decide on the max hops that a packet can travel and get the max cost as max_hop*average_path_cost. Please adjust it according to the topology.
As the cost to the crashed server reaches this value, it is assigned as infinity in the Routing table and distanceVector.