Report on CSE 4/589 Fall-2015
Programming Assignment 2
Routing Protocols

<u>Prepared By-</u>

Anuj Rastogi

anujrast@buffalo.edu

UBPerson# 50134324

DATE: 20 November 2015

INTRODUCTION

This report is an attempt to describe the implementation of the program and the commands in the program that do not work. The report describes an overview of the working commands and the minor faults in the program that should be avoided for successful execution of the application. Along with the report the module also contains a README file which describes the file in which the commands and functionality is present along with what will work and what will not.

The description of implementation is as below:

Implementation of Start-Up command

The program takes the start-up command in the following format:

server -t ***<topology-file-name>*** **-i** ***<routing-update-interval>***

- If this command is not correct the program displays the correct command and asks the user to start the program again.
- This command is always passed as COMMAND LINE ARGUMENTS.
- If the file name is incorrect then the program pings the user with appropriate message and stops execution.
- Only when the file is present and the command format is right is the program executed.
- There is a function in "helperFunction.c" that validates the command when it is entered. The function is: "**int validateStartCommand(char *tempStart)**"(Line numbers from 268 to 324)

Implementation of File reading

Based on the file supplied at the time of execution the program reads the file. The file read function works on the following logic:

- The program reads the file with the function fgetc.
- As soon as it encounters '\n' it recognizes the end of line and hence, puts the space between the last word of the first line and starting word of the 2nd line.
- While reading program keeps on storing the characters in an array and hence after complete execution it has an array of characters which can then be used as a string.

- Once a string is formed, a "strtok" function is used to split the string at spaces and the words returned thus are stored in an array of character pointers.
- Thus, finally we have an array of strings which can be used now to fill different tables.
- This is implemented through two functions "**char *FileString(char *FilePath)**" and "**char **ArrayStrings(char *strTemp)**". (Lines 329 to 362 and Lines 216 to 262 in "helperFunctions.c")

NOTE OF CAUTION** the file format should be absolutely correct without unnecessary spaces.

Implementation of how data is stored in table post file reading

Once we read the file we have an array of strings. With the help of this array we can assign different array components to different table attributes based on the matchup.

Implementation of routing table and Information table

There are two tables that are maintained in the program. These tables are maintained in the form of "**struct**" namely "**NetworkInfo**" and **"RoutingTable".**

- For NetworkInfo struct we have an array of NetworkInfo struct. This struct stores the details of the servers present in the network.

```
struct NetworkInfo
{
   int ServerIDs;
   char IP[100];
   char port[20];
};
```

- For RoutingTable we have an array of struct RoutingTable which maintains the details of routing information for that server. This looks like:

```
struct RoutingTable
{
   int SourceId;            //Id of the destination server
   int NId;                        //Id of the destination
   char NIP[100];                  //IP of the neighbour
   char NPort[20];                 //Port of the neighbour
   int IntermediateNode;  //Next hop ID
   int IsNeighbour;                //Flag to identify if the NId is the neighbour
```

```
    int cost;                         // stores the cost from SourceId to NId
    int IsDisabled;                   //Flag for link status between SourceId and NId
    int Existed;             //Flag to if the Destination ever existed as NId
};
```

The implementation for this is present in the "anujrast_proj2.c" file. Lines 3 to 21.

NOTE**: The tables are filled initially by the File that is read.

Implementation of Update Message

The update message has a format that was mentioned. Since the message over the internet travel in the form of binary string so, there was a task to assure that all message follow the format or the ordering of the information in the same way.

Hence the string created to be sent has the following order:

Number of update fields+ Server port+ Server IP+ Server IP address n+ Server port n+ Server ID n+ Cost n

Message for update is created with the help of the information stored in routing table. The function to create the message to be sent is present in "anujrast_proj2.c" file. The line numbers are from 1206 to 1385 in "anujrast_proj2.c" file.

Implementation of step command

The step command calls the message creation function to create a message and sends the message to the neighbours or the servers for which the IsNeighbour Information in RoutingTable is equal to 1. Code implementation is from line 752 to 784 in file "anujrast_proj2.c" file.

Implementation of Update command

ONLY update<server1><server2><cost> and NOT update <server1><server2><inf>

The update command first checks the IDs of the two machines entered by the user and judges them on several filters. Once these filters are cleared the update is done on the machine which wants the update and also the other machine which is involved in update (code present in lines 433 to 567 in file anujrast_proj2.c).

NOTE** the **update for "inf"** is not implemented, although the code for the same is present. Due to the occurrence of some issues at runtime, the code for the same is commented to ensure proper execution of program. (Lines 570 to 702 in file anujrast_proj2.c)

Implementation of Packets command

The counter in the listener socket is placed which counts the packets whenever something is received. When we execute the packets command it gives the count of packets and makes the count to zero so that it can restart the counter.

Code in lines 788 to 793 in file anujrast_proj2.c.

Implementation of Display command

The display command displays the routing table in the increasing order of NId (Destination ID). It just sorts the routing table based on the NId.

Code in lines 705 to 748 in file anujrast_proj2.c

Implementation of Bellman Ford Algorithm

The bellman ford algorithm is implemented in the message receiver socket with the following logic.

- If the given cost is less than what is already stored in the table it stores that entry and the next hop ID as the one which is making that update.

NOTE** the logic for the bellman ford is present in the lines 924 to 1127 in file anujrast_proj2.c.

There is logic also for handling infinity and disable but since the commands are commented so that logic is not being used.

WHAT TO DO TO AVOID ERRORs

- **Post running the program do not press enter key if there is nothing typed in the screen. The program falters and gives segmentation fault.**

WHAT IS NOT IMPLEMENTED

- **The following commands are not implemented but the code has been done. Due to instability of the commands they are commented in the file.**

- **Update <server1><server2>inf** (Lines 570 to 702 in file anujrast_proj2.c)
- **Disable <ServerId>** (Lines 796 to 870in file anujrast_proj2.c)
- **Crash**