CS542

Link State Routing Protocol

Submitted by

Nithin Rajeev
CWID – A20346359

Introduction

A protocol which determines the route in which a packet travels through the network from a source to a destination is called a routing protocol. Routing algorithms decide which route a packet should travel through, the cost incurred during the propagation and considers various factors like traffic congestion, shortest path and so on before choosing the route the packet has to take. Each router only knows the information of routers that are attached to it and nothing more. Routing protocols make use of the information each router has and makes use of it to decide on a route. They gain knowledge of other routers information and propagates it through the network.

Link State Routing Protocol

In Link State Routing Protocol each node makes use of a graph which is created with the routing information available to it. The graph has information of what all nodes each router has connectivity to. Once this information is available each node has the ability to calculate the distance to the destination and can choose the best available path for communication. Using this information a tree can be made which will have the shortest path to the destination it seeks to send the packet to. Each node as mentioned above only has partial knowledge, that is, it only has the information of the nodes directly connected to. By making use of this partial knowledge of each and every node we can evaluate the complete topology of the network.

So for the link state algorithm to work the first task that needs to be done is to create Link State Packets or LSP which is basically the state of each link for a node. This will have all the information of the nodes that are connected to it and the cost. Now that we have the information and state of each node we need to share this information with every other node in the network. This is done by flooding which is basically sending this information to every other node in the network. Once every node has the information each node will be able to calculate the shortest tree path for itself. Once the shortest path tree is calculated each node can decide for itself how to best route a packet that comes to it. LSP's need to be created periodically so that any change in the topology is reflected to every node. An example is when a node is down. This information needs to be available to every other node so that they can route the packets accordingly. This would prevent data loss. Every time a node receives a LSP it compares the information that it already has with the new information it has received. If there is no change in the information that it already has the LSP is just dropped and no new change needs to be done to the shortest path tree that is already generated.

Once each node receives all the information through the LSP it has to create the shortest path tree. The algorithm that is used to create the shortest path tree is the Dijkstra Algorithm. The Dijkstra algorithm actually calculates the distance of a node to every other node in the network and uses that to create the shortest path. In short the shortest path tree is a tree that has each node connected to every other node in the shortest distance possible.

Dijkstras Algorithm

Let us take an initial node A and another node B. The algorithm will initially assign a distance value and try to improve upon that distance value till it gets the shortest distance from A to B.

1. Initially a distance of infinity is set to each and every other node and the distance of the node to itself is set as zero rightfully.
2. Now we will start from our initial node. Let us take this initial node to be current. At this point we have not visited any other node and hence we will be marking every other node as unvisited.
3. At this point we will we will visit every neighbor of the current node and calculate the distances to them. The newly calculated value that we have is compared to the earlier value that was assigned. In this case we have assigned everything to infinity. So the newly calculated value is compared to the earlier value and whichever is shorter is updated.
4. Once we are done calculating the distances to every neighbor of the current node and updating the distances we mark the current node as visited and remove it from the unvisited node list. Once visited it will not be visited again.
5. If the destination node has been visited or if the shortest distance between the nodes in the unvisited set is infinity then we stop.
6. If not we visit the node in the unvisited node list which has the shortest distance and marks it as the current node and repeats from step 3.

The above steps explains the working of the Dijkstra’s algorithm in the best possible way.

Project Description

The objective of this project is to develop a simulator to find the shortest path in a network. The Dijkstra's algorithm needs to be used to calculate the shortest distance. The project is coded in C on the MAC OS X platform. The simulator accepts a text file as input, which contains a topology matrix.

Design

The projects gives the user a menu driven prompt which has all the functionalities that are listed below.

1. Create a Network Topology
2. Build a connection Table
3. Shortest Path to Destination Router
4. Modify network topology
5. Exit

Implementation

Option 1

1. Gets input form the user in the form of a text file.
2. Parses the text file and reads the matrix given as the input and creates the network topology.
3. Displays the network topology.

Option 2

1. Gets the source router from the user and checks if the source is valid.
2. Creates the connection table and displays the connection table.

Option 3

1. This option takes a source and destination from the user.
2. The source and destination are checked to see if they are valid.
3. Once the validity is confirmed then the shortest path between the two are calculated and the cost is displayed.
4. The path from the source to the destination is displayed.

Option 4

1. Accepts a router from the user.

2. The router that is entered is removed from the network topology deeming it unreachable or making it down.
3. The new topology matrix is created and displayed.

Option 5

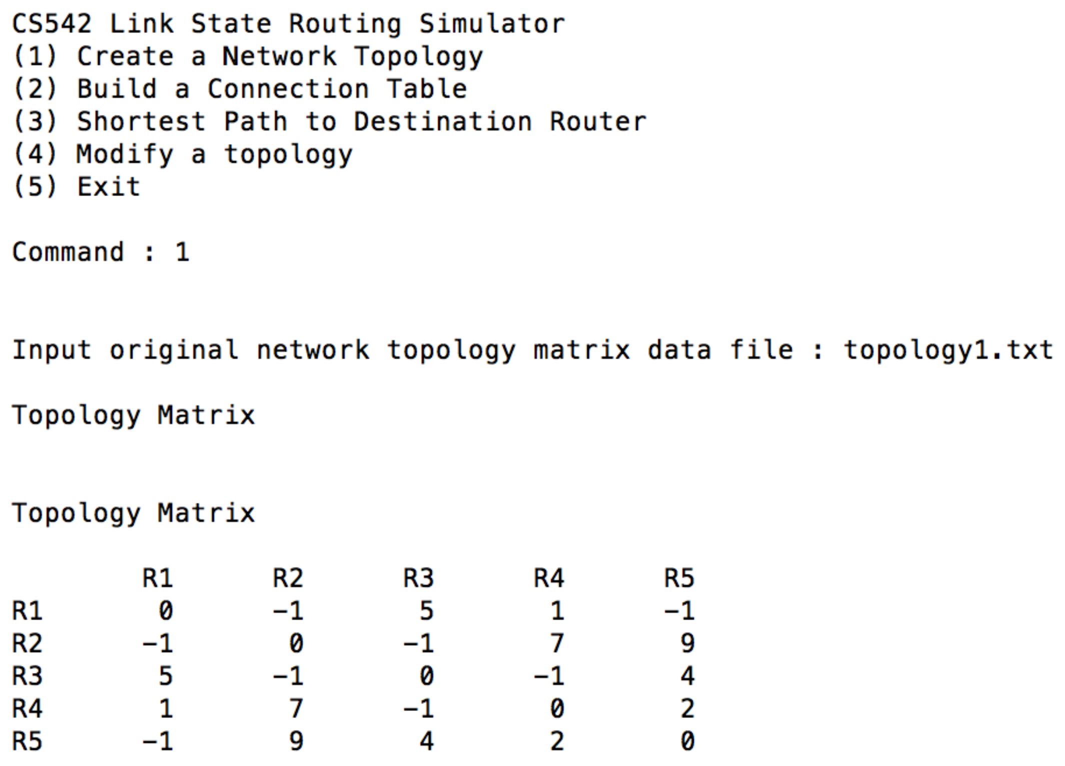
1. Exit from the program.

Pseudocode for Dijkstra’s Algorithm

```
1.  Initialization:
2.  N' = {u} (permanent node)
3.  for all nodes v
4.  if v adjacent to u
5.  then D(v) = c(u,v)
6.  else D(v) = ∞
7.  Loop
8.  find w not in N' such that D(w) is a minimum
9.  add w to N'
10. update D(v) for all v adjacent to w and not in N' :
11. D(v) = min( D(v), D(w) + c(w,v) )
12. /* new cost to v is either old cost to v or known
13. shortest path cost to w plus cost from w to v */
14. until all nodes in N'
```

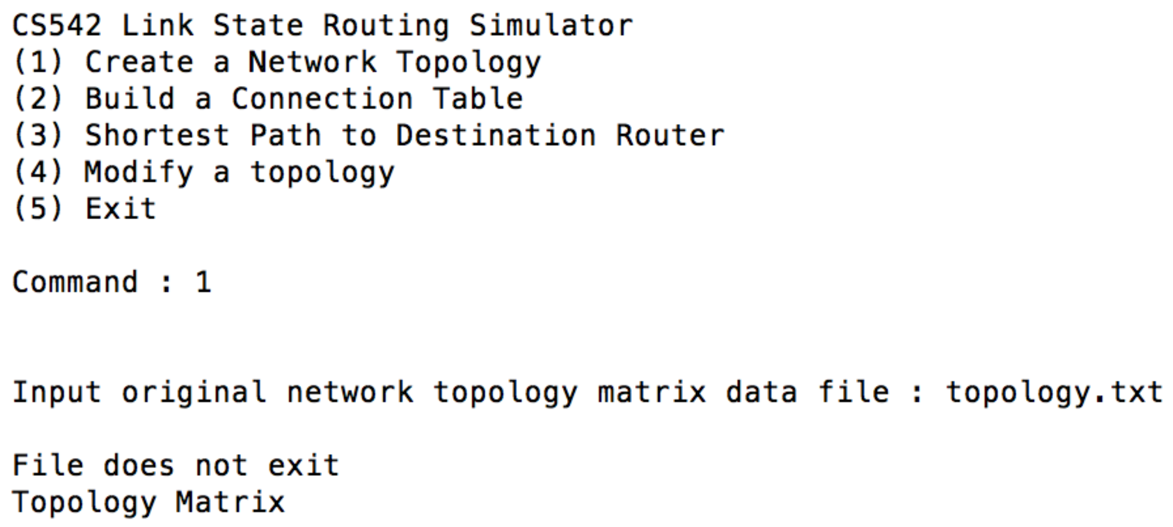
Test Cases

1. Create a network topology with a 5 X 5 matrix



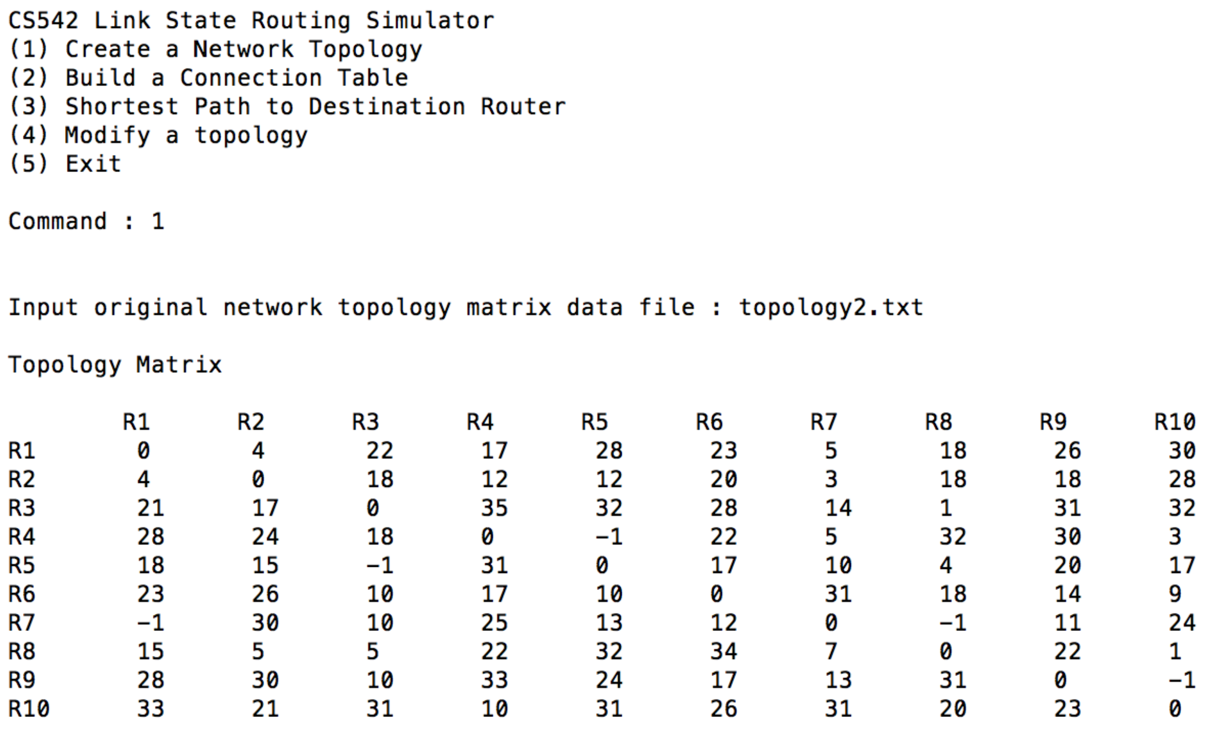
```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 1


Input original network topology matrix data file : topology1.txt

Topology Matrix


Topology Matrix

        R1      R2      R3      R4      R5
R1       0      -1       5       1      -1
R2      -1       0      -1       7       9
R3       5      -1       0      -1       4
R4       1       7      -1       0       2
R5      -1       9       4       2       0
```

2. Create a network topology with a non existing input file



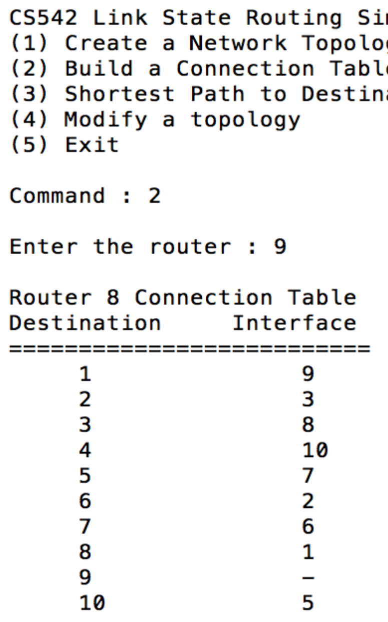
```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 1


Input original network topology matrix data file : topology.txt

File does not exit
Topology Matrix
```

3. Create a network topology with a 10 X 10 matrix

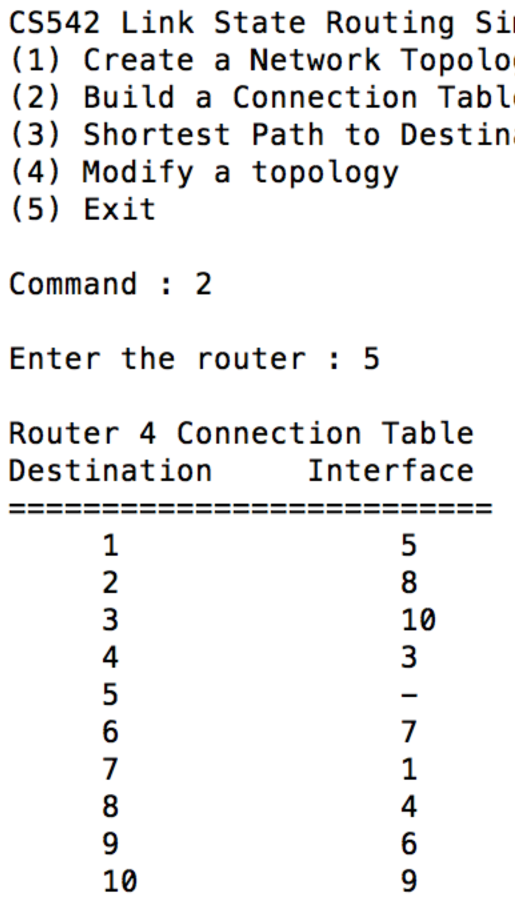


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 1


Input original network topology matrix data file : topology2.txt

Topology Matrix

        R1      R2      R3      R4      R5      R6      R7      R8      R9      R10
R1      0       4       22      17      28      23      5       18      26      30
R2      4       0       18      12      12      20      3       18      18      28
R3      21      17      0       35      32      28      14      1       31      32
R4      28      24      18      0       -1      22      5       32      30      3
R5      18      15      -1      31      0       17      10      4       20      17
R6      23      26      10      17      10      0       31      18      14      9
R7      -1      30      10      25      13      12      0       -1      11      24
R8      15      5       5       22      32      34      7       0       22      1
R9      28      30      10      33      24      17      13      31      0       -1
R10     33      21      31      10      31      26      31      20      23      0
```

4. Display connection tables

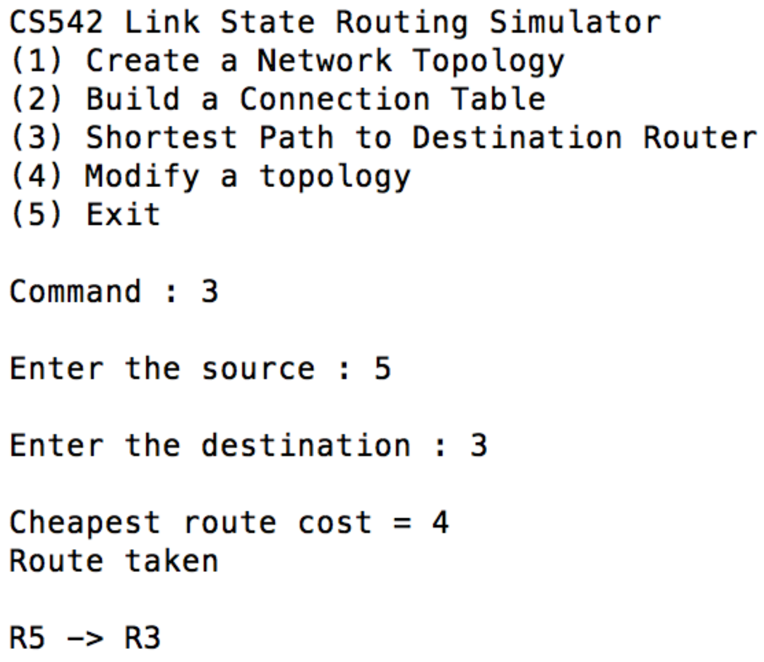


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 2

Enter the router : 9

Router 8 Connection Table
Destination      Interface
==========================
     1               9
     2               3
     3               8
     4               10
     5               7
     6               2
     7               6
     8               1
     9               -
     10              5
```



```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 2

Enter the router : 5

Router 4 Connection Table
Destination     Interface
==========================
     1               5
     2               8
     3               10
     4               3
     5               -
     6               7
     7               1
     8               4
     9               6
     10              9
```

5. Shortest Path to destination router

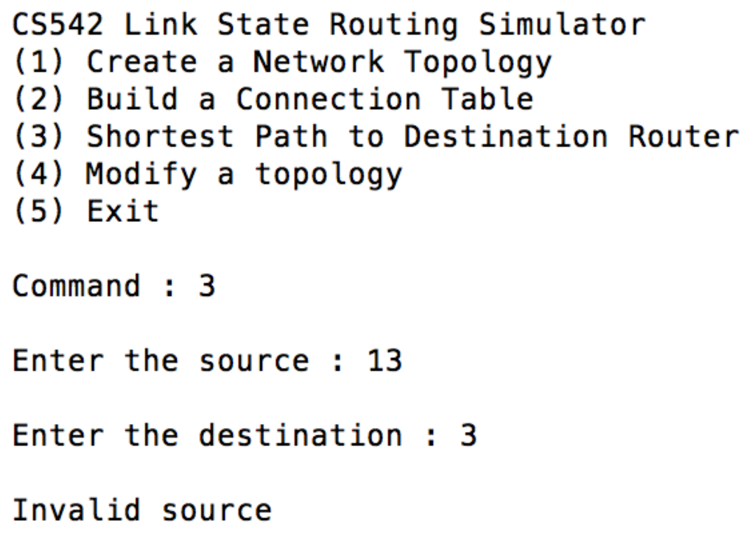


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 5

Enter the destination : 3

Cheapest route cost = 4
Route taken

R5 -> R3
```

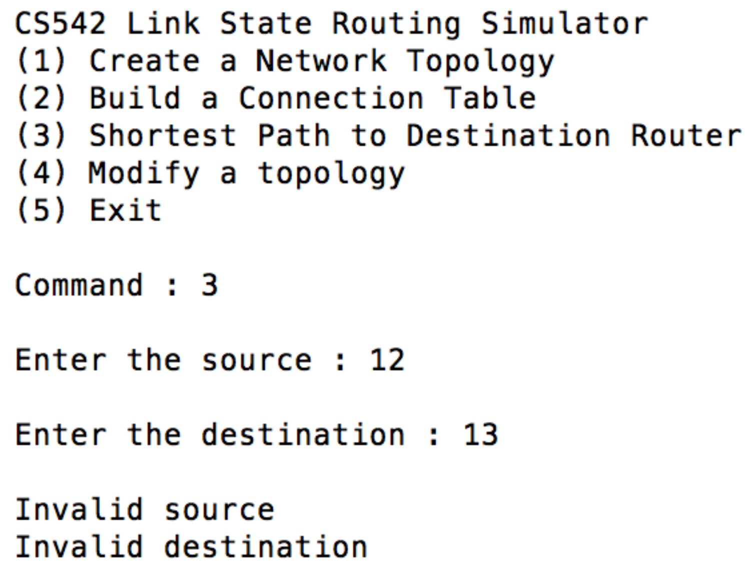
```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 3

Enter the destination : 4

Cheapest route cost = 6
Route taken

R3 -> R5 -> R4
```

```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 1

Enter the destination : 5

Cheapest route cost = 3
Route taken

R1 -> R4 -> R5
```

6. Invalid source and destination routers

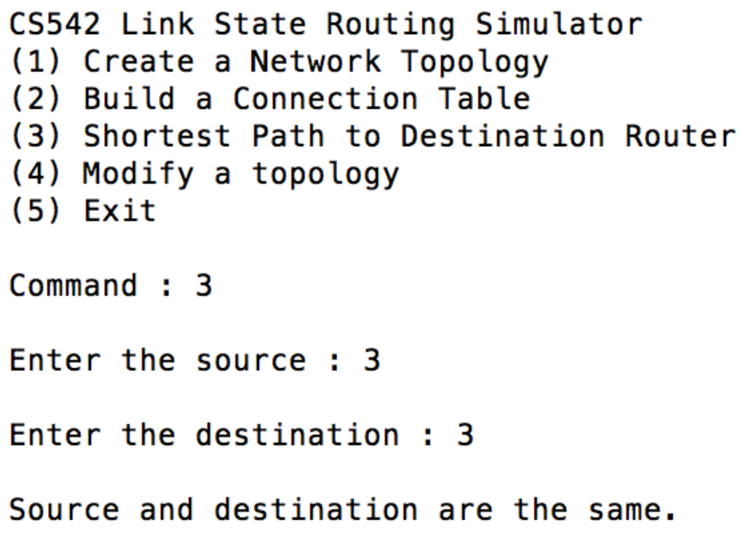


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 13

Enter the destination : 3

Invalid source
```



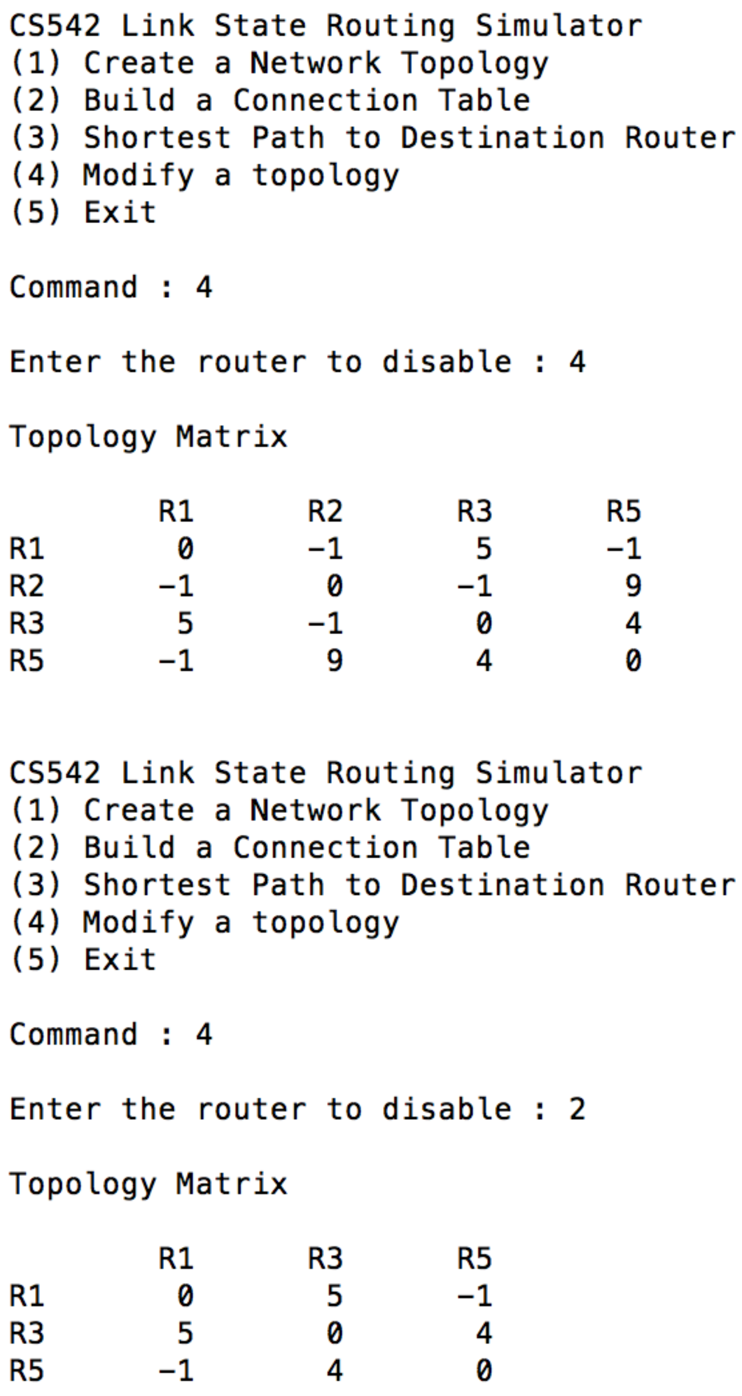
```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 12

Enter the destination : 13

Invalid source
Invalid destination
```

7. Same source and destination



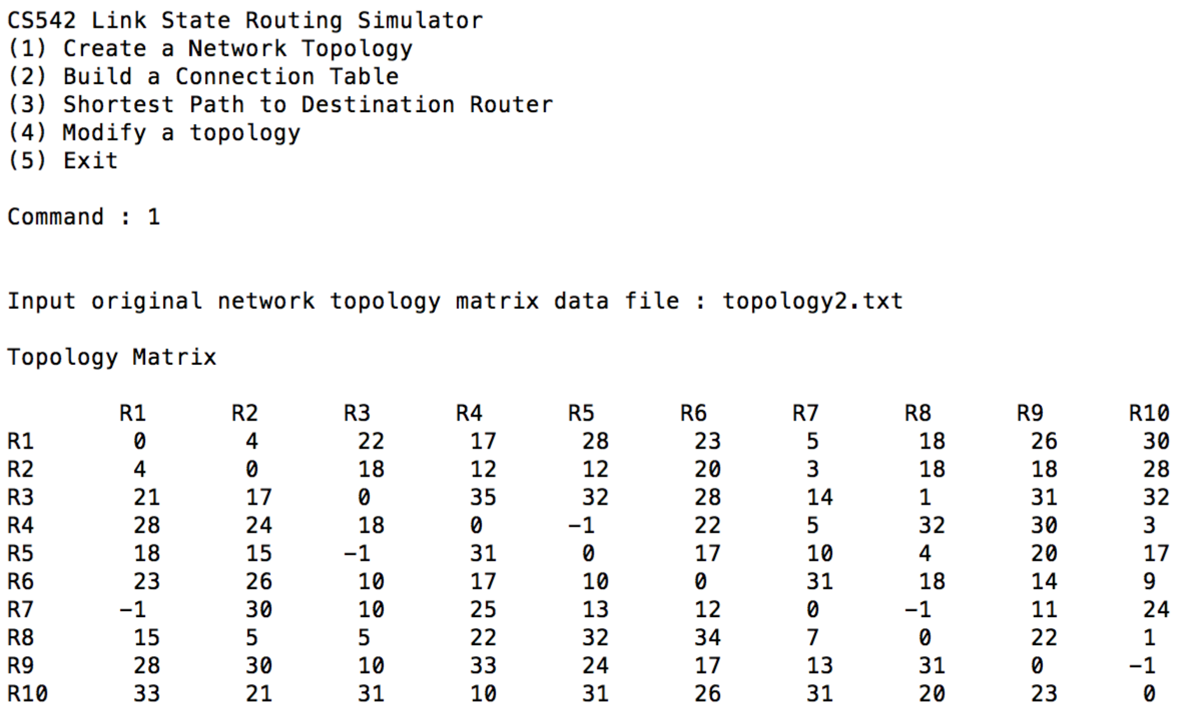
```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 3

Enter the source : 3

Enter the destination : 3

Source and destination are the same.
```

8. Modifying a network topology

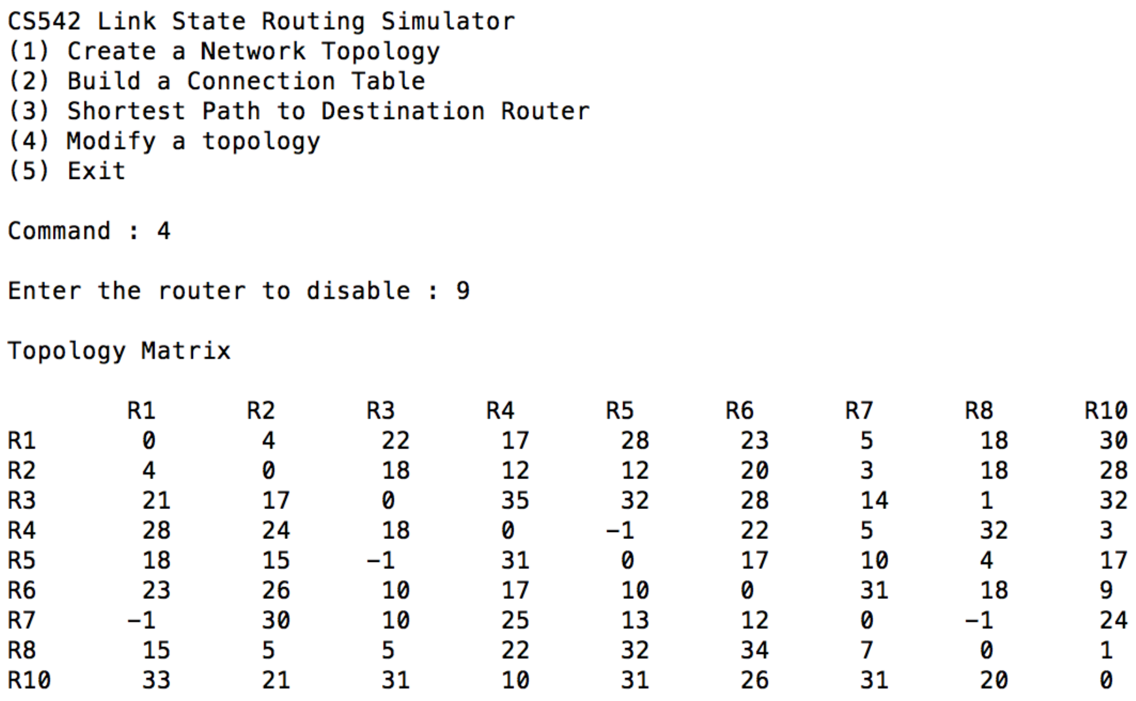


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 4

Enter the router to disable : 4

Topology Matrix

        R1      R2      R3      R5
R1       0      -1       5      -1
R2      -1       0      -1       9
R3       5      -1       0       4
R5      -1       9       4       0


CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 4

Enter the router to disable : 2

Topology Matrix

        R1      R3      R5
R1       0       5      -1
R3       5       0       4
R5      -1       4       0
```

9. Modifying network topologies for larger topologies.

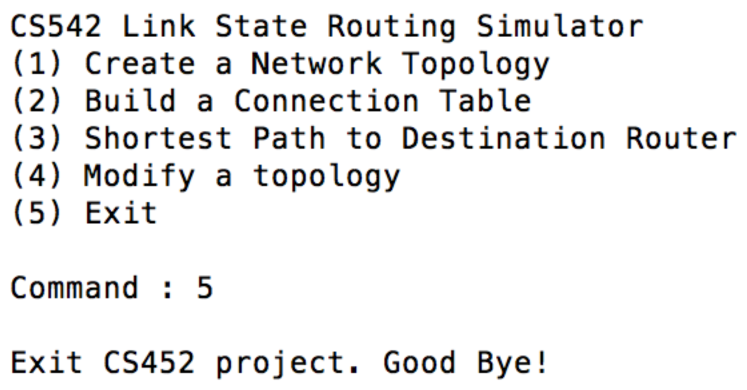


```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 1


Input original network topology matrix data file : topology2.txt

Topology Matrix

              R1        R2        R3        R4        R5        R6        R7        R8        R9        R10
R1            0         4          22        17        28        23        5         18        26        30
R2            4         0          18        12        12        20        3         18        18        28
R3            21        17        0         35        32        28        14        1         31        32
R4            28        24        18        0         -1        22        5         32        30        3
R5            18        15        -1        31        0         17        10        4         20        17
R6            23        26        10        17        10        0         31        18        14        9
R7            -1        30        10        25        13        12        0         -1        11        24
R8            15        5         5         22        32        34        7         0         22        1
R9            28        30        10        33        24        17        13        31        0         -1
R10           33        21        31        10        31        26        31        20        23        0
```



```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 4

Enter the router to disable : 9

Topology Matrix

          R1        R2        R3        R4        R5        R6        R7        R8        R10
R1        0         4          22        17        28        23        5         18        30
R2        4         0          18        12        12        20        3         18        28
R3        21        17        0         35        32        28        14        1         32
R4        28        24        18        0         -1        22        5         32        3
R5        18        15        -1        31        0         17        10        4         17
R6        23        26        10        17        10        0         31        18        9
R7        -1        30        10        25        13        12        0         -1        24
R8        15        5         5         22        32        34        7         0         1
R10       33        21        31        10        31        26        31        20        0
```

10. Exit function.



```
CS542 Link State Routing Simulator
(1) Create a Network Topology
(2) Build a Connection Table
(3) Shortest Path to Destination Router
(4) Modify a topology
(5) Exit

Command : 5

Exit CS452 project. Good Bye!
```

Compilation Instructions and execution

The code has been implemented on the MAC OS X and works efficiently on the MAC OS X. The code has been written in C and has been compiled with the gcc compiler.

Instructions to compile

1. Download the zip folder of the project.
2. Unzip the compressed folder.
3. Go to the src folder.
4. Compile using the following command.
    a. Gcc linkStateRoute.c
5. Run the a.out file using the following syntax.
    a. ./a.out
6. Alternatively you can execute the a.exe executable that can be found in the src folder.

Conclusion

The project has been successfully completed and can accept topologies of any size. The simulator was able to successfully calculate the shortest paths for all nodes despite the size of the topology and was able to successfully print the route from the source to the destination.