Finding Network Topology

Baradi Sandeep– A20352613

Bhagavatula Prasanth – A20355611

Mamidi Sudheer Durga Venkata– A20344048

Contents

Introduction

The aim of the project is to find-out the entire network topology for a router. The router will know only the information about the neighbouring nodes. Using this project, the router will be able to identify the entire topology of the network.

Problem Statement

The project needs to solve the below mentioned problems.

- Format of the packets exchanged between the routers.
- Choosing a metric to determine the progress of the network topology identification process.
- Computer program simulating the network topology update process.
- Identifying the termination criterion for the simulation process.

Format of the message

We use two different kinds of messages to find out the network topology, one "Request" message and one "Response" message. Below are the formats of the messages.

Request Message Format

The request message format is given below.

<table>
<tr><td colspan="3">Source IP Address</td></tr>
<tr><td colspan="3">Destination IP Address</td></tr>
<tr><td>Request
(1-bit)</td><td>Length
(7-bits)</td><td></td></tr>
</table>

Source IP Address

This is the IP address of the router which will request the destination router to send the information regarding its neighbouring routers.

Destination IP Address

This is the IP address of the router to which this request message is intended to. Once this message is received, it will create a response message which will have the information related to its neighbouring routers.

Request bit

This 1-bit field is set to '0' to indicate that this message is a request message.

Length

This 7-bit field indicates the total length of the message in bytes.

Response Message Format

The response message format is given below.

<table>
<tr><td colspan="8">Source IP Address</td></tr>
<tr><td colspan="8">Destination IP Address</td></tr>
<tr><td>Response
(1-bit)</td><td colspan="2">Length
(7-bits)</td><td colspan="2">Time to Live
(1 byte)</td><td colspan="3">Neighbouring router – 1
(2 bytes)</td></tr>
<tr><td colspan="4">Neighbouring router-1
(2 bytes)</td><td>Router role
(1 bit)</td><td>Cost of Link
(7 bits)</td><td colspan="2">Neighbouring router – 2
(1 byte)</td></tr>
<tr><td colspan="6">Neighbouring router – 2
(3 bytes)</td><td>Router role
(1 bit)</td><td>Cost of Link
(7 bits)</td></tr>
<tr><td colspan="8">…</td></tr>
<tr><td colspan="2">Neighbouring router – 20
(1 byte)</td><td>Router role
(1 bit)</td><td>Cost of Link
(7 bits)</td><td colspan="4"></td></tr>
</table>

Source IP Address

This is the IP address of the router who is sending this message containing the details of its neighbouring routers.

Destination IP Address

This is the IP address of the router who has sent the request message to send the information about the neighbouring routers.

Response bit

This 1-bit field is set to ‘1’ to indicate that this is a response message.

Length

This 7-bit field indicates the total length of the message in bytes. The maximum size of the response message can be 110 bytes (up to 20 router’s information can be sent in a single response message).

Time to live

This is a 1-byte field represents the life time of the message. Each time this message is processed at a router, if the request is not intended for this router, it will reduce the value of count in this field by 1 and forward the message. In case, this router knows the destination router, it will forward the message using that interface. Otherwise, the router will forward the packet on all of its outgoing interfaces. Once the value of this field becomes zero, the message is simply dropped.

Neighbouring Router Address

This field is 4-byte long and it represents the IP address of the neighbouring node to the sending router.

Router role

This 1 bit field is set to '0' if this neighbouring router sends information to this router (incoming link) and the field is set to '1' if this neighbouring router receives information from this router (outgoing link).

Cost of the Link

This 7-bit field is used to represent the cost of the link between the neighbouring router and this router.

Metric of the Network Topology Update process

The Network Topology is identified over a number of iterations; every iteration involves requesting new routers to provide information about their neighbouring nodes. In order to keep track of the progress of the update process, we need a metric and the metric chosen for this update process is the "Number of Routers that can be reached" in the topology.

Description of Update process

Below is the step-by-step process of updating the Network Topology.

1. Initially, the router knows the information about its neighbouring nodes only. Although, the router has information for both incoming and outgoing interfaces, we consider only the routers that can be reached using the outgoing interfaces only.
2. In the first iteration, a request message is sent to each of the neighbouring nodes (nodes that can be reached via outgoing interfaces), requesting information about their neighbouring nodes.
3. In the response message, the routers will send information about the neighbouring routers (all its incoming and outgoing interfaces).

4. While sending the response, if the router knows the interface to use for the message to reach the destination, it will use the same. Otherwise, it will send the response through all the outgoing interfaces.
5. Once the response message is received, the router will add the new routers and cost of the links into its database.
6. Now the router will send new request messages to the new routers, which can be reached using the outgoing link, that were detected in the previous iteration.
7. This process is repeated until we get no new router information at the end of the iteration.

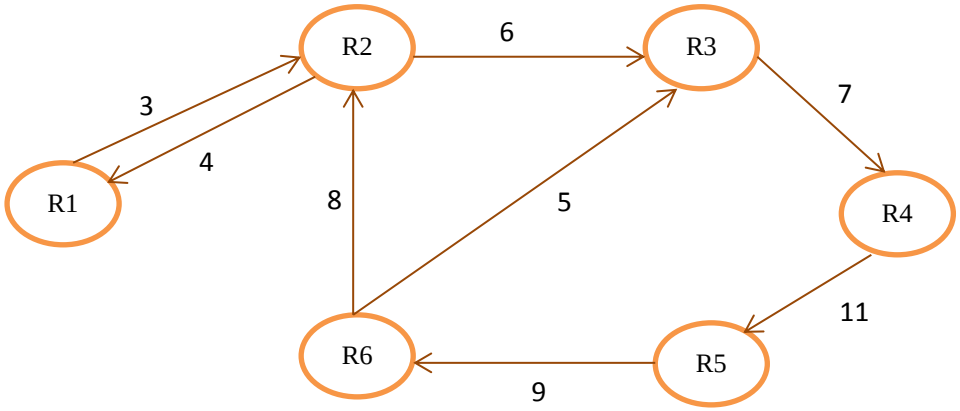
Termination Criteria for update process

At each and every iteration, the router receives response message containing information about routers in the network topology. Since only those routers information is provided which can be reached via an outgoing interface, when we find all the outgoing interfaces of all routers, it means that we found all the incoming interfaces of all routers. Hence, in the last iteration, we will receive response messages containing router information which are already present in the database. So this will serve as termination criteria for the update process.

Simulation results

For the below network topology, results for each and every iteration have been documented for routers R1 and R6.

Network Topology

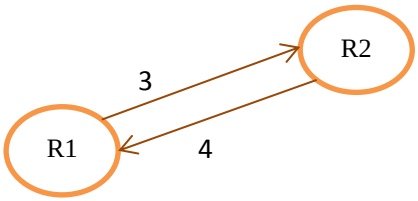


Results for router R1

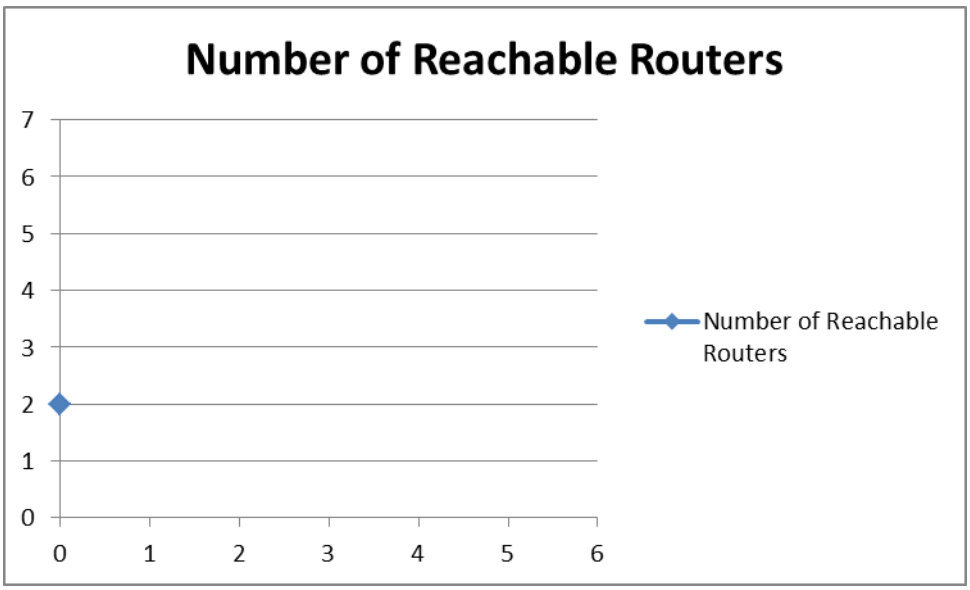
Below are the results of the simulation process for every iteration of router R1. The results include the progress of the update process represented in the form of a graph.

Iteration – 0:

In this iteration, router R1 knows only about its immediate neighbours and since we consider only the outgoing links, the topology looks like below.



The metric graph representation is given below.

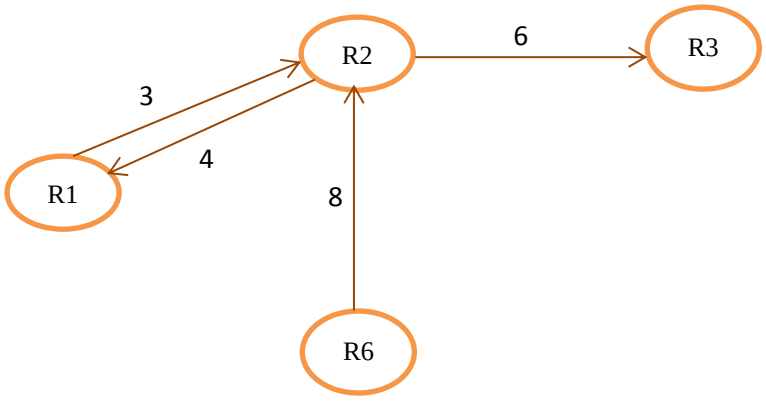


The simulation program output is:

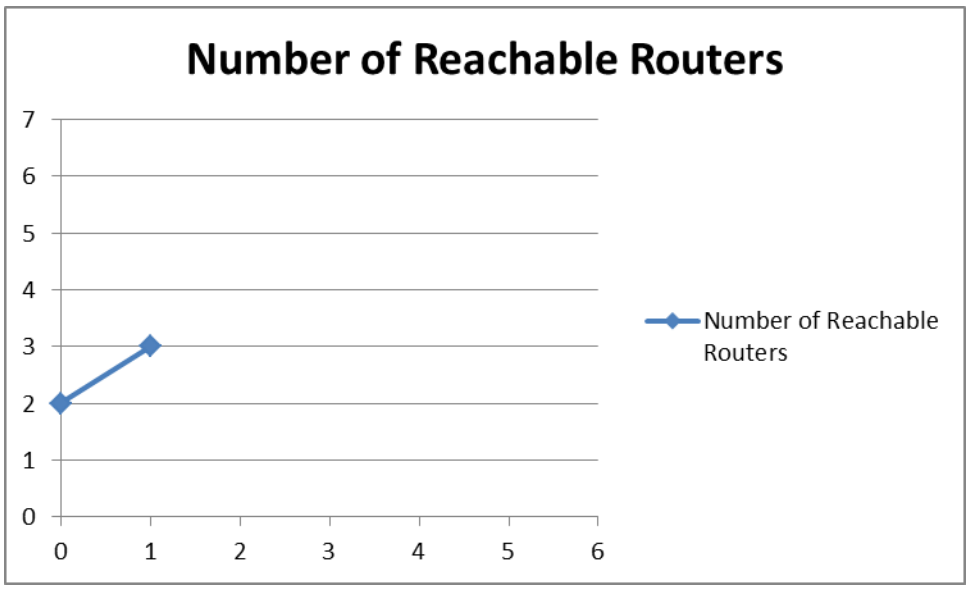
	R1	R2
R1	-1	3
R2	4	-1

Iteration – 1:

In this iteration, router R1 requests R2 about its neighbouring nodes and thus gets to know about routers R3 and R6. The network topology looks like below.



The metric graph representation is given below.



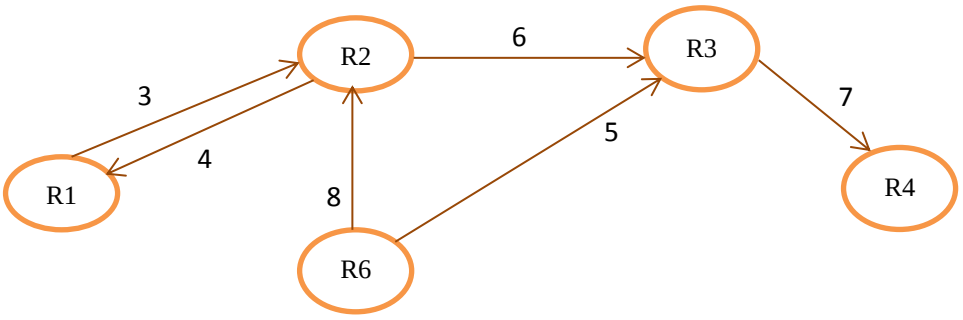
The simulation program output is:

Router - 1 asks Router - 2

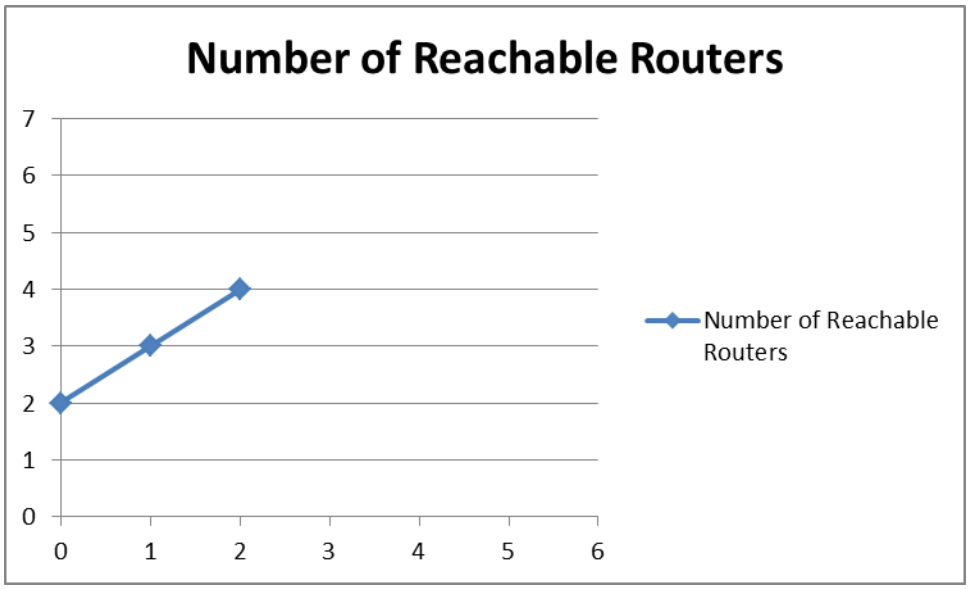
	R1	**R2**	**R3**	**R6**
R1	-1	3	0	0
R2	4	-1	6	0
R3	0	0	-1	0
R6	0	8	0	-1

Iteration – 2:

In this iteration, router R1 requests R3 about its neighbouring nodes. Note that R6 router is not requested because; it has no incoming links through the already discovered routers. The network topology looks like below.



The metric graph representation is given below.



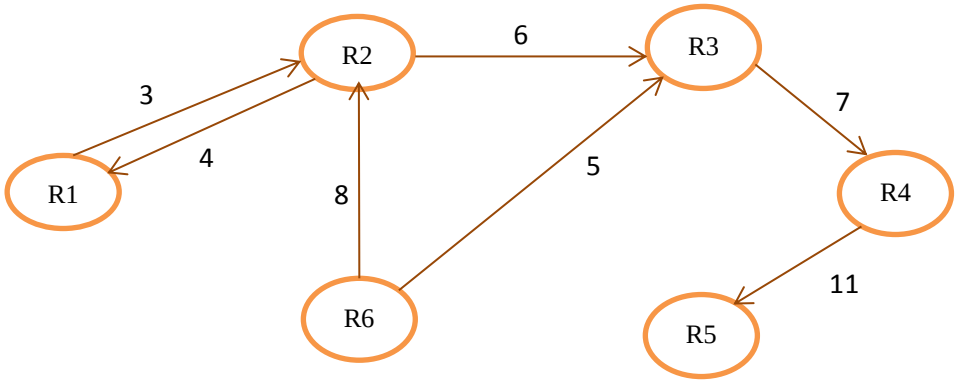
The simulation program output is:

Router - 1 asks Router - 3

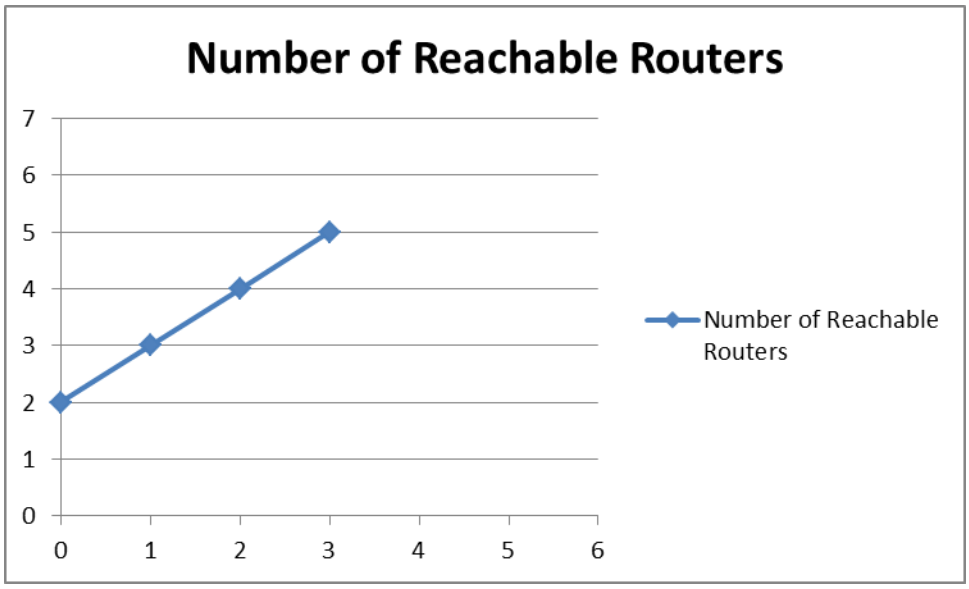
	R1	R2	R3	R6	R4
R1	-1	3	0	0	0
R2	4	-1	6	0	0
R3	0	0	-1	0	7
R6	0	8	5	-1	0
R4	0	0	0	0	-1

Iteration – 3:

In this iteration, router R1 gets to know about router R5 from the response received from router R4. The network topology looks like below.



The metric graph representation is given below.



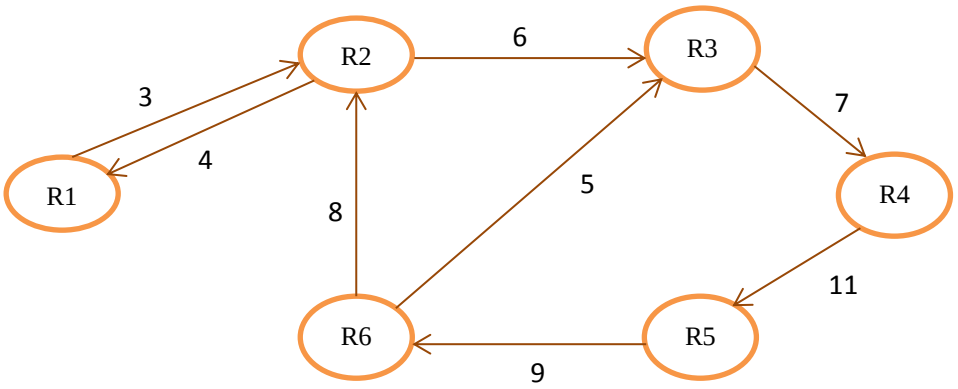
The simulation program output is:

Router - 1 asks Router - 4

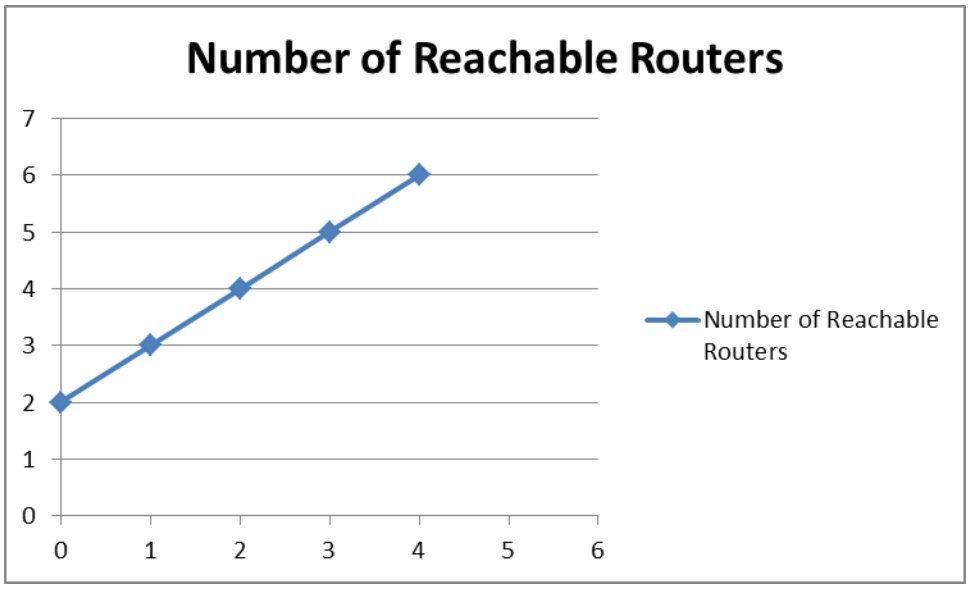
	R1	R2	R3	R6	R4	R5
R1	-1	3	0	0	0	0
R2	4	-1	6	0	0	0
R3	0	0	-1	0	7	0
R6	0	8	5	-1	0	0
R4	0	0	0	0	-1	11
R5	0	0	0	0	0	-1

Iteration – 4:

In this iteration, router R1 gets to know about router R6 from the response received from router R5. The network topology looks like below.



The metric graph representation is given below.



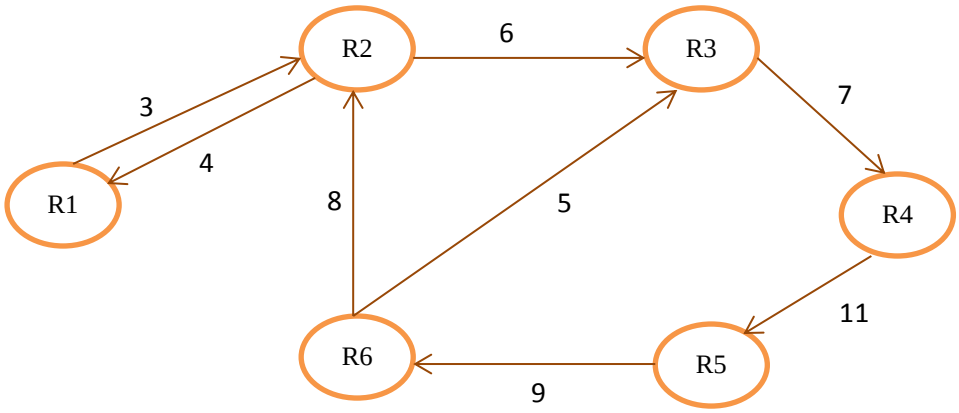
The simulation program output is:

Router - 1 asks Router - 5

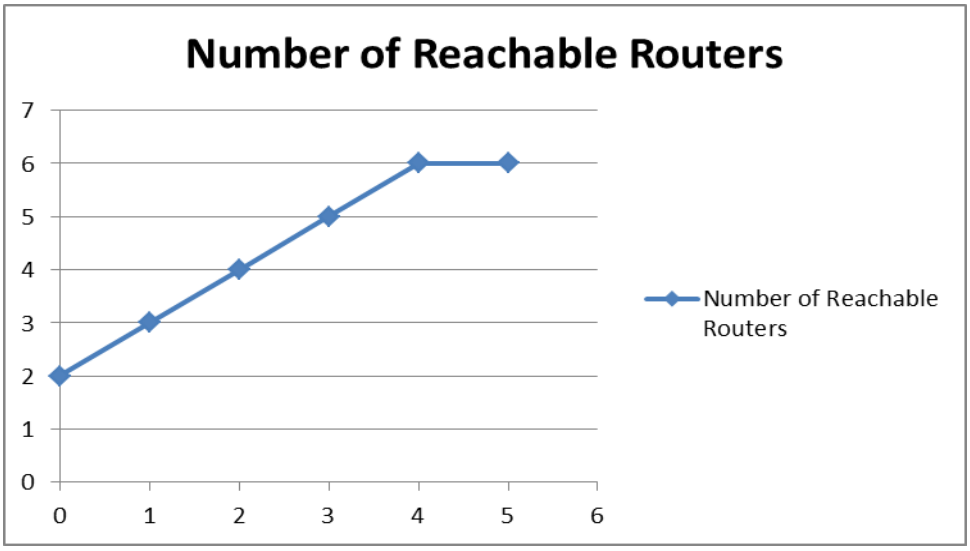
	R1	R2	R3	R6	R4	R5
R1	-1	3	0	0	0	0
R2	4	-1	6	0	0	0
R3	0	0	-1	0	7	0
R6	0	8	5	-1	0	0
R4	0	0	0	0	-1	11
R5	0	0	0	9	0	-1

Iteration – 5:

In this iteration, router R1 gets to know about router R2 and R3, from the response from router R6, which are already known and so this satisfies the termination criteria and this is the last iteration. The network topology looks like below.



The metric graph representation is given below.



The simulation program output is:

Router - 1 asks Router - 6

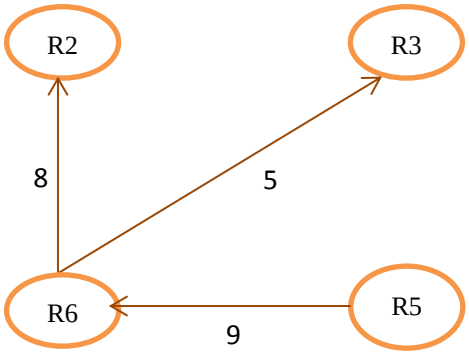
	R1	R2	R3	R6	R4	R5
R1	-1	3	0	0	0	0
R2	4	-1	6	0	0	0
R3	0	0	-1	0	7	0
R6	0	8	5	-1	0	0
R4	0	0	0	0	-1	11
R5	0	0	0	9	0	-1

Results for router R6

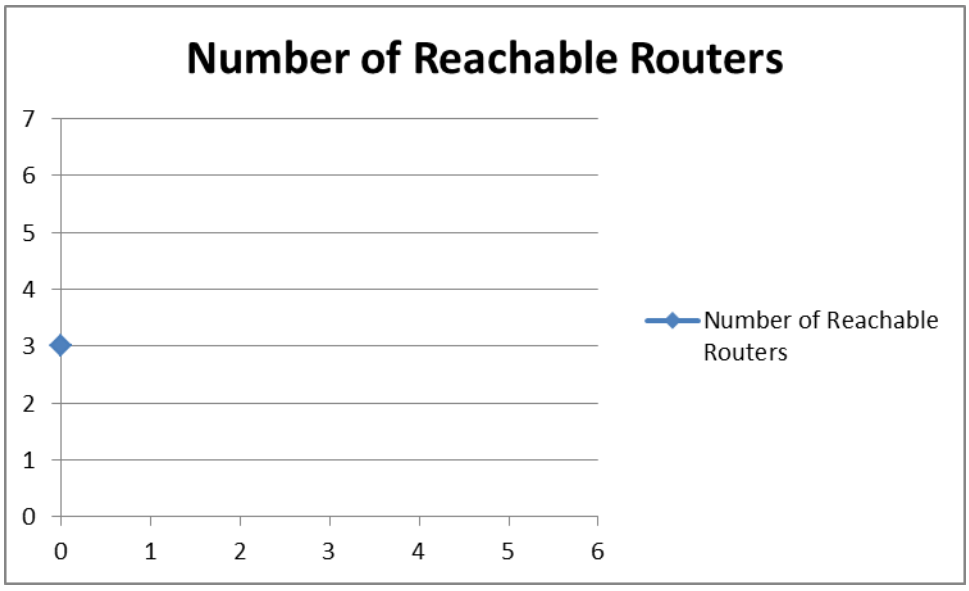
Below are the results of the simulation process for every iteration of router R6. The results include the progress of the update process represented in the form of a graph.

Iteration – 0:

In this iteration, router R6 knows only about its immediate neighbours and since we consider only the outgoing interfaces, the topology looks like below.



The metric graph representation is given below.

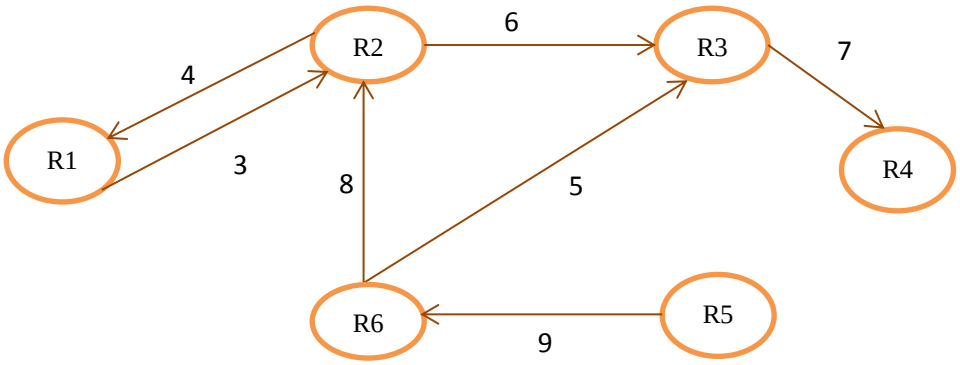


The simulation program output is:

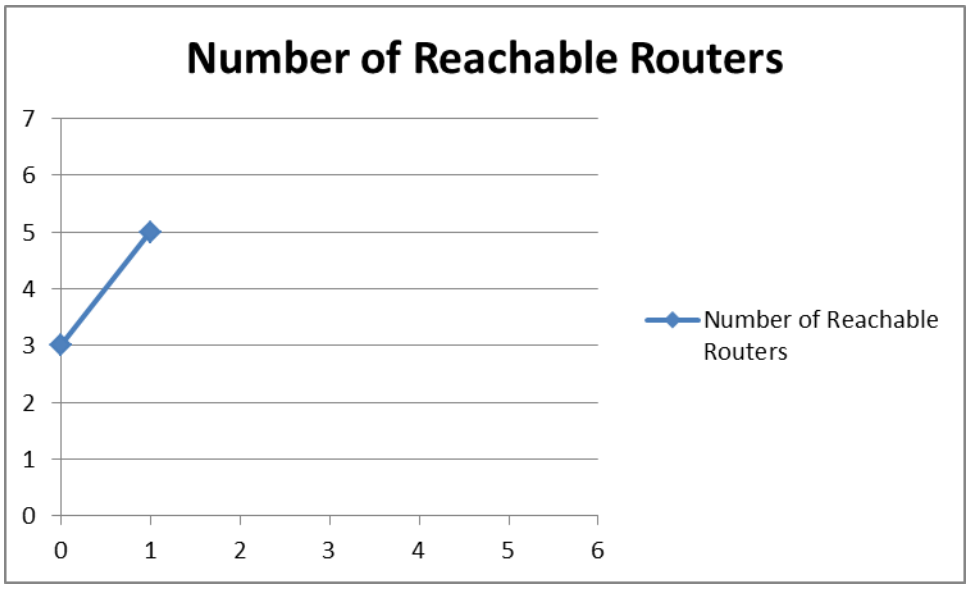
	R6	R2	R3	R5
R6	-1	8	5	0
R2	0	-1	0	0
R3	0	0	-1	0
R5	9	0	0	-1

Iteration – 1:

In this iteration, router R6 discovers routers R1, R3 from the response from R2 and router R4 from the response from R6. Hence the topology looks like below.



The metric graph representation is given below.



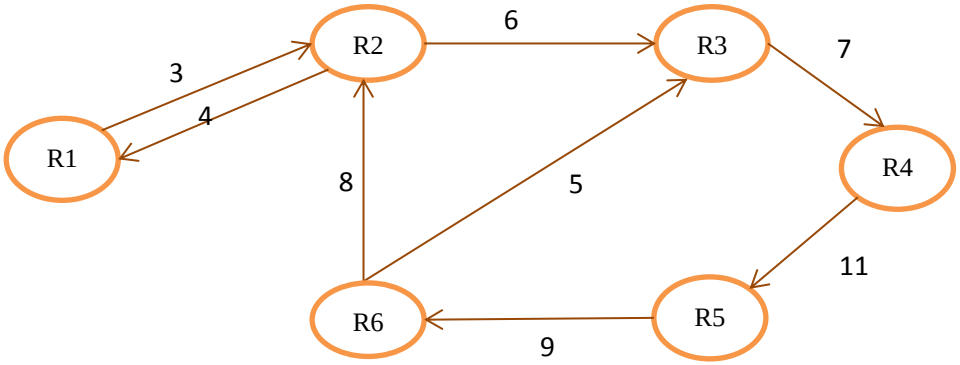
The simulation program output is:

Router - 6 asks Router – 2, Router - 3

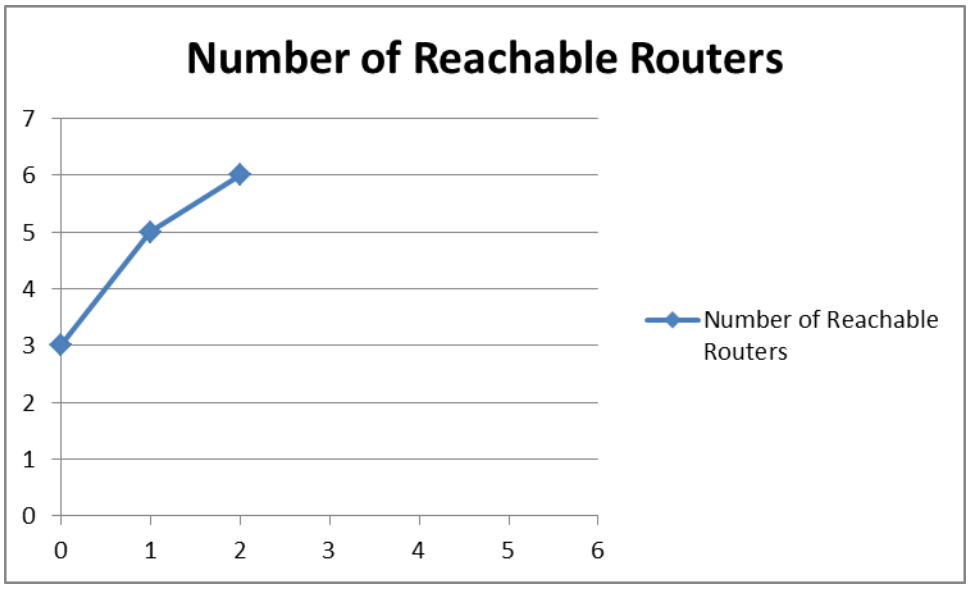
	R6	R2	R3	R5	R1	R4
R6	-1	8	5	0	0	0
R2	0	-1	6	0	4	0
R3	0	0	-1	0	0	7
R5	9	0	0	-1	0	0
R1	0	3	0	0	-1	0
R4	0	0	0	0	0	-1

Iteration – 2:

In this iteration, router R6 discovers router R2 from response from R1 and router R5 from response from R4. The topology looks like.



The metric graph representation is given below.



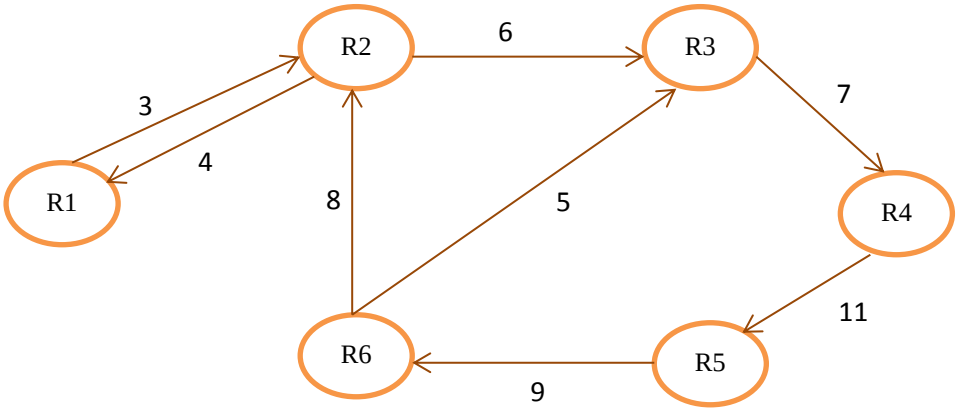
The simulation program output is:

Router - 6 asks Router – 1, Router - 4

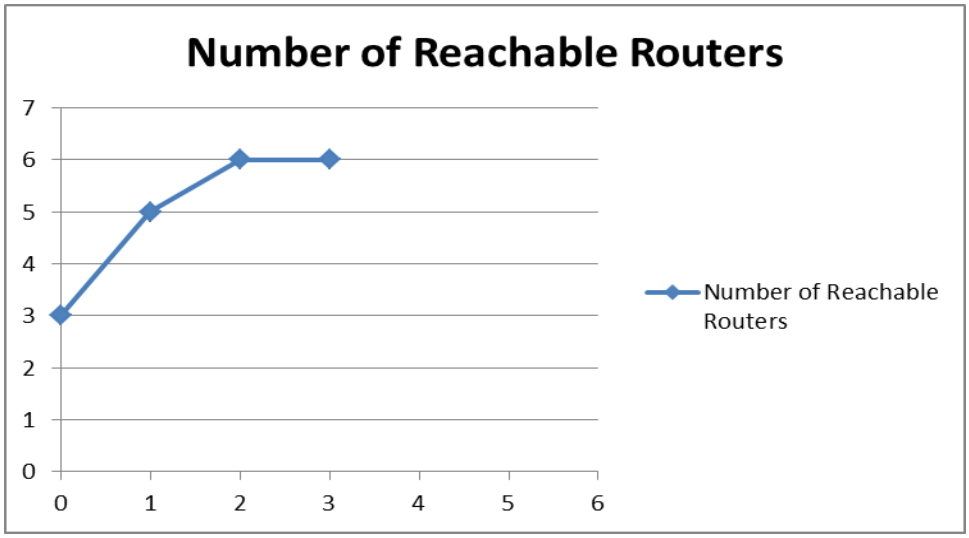
	R6	R2	R3	R5	R1	R4
R6	-1	8	5	0	0	0
R2	0	-1	6	0	4	0
R3	0	0	-1	0	0	7
R5	9	0	0	-1	0	0
R1	0	3	0	0	-1	0
R4	0	0	0	11	0	-1

Iteration – 3:

In this iteration, router R6 discovers no new routers from the response from router R5 and this satisfies the termination criteria and this is the last iteration. The topology looks like below.



The metric graph representation is given below.



The simulation program output is:

Router - 6 asks Router – 5

	R6	R2	R3	R5	R1	R4
R6	-1	8	5	0	0	0
R2	0	-1	6	0	4	0
R3	0	0	-1	0	0	7
R5	9	0	0	-1	0	0
R1	0	3	0	0	-1	0
R4	0	0	0	11	0	-1

Source Code

```
public class NetworkTopology {
   private static int[][] inputMatrix;
   private static int numberOfRouters;
/* Main Method */
   public static void main(String args[]){
      getNetworkTopologyInput();
      System.out.println("\nRouter-1 : ");
      computeForRouter(0);
      System.out.println("\nRouter-6 : ");
      computeForRouter(5);
   }
 /* Method to get Input Matrix */
   public static void getNetworkTopologyInput(){
      numberOfRouters = 6;
      inputMatrix = new int[6][6];
      int i,j;
      for (i=0; i< numberOfRouters; i++){
         for (j=0; j< numberOfRouters; j++){
            if (i == j){
               inputMatrix[i][j] = -1;
            }else{
               inputMatrix[i][j] = 0;
            }
         }
      }
      /* Actual cost values */
      inputMatrix[0][1] = 3;
      inputMatrix[1][0] = 4;
      inputMatrix[1][2] = 6;
      inputMatrix[2][3] = 7;
      inputMatrix[3][4] = 11;
```

```
        inputMatrix[4][5] = 9;
        inputMatrix[5][1] = 8;
        inputMatrix[5][2] = 5;

        System.out.println("Input Matrix\n");
        System.out.print("     ");
        for (i=0; i<numberOfRouters; i++){
            j = i + 1;
           System.out.print(" R" + j + "  ");
        }
        System.out.print("\n");

        for (i=0; i< numberOfRouters; i++){
           i++;
           System.out.print(" R" + i);
           i--;
           for (j=0; j< numberOfRouters; j++){
              System.out.print("  " + inputMatrix[i][j] + "  ");
           }
           System.out.print("\n");
        }
    }
/* Simulating the Network Topology Update Process for Router */
    public static void computeForRouter(int router){
        int iteration, newRouterFound, tempRouterNum, routerProcessed;
        int i, j, k ,nextNewRouterToBeProcessed, newOutRtrListPos, newAllRtrListPos, lastNewRouterPos;
        int[] newOutRtrList = new int[numberOfRouters];
        int[] newAllRtrList = new int[numberOfRouters];
        int rowRouter, colRouter, rtrOutInd, rtrInInd;
        iteration = -1;

        for(i=0; i< numberOfRouters; i++){
           newOutRtrList [i] = -1;
```

```
        newAllRtrList [i] = -1;
      }
      newRouterFound = 0;
      newOutRtrListPos = 0;
      newAllRtrListPos = 0;
      nextNewRouterToBeProcessed=0;
      newOutRtrList[newOutRtrListPos] = router;
      newAllRtrList[newOutRtrListPos] = router;
      newOutRtrListPos++;
      newAllRtrListPos++;
      do{
        newRouterFound = 0;
        iteration++;
        lastNewRouterPos = newOutRtrListPos;
        for (i=nextNewRouterToBeProcessed; i < lastNewRouterPos; i++){
/* Only routers that can be reached will be present in newOutRtrList  Array */
          tempRouterNum = newOutRtrList[i];
          for (j=0; j<numberOfRouters; j++){
            if (inputMatrix[tempRouterNum][j] > 0){
              routerProcessed = 0;
              for (k=0; k < numberOfRouters; k++){
                if (j == newOutRtrList[k]){
                  routerProcessed = 1;
                }
              }
              if (routerProcessed == 0){
                newOutRtrList[newOutRtrListPos] = j;
                newOutRtrListPos++;
                newRouterFound = 1;
              }
              routerProcessed = 0;
              for (k=0; k < numberOfRouters; k++){
                if (j == newAllRtrList[k]){
```

```
                    routerProcessed = 1;
                  }
                }
/* All found routers will be present in newAllRtrList  Array */
                if (routerProcessed == 0){
                  newAllRtrList[newAllRtrListPos] = j;
                  newAllRtrListPos++;
                }
              }
              if (inputMatrix[j][tempRouterNum] > 0){
                routerProcessed = 0;
                for (k=0; k < numberOfRouters; k++){
                  if (j == newAllRtrList[k]){
                    routerProcessed = 1;
                  }
                }
                if (routerProcessed == 0){
                  newAllRtrList[newAllRtrListPos] = j;
                  newAllRtrListPos++;
                  newRouterFound = 1;
                }
              }
            }
          }
/* Printing the Iteration details */
          System.out.println("\nIteration : " + iteration + "\n");
          if (iteration != 0){
            router++;
            System.out.print("Router - " + router + " asks ");
            router--;
            for (i=nextNewRouterToBeProcessed; i < lastNewRouterPos; i++){
              tempRouterNum = newOutRtrList[i];
              tempRouterNum++;
```

```
            System.out.print(" Router - " + tempRouterNum + " ");
        }
        System.out.print("\n");
    }
    nextNewRouterToBeProcessed = lastNewRouterPos;
    System.out.print("      ");
    for (j=0; j<newAllRtrListPos; j++){
        colRouter = newAllRtrList[j];
        colRouter++;
        System.out.print(" R" + colRouter + "  ");
    }
    System.out.print("\n");
    for (i=0; i<newAllRtrListPos; i++){
        rowRouter = newAllRtrList[i];
        rowRouter++;
        System.out.print(" R" + rowRouter + "  ");
        rowRouter--;
/* Condition to check whether the router printed on the row is a router on newOutRtrList */
        rtrOutInd = 0;
        for (k=0; k < lastNewRouterPos; k++){
            if (rowRouter == newOutRtrList[k]){
                rtrOutInd = 1;
            }
        }
        for (j=0; j<newAllRtrListPos; j++){
            colRouter = newAllRtrList[j];
            if (rtrOutInd == 1){
                System.out.print("  " + inputMatrix[rowRouter][colRouter] + "  ");
            }else{
/* Condition to check whether the router printed on the column is a router on newOutRtrList */
                rtrInInd = 0;
                for (k=0; k < lastNewRouterPos; k++){
                    if (colRouter == newOutRtrList[k]){
```

```
                    rtrInInd = 1;
                  }
                }
                if (rtrInInd == 1){
                  System.out.print("  " + inputMatrix[rowRouter][colRouter] + "  ");
                }else{
                  if (i == j){
                    System.out.print("  -1  ");
                  }else{
                    System.out.print("  0  ");
                  }
                }
              }
            }
            System.out.print("\n");
          }
        }while(newRouterFound == 1);
    }
}
```