

IMPLEMENTATION AND SIMULATION OF DISTANCE VECTOR ROUTING PROTOCOL

*Report submitted to the SASTRA Deemed to be University
as the requirement for the course*

CSE302: COMPUTER NETWORKS

Submitted by

SHYAM SUNDAR G.S
(Reg. No.: 124015145, B.Tech. Information Technology)

DECEMBER,2022.



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401



SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled “**IMPLEMENTATION AND SIMULATION OF DISTANCE VECTOR ROUTING PROTOCOL**” submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Shri Shyam Sundar G.S(Reg. No.124015145, B.Tech. Information Technology)** during the academic year 2022-23, in the School of Computing

Project Based Work *Viva voce* held on _____

Examiner 1

Examiner 2

List of Figures

Figure No.	Title	Page No.
1.1	Different Layers of OSI Model and their Functions	1
1.2	A Router connected between two devices	2
1.3	Types of Routing	2
1.4	Types of Routing Algorithms	3
1.5	Routing Metrics	5
1.6	DVRP Example	8
2.1	Node Diagram	10
2.2	Data Files	18
2.3	Network Architecture	19
3.1	Packet Acknowledgment	32
3.2	Simulation Event List	33

List of Tables

Table No.	Table name	Page No.
1.1	Routing Metrics for Optimal Path	7

ABBREVIATIONS

IP - Internet Protocol

TCP/IP - Transmission Control Protocol/ Internet Protocol

BPS - Bits Per Second

OSI - Open Systems Interconnection

ISO - International Organization for Standardization

ARPRANET - Advanced Research Projects Agency Network

DVR - Distance Vector Routing

DVRP - Distance Vector Routing Protocol

RIP – Routing Information Protocol

IGRP – Interior Gateway Routing Protocol

BGP – Border Gateway Protocol

IPV4 – Internet Protocol version 4

IPV6 – Internet Protocol version 6

ASCII – American Standard Code for Information Interchange

Abstract

When we establish a communication and share some information, we must follow a routing protocol to traverse the data packets among routers to achieve the communication. Routing establishes the connection between the source and destination and takes care of the traversal of these data packets. The Routing Protocol specifies how multiple routers communicate among themselves to select a possible route to successfully achieve this communication.

In the modern era, it is important not only to achieve communication but to achieve it in a quick way, so we must ensure that the data packets traverse across the shortest path available to them. Distance Vector Routing algorithm is used to find the shortest path with the knowledge of Bellman-Ford algorithm. In this project, we will implement the Distance Vector Routing protocol using Java and we will also simulate the same with the help of Cisco packet tracer.

KEY WORDS: Networks, Routers, Routing Algorithm, Distance Vector Routing Algorithm.

Table of Contents

Title	Page No.
Bonafide Certificate	ii
List of Figures	iii
List of Tables	iii
Abbreviations	iv
Abstract	v
1. Introduction, Merits and Demerits	1-9
1.1. Types of Routing	2
1.2. Types of Routing Algorithms	3
1.3. Routing Metrics	5
1.4. Distance Vector Routing Algorithm	7
1.5. Merits	9
1.6. Demerits	9
2. Source Code	10-25
2.1. Implementation	10
2.2. Simulation	19
3. Results and Discussion	26-33
3.1 Implementation Output	26
3.2 Simulation Output	30
4. Conclusion and Future Plans	34
5. References	35

CHAPTER-1

INTRODUCTION, MERITS AND DEMERITS

The Open Systems Interconnection Model (OSI Model) is a conceptual model developed by the International Organization for Standardization to enable communication devices to communicate using standard protocols. It consists of seven layers namely Application Layer, Presentation Layer, Session Layer, Transport Layer, Network Layer, Data Link Layer and Physical Layer.

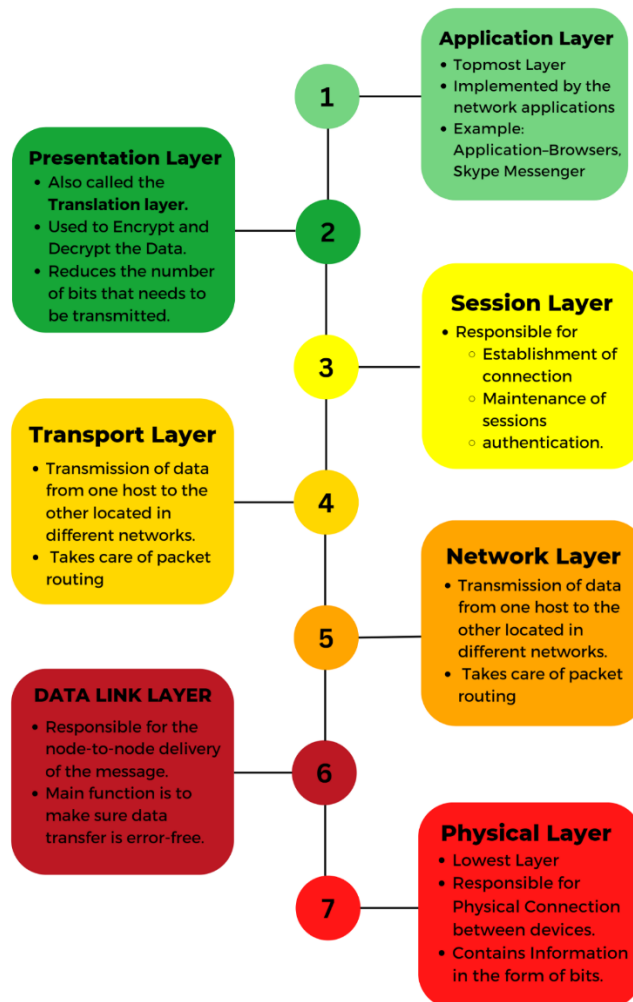


Fig 1.1 – Different Layers of OSI Model and their Functions

Routing works at the Network Layer in the OSI reference model. Routing is used for selecting a path from one router to another router. Routing selects the transmission path for packets when they traverse from one router to another router (or) from one end system to another end system. This process is carried out by a device named Router. The Router is a networking device which forwards the incoming packets based on the forwarding table and the header of the packet. To transmit the incoming packets, the router makes use of the Routing algorithms. The Routing algorithms are nothing but a software used to decide the optimal path via which the packets can be sent/received. To determine the optimal path, the Routing algorithms develop a routing table and update it after every movement of the packets.

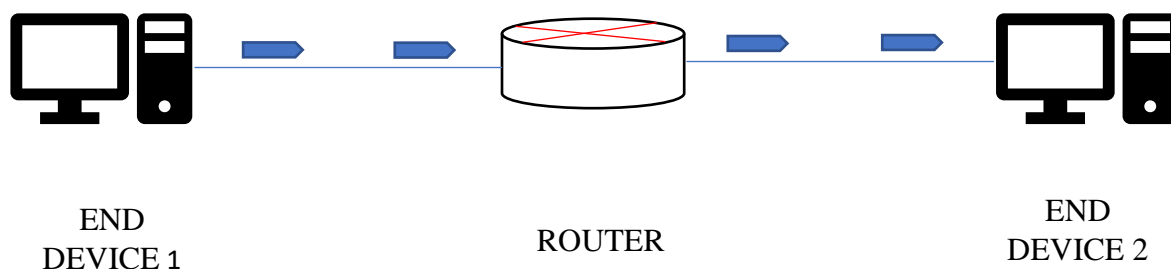


Fig 1.2 – A Router connected between two devices

1.1. TYPES OF ROUTING:

There are three types of Routing. They are:

- Static Routing
- Default Routing
- Dynamic Routing

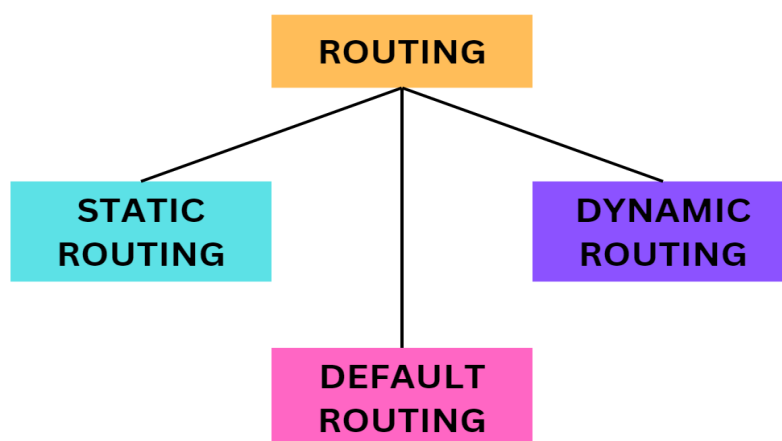


Fig 1.3 – Types of Routing

1.1.1. Static Routing:

The optimal path between the possible source and destination are pre-defined and are fed into the routing table of the routers in the given network. It is also known as Non-Adaptive Routing. The router can transmit the packets only in the pre-defined path and not in any other path. The routing decisions are not based on the topology of the network.

1.1.2. Default Routing:

The Router is configured to send all the packets to the same hop device irrespective of whether the packets belong to the same network or different networks. All the packets are transmitted to the router which is configured via the default routing.

1.1.3. Dynamic Routing:

This method allows the router update the routing table by itself and make automatic adjustments. In this way, the router itself can decide the path that the incoming packet must follow to reach the destination. If one route goes down, automatic adjustments are done to make sure that the packet reaches the desired destination.

It has two features:

- All the routers in the network must be running the same dynamic protocol to exchange the routes.
- If a router detects a topology change, it passes it to all other routers.

1.2. TYPES OF ROUTING ALGORITHMS:

The Router selects the optimal path for the transmission of packets with the help of Routing Algorithms. These Routing algorithms are broadly classified into two types namely, Adaptive and Non-Adaptive Routing Algorithms.

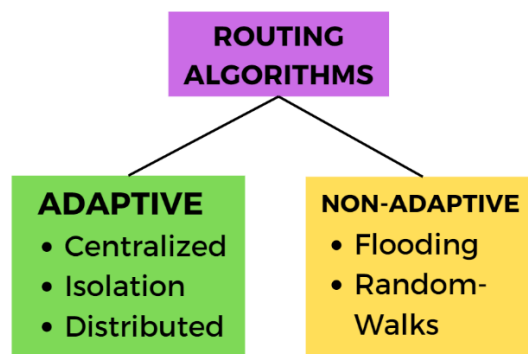


Fig 1.4 – Types of Routing Algorithms

1.2.1. Adaptive Routing Algorithms:

An adaptive routing algorithm makes routing decisions based on the topology and the traffic on the network. It is also called as Dynamic Routing algorithm. They make use of dynamic data like current topology, load and delay to select the path of transmission. Distance, range of hop and approximated transit time are the main parameters with respect to this algorithm.

It can be further classified into three parts:

Centralized Algorithm:

This algorithm calculates the least cost path between the source and the destination by using complete knowledge about the network and its topology. Also known as global routing algorithm. This algorithm takes the connectivity between all nodes and all link costs as inputs.

Example: Link State Routing Algorithm

Isolated Algorithm:

This algorithm obtains the routing information by making use of the local information unlike the Centralized algorithm. In this algorithm every node makes its routing choices without seeking information from the other nodes.

Example: Hot Potato Routing

Distributed Algorithm:

The shortest path between the source and destination is calculated in an iterative and distributed way. It is also known as decentralized algorithm. The node contains information only about its attached nodes initially, later it computes optimal path to all other nodes in an iterative and distributive manner.

Example: Distance Vector Routing Protocol

1.2.2. Non-Adaptive Routing Algorithms:

A non-adaptive routing algorithm is a one in which the routing decisions are not taken based upon the topology and traffic of the network. This is also known as Static routing algorithm. They store the routing information in the router in which it is booted up.

It can be classified into further two types:

Flooding:

Every incoming packet is sent to all outgoing links except from the one which the packet was received. It can occur in three ways namely Controlled flooding, Uncontrolled flooding and selective flooding.

Controlled Flooding:

Certain methods can be applied to control the transmission of packets to the neighbouring nodes. Sequence Number Controlled Flooding (SNCF) and Reverse Path Forwarding (RPF) are some of the popular algorithms used.

Uncontrolled Flooding:

The incoming packets are unconditionally sent to all the nodes.

Selective Flooding:

Nodes are configured to send the packets to routers in only one direction. Better than uncontrolled flooding but not as sophisticated as controlled flooding.

Random Walks:

This is an algorithm in which the incoming packet is randomly sent to any of the neighbouring routers. Since it takes a random path, it is called as Random Walk. It is very easy to implement and will be suitable for small devices.

1.3. Routing Metrics:

A router typically makes use of the metrics to determine the optimal path for the transmission of incoming packets. Several factors play a key role in determining the optimal path. These factors are termed as metrics.

Some of the common metrics are:

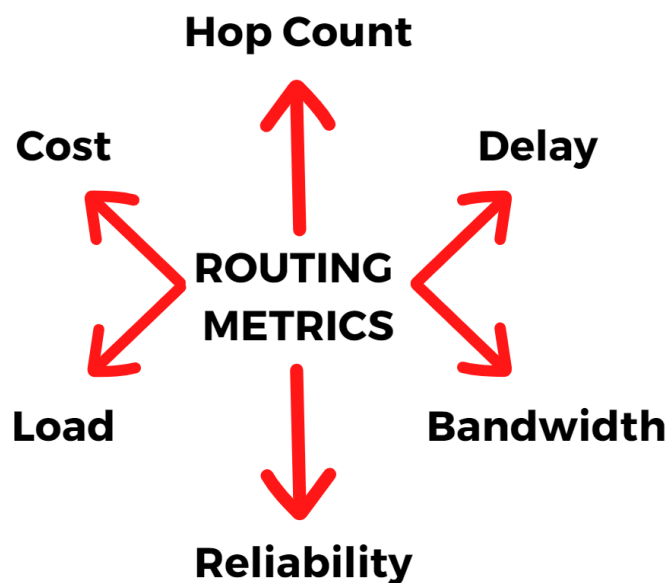


Fig 1.5 – Routing Metrics

1.3.1. Hop Count:

It is the count of passes in between networking devices such as routers, through which the packet traverses to reach the destination. Simply, it is a metric value used to measure distance based on the number of networking devices the packets travel. A path with the least number of hops will be considered as an optimal path.

1.3.2. Bandwidth:

The transmission capacity of the link is called as the Bandwidth of the link. The link with higher bandwidth is preferred to the one with lower bandwidth. It is usually measured in bits-per-second (bps). The cumulative bandwidth of all the links over a path will be determined and the one with higher bandwidth will be chosen as the optimal path.

1.3.3. Delay:

The time taken by a router to receive, queue and release a packet is known as delay. The cumulative delay of all the links over a path will be determined and the one with less delay will be chosen as the optimal path.

1.3.4. Load:

Load refers to the amount of traffic the link is occupying based on the link's capacity. When the load in a link is full, congestion occurs and congestion control protocols must be applied to reduce the load. The path which has low overall load will be chosen as the optimal path.

1.3.5. Reliability:

It is usually a fixed value and is measured dynamically. Links which experience less problems are considered more reliable and vice-versa. The value is usually measured in percentage and can be measured based on any reliability factors.

1.3.6. Cost:

Costs are arbitrary values assigned to each links. The lower the cost, the higher the optimality. The cumulative cost of all the links in a path are calculated and the one with lower cost is known as the optimal path.

METRIC	VALUE FOR OPTIMAL PATH
Hop Count	Low Hop Count
Bandwidth	Higher Bandwidth
Delay	Low Delay
Load	Low Load
Reliability	High Reliability
Cost	Less Cost

Table 1.1 – Routing Metrics for Optimal Path

1.4. Distance Vector Routing Algorithm:

Distance Vector Routing protocol is a dynamic routing protocol. Unlike the Link state routing algorithm, Distance Vector algorithm is iterative, asynchronous, and distributed. It requires that a router periodically inform its neighbours about the topology changes. It was previously called as the old ARPANET routing algorithm.

The Distance Vector Routing algorithm is also called as Bellman-Ford Algorithm as it is used to find the shortest path. It calculates the distance and direction of the vector of the next hop from the information received from the neighbouring router. Every node in the network will have information about its neighbouring node. The algorithm is based on the Bellman-Ford Equation,

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

where the \min_v is calculated for all of x's neighbours.

In DVR algorithm, a node maintains the following information:

- Network Information:
 - Every node in network will have information about its neighbouring node.
 - The nodes are designed in such a way that they share information to all other nodes.
- Routing Pattern:
 - The data shared are only transmitted to the nodes that are linked directly to one or more nodes in the network.
- Data Sharing:
 - The data is shared time to time to other nodes to indicate the network topology changes.

1.4.1. Algorithm:

At each node x,

Initialization

for all destinations y in N:

$D_x(y) = c(x,y)$ // If y is not a neighbor then $c(x,y) = \infty$

for each neighbor w

$D_w(y) = ?$ for all destination y in N.

for each neighbor w

send distance vector $D_x = [D_x(y) : y \text{ in } N]$ to w

loop

wait(until I receive any distance vector from some neighbor w)

for each y in N:

$D_x(y) = \min_v \{ c(x,v) + D_v(y) \}$

If $D_x(y)$ is changed for any destination y

Send distance vector $D_x = [D_x(y) : y \text{ in } N]$ to all neighbors

forever

1.4.2. Example:

Let us look at a simple example to get a clear and precise understanding of the Distance vector routing algorithm. The mentioned example has been taken from the book: Computer Networking – A Top-Down Approach (7th edition).

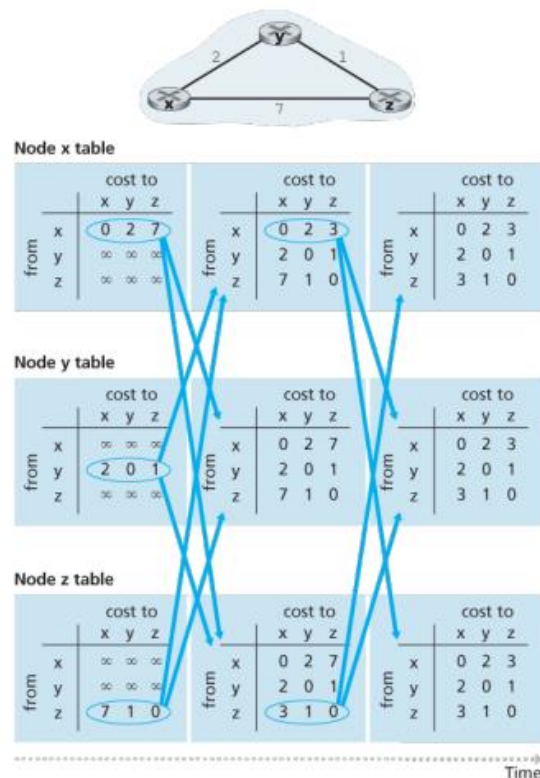


Fig 1.6 – DVRP Example

1.5. Merits:

- Easy to implement in small networks.
- The task of debugging is not very complex.
- It has a very limited redundancy in small networks.

1.6. Demerits:

- Suffers from Count-to-Infinity problem (Count-to-Infinity Problem: A broken link must be updated to all routers in the network immediately) The DVRP algorithm takes a considerable amount of time to update broken links.
- The convergence time (Time taken to produce appropriate routing tables) is excessive in terms of large networks. This is one of the reasons Distance Vector Routing Protocol is not preferred for large networks.
- As the data is shared to the neighbouring nodes in a periodic manner, it creates an additional traffic in the network which might delay the transmission of some of the packets.

CHAPTER 2

SOURCE CODE

2.1. IMPLEMENTATION:

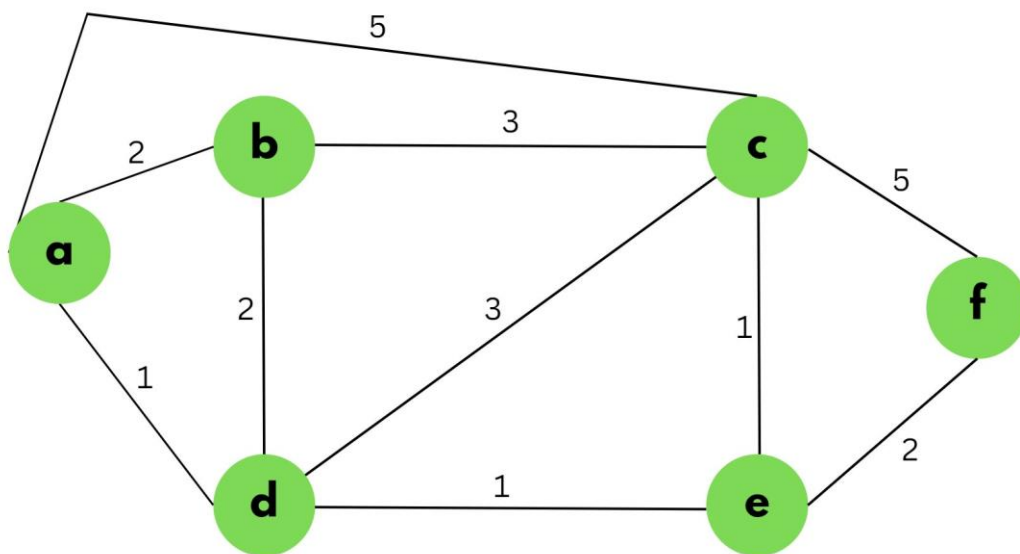


Fig 2.1 – Node Diagram

DVR.java:

```
import java.io.File;
import java.io.IOException;
import java.util.HashSet;
import java.util.Scanner;

public class DVR {

    public static void main(String[] args) {

        HashSet<Integer> hashSetPorts = new HashSet<>();
        if (args.length == 0) {
            System.out.println("Include The Directory Path of the file containing
node information!");
            return;
        } else if (args.length > 1) {
```



```

        System.out.println("No more than one argument must be given!");
        return;
    }

    String path = args[0];
    File dir = new File(path);

    if (!dir.isDirectory()) {
        System.out.println("Incorrect Directoy Path! Enter Correct path!");
        return;
    }
    File data[] = dir.listFiles();
    int size = data.length;
    int[] ports = new int[size];
    String allNodes = "";

    System.out.println("Initilize the Port Number for all " + size + " Routers");
    Scanner scanner = new Scanner(System.in);

    for (int i = 0; i < size; i++) {
        String val = data[i].getName();
        val = val.substring(0, val.indexOf(".dat"));
        System.out.println("Enter Port No for Router: " + val);

        boolean status = true;

        while (status) {
            try {
                int num = Integer.parseInt(scanner.nextLine());

                if (num <= 1024 || num >= 65536) {
                    throw new NumberFormatException();
                }
                if (hashSetPorts.contains(num)) {
                    throw new Exception();
                }
                ports[i] = num;
                hashSetPorts.add(num);
                status = false;
            } catch (NumberFormatException e) {
                System.out.println("Enter a valid Port Number Higher
than 1024 and less than 65536");
                status = true;
            } catch (Exception e) {
                System.out.println("Port Number is Already in Use.");
                status = true;
            }
        }
        allNodes += " " + val + ":" + ports[i];
    }
}

```

```

        scanner.close();

        for (int i = 0; i < size; i++) {
            ProcessBuilder processBuilder = new ProcessBuilder("cmd.exe", "/c",
"start java MainRouter " + (i + 1) + "\"
+ data[i].getParent().replace("\\", "/") + "\" " + size +
allNodes);
            try {
                processBuilder.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        System.out.println("Distance Vector Algorithm Started");
    }
}

```

Router.java:

```

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.Arrays;

class MainThread extends Thread {

    private String vector;
    private int port;
    private String type;
    private int k;

    public MainThread(String type, String vector, int port) {
        this.type = type;
        this.vector = vector;
        this.port = port;
        this.k=0;
    }

    public MainThread(String type) {
        this.type = type;
    }

    public void run() {

```

```

        if (type.equalsIgnoreCase("r")) {
            MainRouter.readData();

        } else if (type.equalsIgnoreCase("w")) {

            while (k<5) {
                try {

                    MainRouter.read();
                    MainRouter.output();
                    MainRouter.broadCast();

                    Thread.sleep(6000);

                    MainRouter.distanceAlgorithm();

                    Thread.sleep(12000);

                } catch (Exception e) {
                }
                k=k+1;
            }
        } else if (type.equalsIgnoreCase("u")) {

            MainRouter.updateNetworkVectors(vector.split(":"), port);
        }
    }
}

```

```

public class MainRouter {

```

```

    public static int router_DisplayCount = 1;
    public static double[][] router_NetworkVectors;
    public static int[] router_Ports;
    public static String[] router_Nodes;
    public static int router_id;
    public static double[] router_MyVector;
    public static String[] router_MyHopList;
    public static DatagramSocket router_Socket;
    public static File router_File;
    public static String[] router_Neighbours;

```

```

    public static void setParameters(int len, String[] args, int id, String parent) {

```

```

        router_NetworkVectors = new double[len][len];
        router_Ports = new int[len];
        router_Nodes = new String[len];

```

```

        for (int i = 0; i < len; i++) {

```

```

        Arrays.fill(router_NetworkVectors[i], Double.MAX_VALUE);
        router_NetworkVectors[i][i] = 0.0;
        String[] temp = args[i + 3].split(":");
        router_Nodes[i] = temp[0];
        router_Ports[i] = Integer.parseInt(temp[1]);
    }

    router_id = id;
    router_MyVector = new double[len];
    router_MyHopList = new String[len];
    Arrays.fill(router_MyVector, Double.MAX_VALUE);
    router_MyVector[router_id - 1] = 0.0;
    router_File = new File(parent + "/" + router_Nodes[router_id - 1] + ".dat");
    try {
        router_Socket = new DatagramSocket(router_Ports[router_id - 1]);
    } catch (SocketException e) {

        e.printStackTrace();
    }
    System.out.println("Router " + router_Nodes[router_id - 1] + " is
Working..!");
}

public static void distanceAlgorithm() {

    for (int i = 0; i < router_Neighbours.length; i++) {
        int ind = indexFinder(router_Neighbours[i]);
        for (int j = 0; j < router_MyVector.length; j++) {
            if (j == router_id - 1) {
                continue;
            } else if (i == 0) {

                router_MyVector[j] = router_NetworkVectors[router_id
- 1][ind] + router_NetworkVectors[ind][j];
                router_MyHopList[j] = router_Neighbours[i];
            } else {

                if (router_MyVector[j] >
router_NetworkVectors[router_id - 1][ind] + router_NetworkVectors[ind][j]) {
                    router_MyHopList[j] = router_Neighbours[i];
                    router_MyVector[j] =
router_NetworkVectors[router_id - 1][ind] + router_NetworkVectors[ind][j];
                }

            }
        }
    }
}

```

```

    }

    public synchronized static void updateNetworkVectors(String[] vector, int port) {

        int index = 0;
        int ports_Length=router_Ports.length;
        while(index < ports_Length)
        {
            if (router_Ports[index] == port)
            {
                break;
            }
            index++;
        }
        if (index == ports_Length)
        {
            return;
        }
        for (int i = 0; i < vector.length; i++)
        {
            router_NetworkVectors[index][i] = Double.parseDouble(vector[i]);
        }
    }

    public static void 15oolean15t() {

        try {
            for (int i = 0; i < router_Neighbours.length; i++) {
                String data = "";
                for (int j = 0; j < router_MyVector.length; j++) {
                    if (router_Neighbours[i].equals(router_MyHopList[j]))
                    {
                        data = data + Double.MAX_VALUE + ":";
                    } else {
                        data += router_MyVector[j] + ":";
                    }
                }
                DatagramPacket packet = new
DatagramPacket(data.getBytes(), data.getBytes().length);
                packet.setAddress(InetAddress.getByName("localhost"));

                packet.setPort(router_Ports[indexFinder(router_Neighbours[i])]);
                router_Socket.send(packet);

            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

    }

}

public static void output() {

    System.out.println("> output number " + router_DisplayCount++);
    System.out.println();
    String src = router_Nodes[router_id - 1];
    for (int i = 0; i < router_MyVector.length; i++) {
        if (i != (router_id - 1)) {
            String dest = router_Nodes[i];
            if (router_MyVector[i] == Double.MAX_VALUE) {
                System.out.println("Optimal path " + src + "-" + dest +
": " + " no route found");
            } else {
                System.out.println("Optimal path " + src + "-" + dest +
": the next hop is " + router_MyHopList[i]
+ " and the cost is " +
router_MyVector[i]);
            }
        }
    }
}

public static void readData() {
    boolean status = true;
    while (status) {
        try {
            String type = "u";
            byte[] data = new byte[1024];
            int size = data.length;
            DatagramPacket packet = new DatagramPacket(data, size);
            router_Socket.receive(packet);
            int length = packet.getLength();
            String vector = new String(packet.getData(), 0, length);
            MainThread.updateThread = new MainThread(type, vector,
packet.getPort());

            updateThread.start();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

public static void main(String args[]) {
    int total = Integer.parseInt(args[2]);
    int currentNum = Integer.parseInt(args[0]);

```

```

String path = args[1];
setParameters(total, args, currentNum, path);

MainThread readThread = new MainThread("r");
readThread.start();

MainThread writeThread = new MainThread("w");
writeThread.start();

while (true)
    ;
}

public static void read() {
    try {
        Arrays.fill(router_NetworkVectors[router_id - 1],
Double.MAX_VALUE);
        router_NetworkVectors[router_id - 1][router_id - 1] = 0.0;
        BufferedReader br = new BufferedReader(new
FileReader(router_File));
        int length = Integer.parseInt(br.readLine());
        router_Neighbours = new String[length];
        for (int i = 0; i < length; i++) {

            String[] temp = br.readLine().split(" ");
            int ind = indexFinder(temp[0]);
            router_Neighbours[i] = temp[0];

            if (router_DisplayCount == 1) {
                router_MyHopList[ind] = temp[0];
                router_MyVector[ind] = Double.parseDouble(temp[1]);
            } else {

            }
            router_NetworkVectors[router_id - 1][ind] =
Double.parseDouble(temp[1]);

        }
        br.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public static int indexFinder(String temp) {
    int pos = -1;
    for (int i = 0; i < router_Nodes.length; i++) {
        if (router_Nodes[i].equals(temp)) {
            pos = i;
            break;


```

```


    }
    }
    return pos;
}
}

```


Data Files:

 a - Notepad


File	Edit	Format	View	Help
3				
b	2.0			
c	5.0			
d	1.0			

 b - Notepad


File	Edit	Format	View	Help
3				
a	2.0			
c	3.0			
d	2.0			

 c - Notepad


File	Edit	Format	View	Help
5				
a	5.0			
b	3.0			
d	3.0			
e	1.0			
f	1.0			

 d - Notepad

File	Edit	Format	View	Help
4				
a	1.0			
b	2.0			
c	3.0			
e	1.0			

 e - Notepad

File	Edit	Format	View	Help
3				
c	1.0			
d	1.0			
f	2.0			

 f - Notepad

File	Edit	Format	View	Help
2				
c	1.0			
e	2.0			

Fig 2.2 – Data Files

2.2. Simulation:

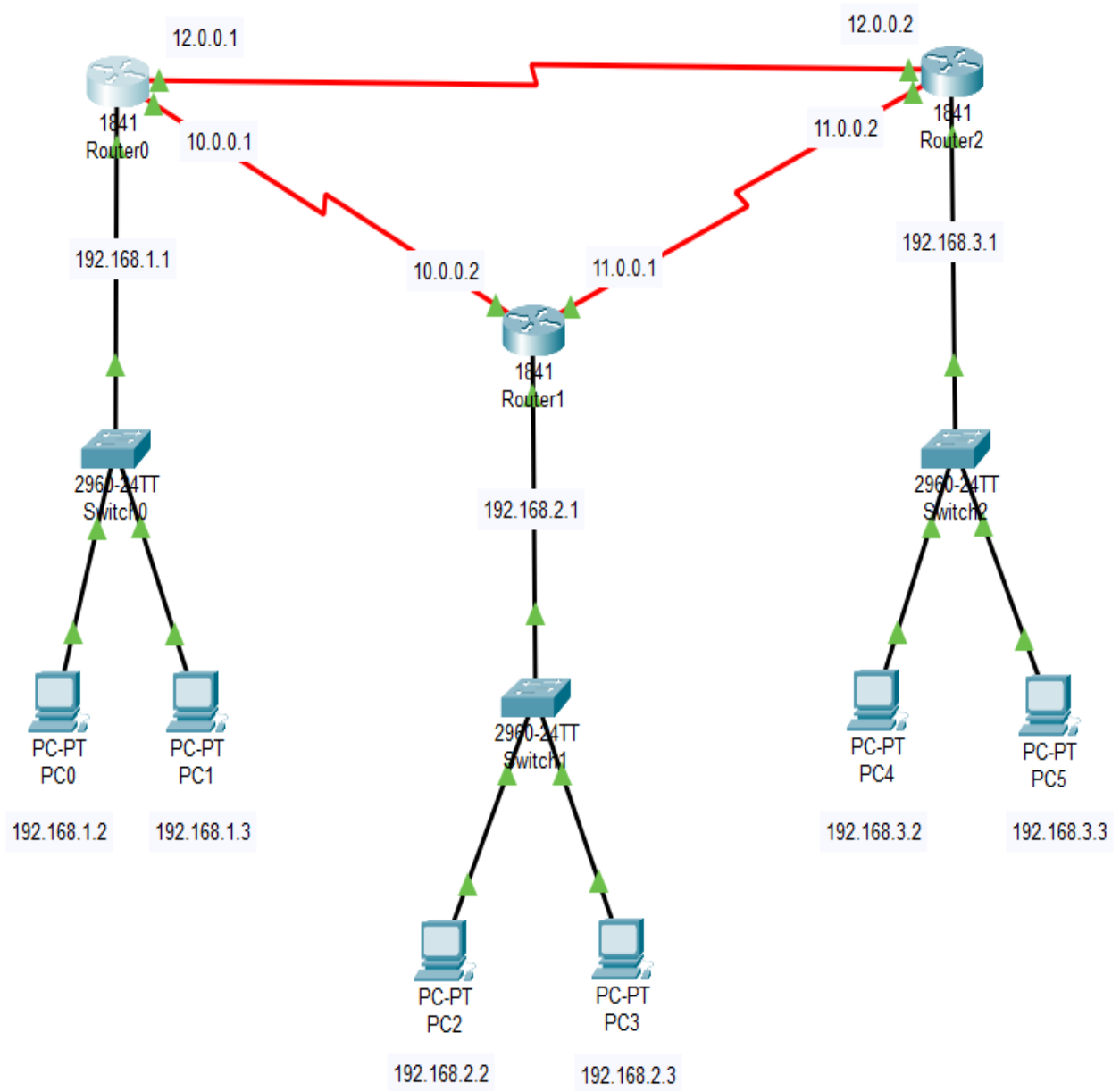
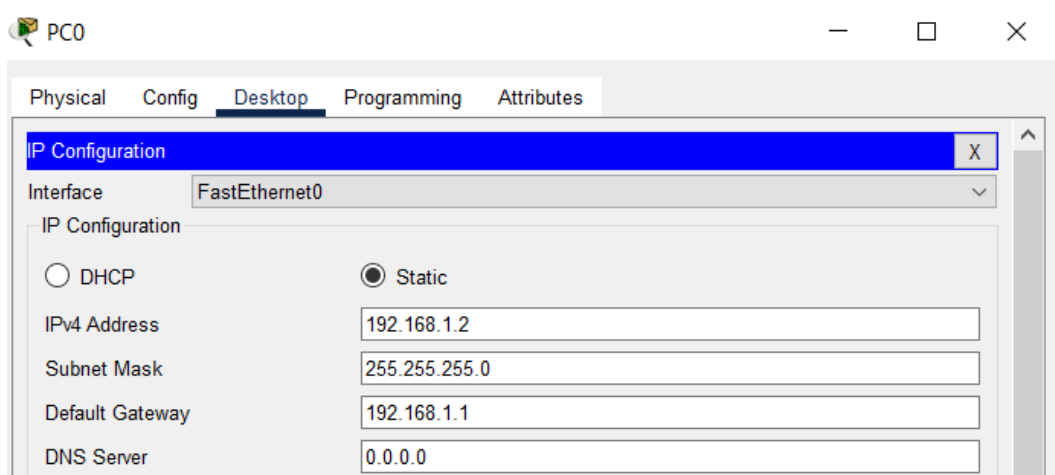


Fig 2.3 – Network Architecture

PC Configurations:

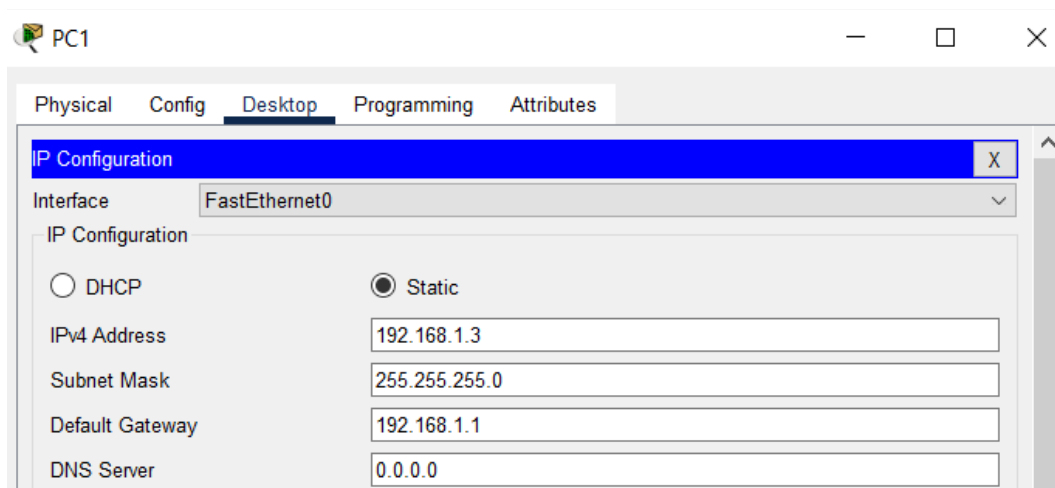
PC0:



The screenshot shows the configuration window for PC0. The 'Desktop' tab is selected. The 'IP Configuration' section is expanded, showing the 'FastEthernet0' interface. The 'Static' radio button is selected for the IP configuration. The fields are filled with the following values:

Field	Value
Interface	FastEthernet0
IP Configuration	Static
IPv4 Address	192.168.1.2
Subnet Mask	255.255.255.0
Default Gateway	192.168.1.1
DNS Server	0.0.0.0

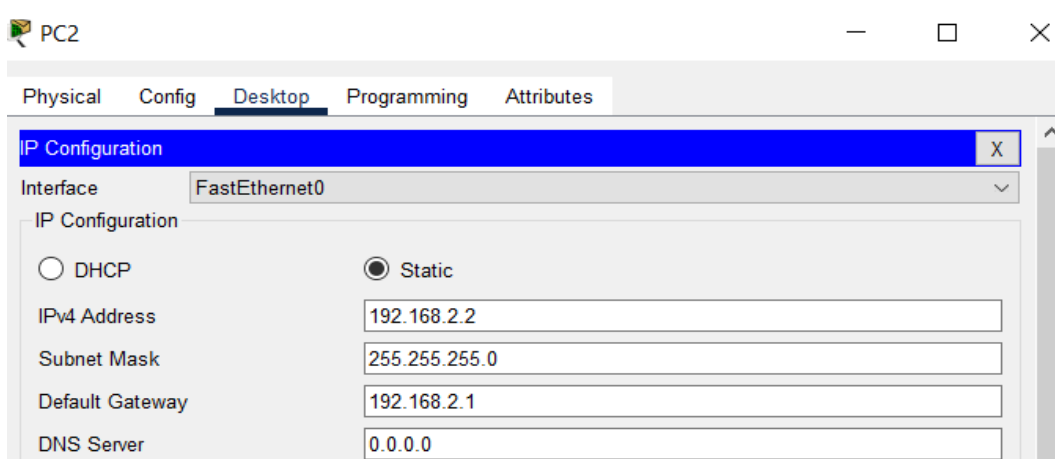
PC1:



The screenshot shows the configuration window for PC1. The 'Desktop' tab is selected. The 'IP Configuration' section is expanded, showing the 'FastEthernet0' interface. The 'Static' radio button is selected for the IP configuration. The fields are filled with the following values:

Field	Value
Interface	FastEthernet0
IP Configuration	Static
IPv4 Address	192.168.1.3
Subnet Mask	255.255.255.0
Default Gateway	192.168.1.1
DNS Server	0.0.0.0

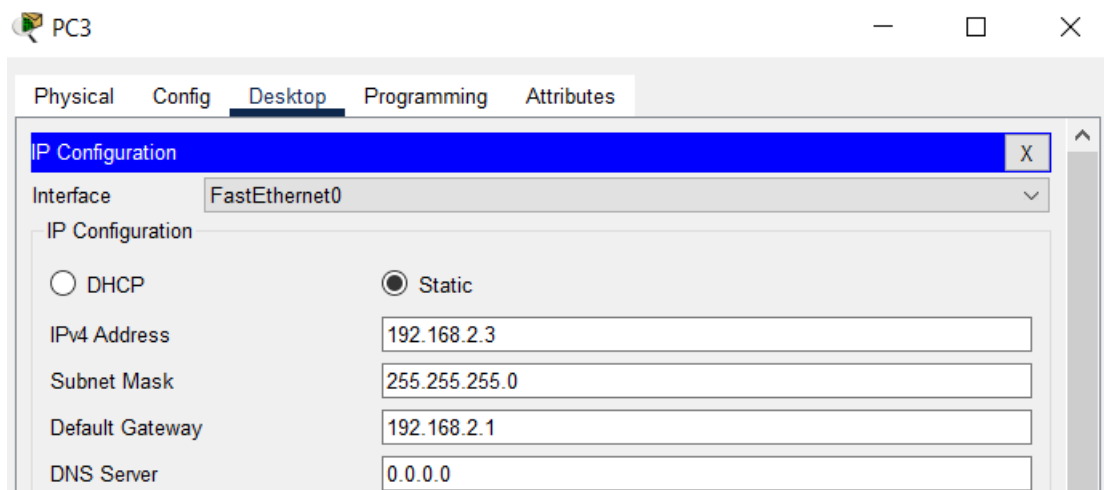
PC2:



The screenshot shows the configuration window for PC2. The 'Desktop' tab is selected. The 'IP Configuration' section is expanded, showing the 'FastEthernet0' interface. The 'Static' radio button is selected for the IP configuration. The fields are filled with the following values:

Field	Value
Interface	FastEthernet0
IP Configuration	Static
IPv4 Address	192.168.2.2
Subnet Mask	255.255.255.0
Default Gateway	192.168.2.1
DNS Server	0.0.0.0

PC3:



PC3

Physical Config **Desktop** Programming Attributes

IP Configuration X

Interface FastEthernet0

IP Configuration

☐ DHCP ☒ Static

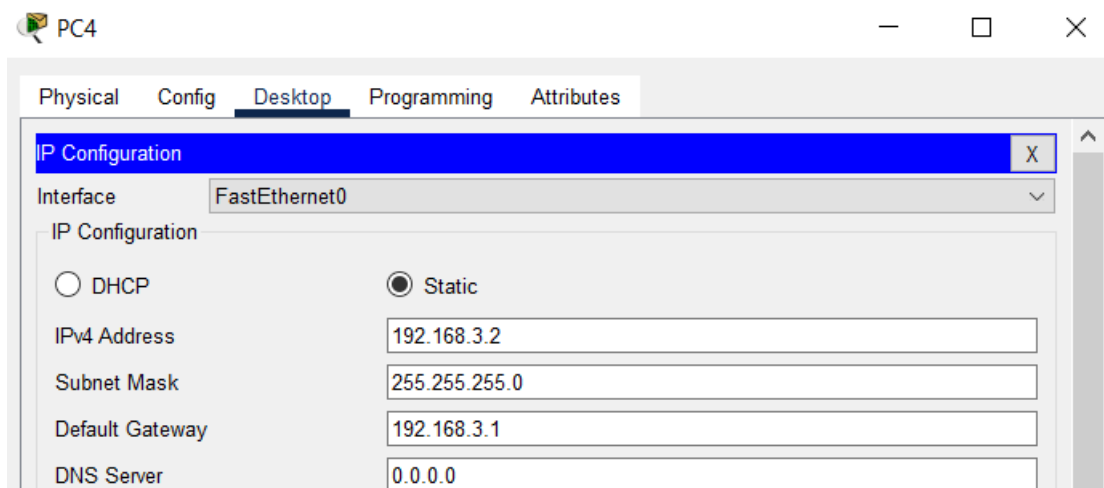
IPv4 Address 192.168.2.3

Subnet Mask 255.255.255.0

Default Gateway 192.168.2.1

DNS Server 0.0.0.0

PC4:



PC4

Physical Config **Desktop** Programming Attributes

IP Configuration X

Interface FastEthernet0

IP Configuration

☐ DHCP ☒ Static

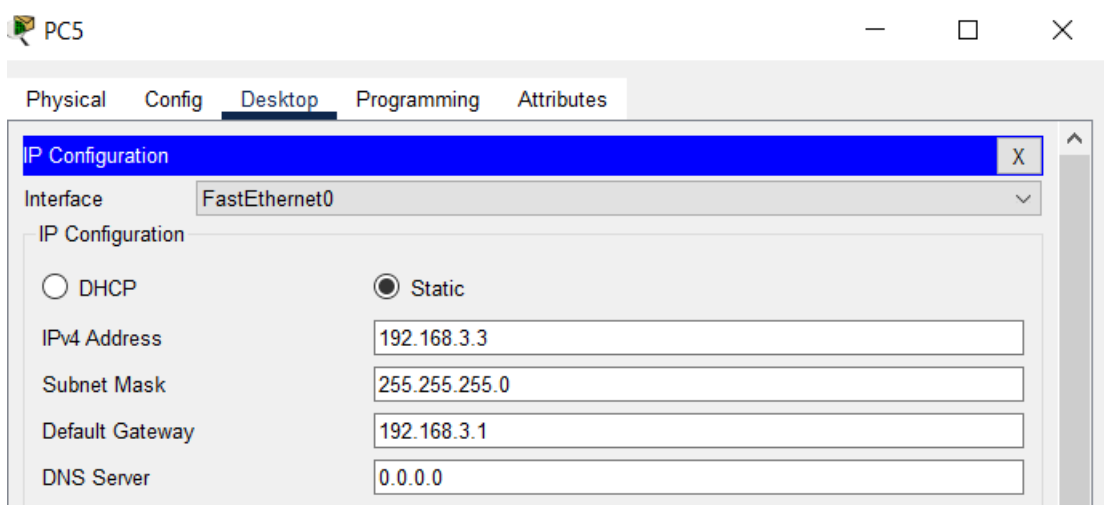
IPv4 Address 192.168.3.2

Subnet Mask 255.255.255.0

Default Gateway 192.168.3.1

DNS Server 0.0.0.0

PC5:



PC5

Physical Config **Desktop** Programming Attributes

IP Configuration X

Interface FastEthernet0

IP Configuration

☐ DHCP ☒ Static

IPv4 Address 192.168.3.3

Subnet Mask 255.255.255.0

Default Gateway 192.168.3.1

DNS Server 0.0.0.0

Router Configurations:

Router 0:

Router0

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

RIP Routing

Network

Add

Network Address
10.0.0.0
11.0.0.0
12.0.0.0
192.168.1.0
192.168.2.0
192.168.3.0

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

FastEthernet0/0

Port Status ☒ On

Bandwidth ☒ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 000B.BED4.E501

IP Configuration

IPv4 Address 192.168.1.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

Serial0/0/0

Port Status ☒ On

Duplex ☒ Full Duplex

Clock Rate 2000000

IP Configuration

IPv4 Address 10.0.0.1

Subnet Mask 255.0.0.0

Tx Ring Limit 10

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

Serial0/0/1

Port Status ☒ On

Duplex ☐ Full Duplex

Clock Rate 2000000

IP Configuration

IPv4 Address 12.0.0.1

Subnet Mask 255.0.0.0

Tx Ring Limit 10

Router 1:

Router1

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

RIP Routing

Network

Add

Network Address

10.0.0.0

11.0.0.0

12.0.0.0

192.168.1.0

192.168.2.0

192.168.3.0

Remove

Physical **Config** CLI Attributes

GLOBAL

Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/0/1

FastEthernet0/0

Port Status ☒ On

Bandwidth ☐ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address 00D0.FF08.5C01

IP Configuration

IPv4 Address 192.168.2.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Physical **Config** CLI Attributes

GLOBAL
 Settings
 Algorithm Settings
ROUTING
 Static
 RIP
SWITCHING
 VLAN Database
INTERFACE
 FastEthernet0/0
 FastEthernet0/1
Serial0/0/0
 Serial0/0/1

Serial0/0/0

Port Status ☒ On

Duplex ☐ Full Duplex

Clock Rate 2000000 ▾

IP Configuration

IPv4 Address 10.0.0.2

Subnet Mask 255.0.0.0

Tx Ring Limit 10

Physical **Config** CLI Attributes

GLOBAL
 Settings
 Algorithm Settings
ROUTING
 Static
 RIP
SWITCHING
 VLAN Database
INTERFACE
 FastEthernet0/0
 FastEthernet0/1
 Serial0/0/0
Serial0/0/1

Serial0/0/1

Port Status ☒ On

Duplex ☐ Full Duplex

Clock Rate 2000000 ▾

IP Configuration

IPv4 Address 11.0.0.1

Subnet Mask 255.0.0.0

Tx Ring Limit 10

Router 2:

Router2
 — □ ×

Physical **Config** CLI Attributes

GLOBAL
 Settings
 Algorithm Settings
ROUTING
 Static
RIP
SWITCHING
 VLAN Database
INTERFACE
 FastEthernet0/0
 FastEthernet0/1
 Serial0/0/0
 Serial0/0/1

RIP Routing

Network
Add

Network Address
10.0.0.0
11.0.0.0
12.0.0.0
192.168.1.0
192.168.2.0
192.168.3.0

Remove

Physical **Config** CLI Attributes

GLOBAL	FastEthernet0/0	
Settings		
Algorithm Settings		
ROUTING		
Static		
RIP		
SWITCHING		
VLAN Database		
INTERFACE		
FastEthernet0/0		
FastEthernet0/1		
Serial0/0/0		
Serial0/0/1		

Port Status ☒ On

Bandwidth ☒ 100 Mbps ☐ 10 Mbps ☒ Auto

Duplex ☐ Half Duplex ☒ Full Duplex ☒ Auto

MAC Address

IP Configuration

IPv4 Address

Subnet Mask

Tx Ring Limit

Physical **Config** CLI Attributes

GLOBAL	Serial0/0/0	
Settings		
Algorithm Settings		
ROUTING		
Static		
RIP		
SWITCHING		
VLAN Database		
INTERFACE		
FastEthernet0/0		
FastEthernet0/1		
Serial0/0/0		
Serial0/0/1		

Port Status ☒ On

Duplex ☒ Full Duplex

Clock Rate

IP Configuration

IPv4 Address

Subnet Mask

Tx Ring Limit

Physical **Config** CLI Attributes

GLOBAL	Serial0/0/1	
Settings		
Algorithm Settings		
ROUTING		
Static		
RIP		
SWITCHING		
VLAN Database		
INTERFACE		
FastEthernet0/0		
FastEthernet0/1		
Serial0/0/0		
Serial0/0/1		

Port Status ☒ On

Duplex ☒ Full Duplex

Clock Rate

IP Configuration

IPv4 Address

Subnet Mask

Tx Ring Limit

CHAPTER 3

RESULT AND DISCUSSION

3.1. Implementation Output:

Command Prompt:

```
Command Prompt
Microsoft Windows [Version 10.0.19045.2251]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Shyam Sundar>set path=C:\Program Files\Java\jdk1.8.0_301\bin

C:\Users\Shyam Sundar>cd D:\Projects\Distance_Vector_Implementation\Source_Code

C:\Users\Shyam Sundar>D:

D:\Projects\Distance_Vector_Implementation\Source_Code>javac *.java

D:\Projects\Distance_Vector_Implementation\Source_Code>java DVR D:\Projects\Distance_Vector_Implementation\Source_Code\Data_
Incorrect Directoy Path! Enter Correct path!

D:\Projects\Distance_Vector_Implementation\Source_Code>java DVR D:\Projects\Distance_Vector_Implementation\Data_6
Initilize the Port Number for all 6 Routers
Enter Port No for Router: a
5645
Enter Port No for Router: b
7895
Enter Port No for Router: c
4568
Enter Port No for Router: d
12564
Enter Port No for Router: e
365478
Enter a valid Port Number Higher than 1024 and less than 65536
8965
Enter Port No for Router: f
6542
Distance Vector Algorithm Started


D:\Projects\Distance_Vector_Implementation\Source_Code>
```

Initial Cost:

```
C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router a is Working..!
> output number 1

Optimal path a-b : the next hop is b and the cost is 2.0
Optimal path a-c : the next hop is c and the cost is 5.0
Optimal path a-d : the next hop is d and the cost is 1.0
Optimal path a-e: no route found
Optimal path a-f: no route found
```


 C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router b is Working..!

> output number 1


Optimal path b-a : the next hop is a and the cost is 2.0

Optimal path b-c : the next hop is c and the cost is 3.0

Optimal path b-d : the next hop is d and the cost is 2.0

Optimal path b-e: no route found

Optimal path b-f: no route found

 C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router c is Working..!

> output number 1


Optimal path c-a : the next hop is a and the cost is 5.0

Optimal path c-b : the next hop is b and the cost is 3.0

Optimal path c-d : the next hop is d and the cost is 3.0

Optimal path c-e : the next hop is e and the cost is 1.0

Optimal path c-f : the next hop is f and the cost is 1.0

 C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router d is Working..!

> output number 1


Optimal path d-a : the next hop is a and the cost is 1.0

Optimal path d-b : the next hop is b and the cost is 2.0

Optimal path d-c : the next hop is c and the cost is 3.0

Optimal path d-e : the next hop is e and the cost is 1.0

Optimal path d-f: no route found

 C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router e is Working..!

> output number 1

Optimal path e-a: no route found

Optimal path e-b: no route found

Optimal path e-c : the next hop is c and the cost is 1.0

Optimal path e-d : the next hop is d and the cost is 1.0

Optimal path e-f : the next hop is f and the cost is 2.0



C:\Program Files\Java\jdk1.8.0_301\bin\java.exe

Router f is Working..!

> output number 1

Optimal path f-a: no route found

Optimal path f-b: no route found

Optimal path f-c : the next hop is c and the cost is 1.0

Optimal path f-d: no route found

Optimal path f-e : the next hop is e and the cost is 2.0

Final Cost:

> output number 5

Optimal path a-b : the next hop is b and the cost is 2.0

Optimal path a-c : the next hop is d and the cost is 3.0

Optimal path a-d : the next hop is d and the cost is 1.0

Optimal path a-e : the next hop is d and the cost is 2.0

Optimal path a-f : the next hop is d and the cost is 4.0

> output number 5

Optimal path b-a : the next hop is a and the cost is 2.0

Optimal path b-c : the next hop is c and the cost is 3.0

Optimal path b-d : the next hop is d and the cost is 2.0

Optimal path b-e : the next hop is d and the cost is 3.0

Optimal path b-f : the next hop is c and the cost is 4.0

> output number 5

Optimal path c-a : the next hop is e and the cost is 3.0

Optimal path c-b : the next hop is b and the cost is 3.0

Optimal path c-d : the next hop is e and the cost is 2.0

Optimal path c-e : the next hop is e and the cost is 1.0

Optimal path c-f : the next hop is f and the cost is 1.0

```
> output number 5
```

```
Optimal path d-a : the next hop is a and the cost is 1.0  
Optimal path d-b : the next hop is b and the cost is 2.0  
Optimal path d-c : the next hop is e and the cost is 2.0  
Optimal path d-e : the next hop is e and the cost is 1.0  
Optimal path d-f : the next hop is e and the cost is 3.0
```

```
> output number 5
```

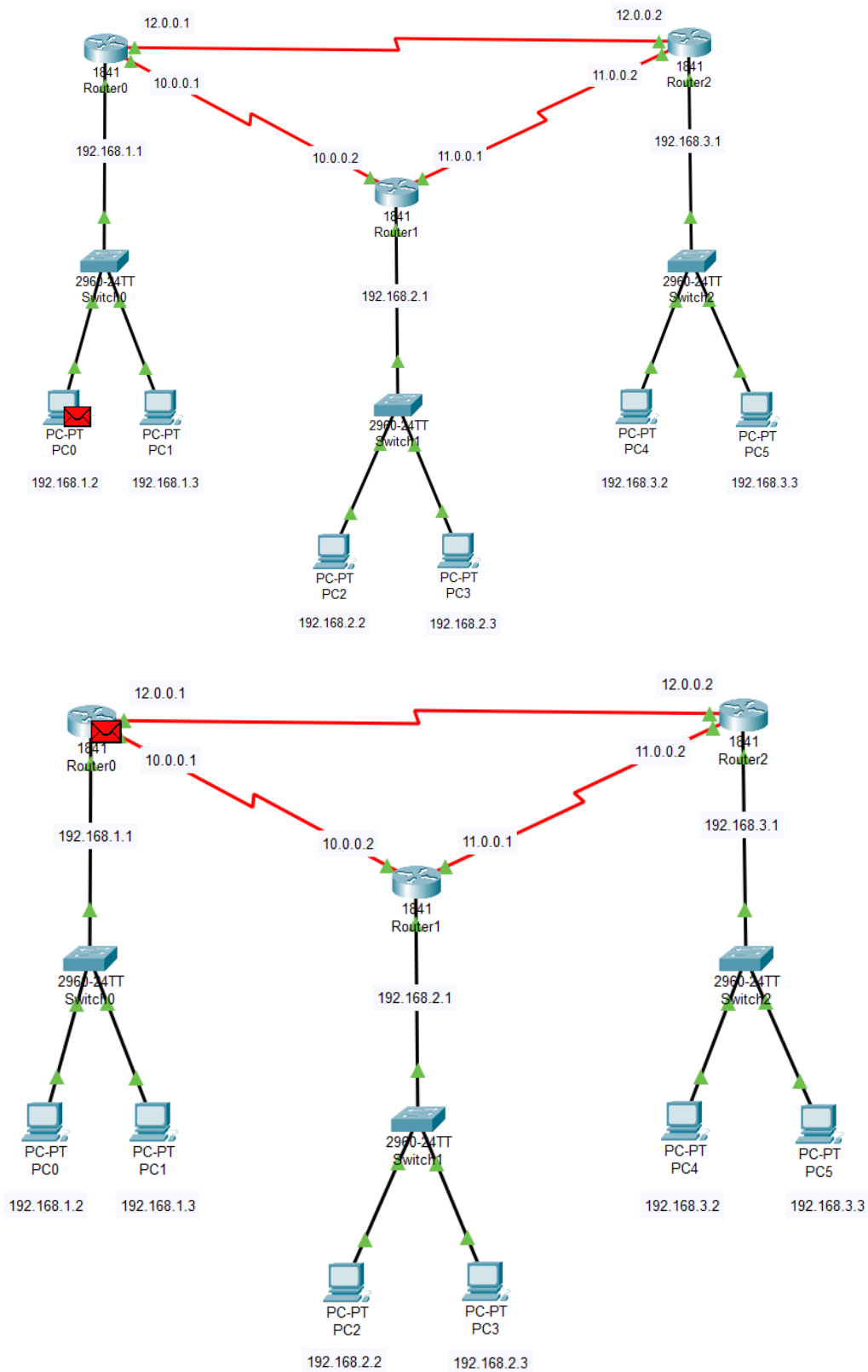
```
Optimal path e-a : the next hop is d and the cost is 2.0  
Optimal path e-b : the next hop is d and the cost is 3.0  
Optimal path e-c : the next hop is c and the cost is 1.0  
Optimal path e-d : the next hop is d and the cost is 1.0  
Optimal path e-f : the next hop is c and the cost is 2.0
```

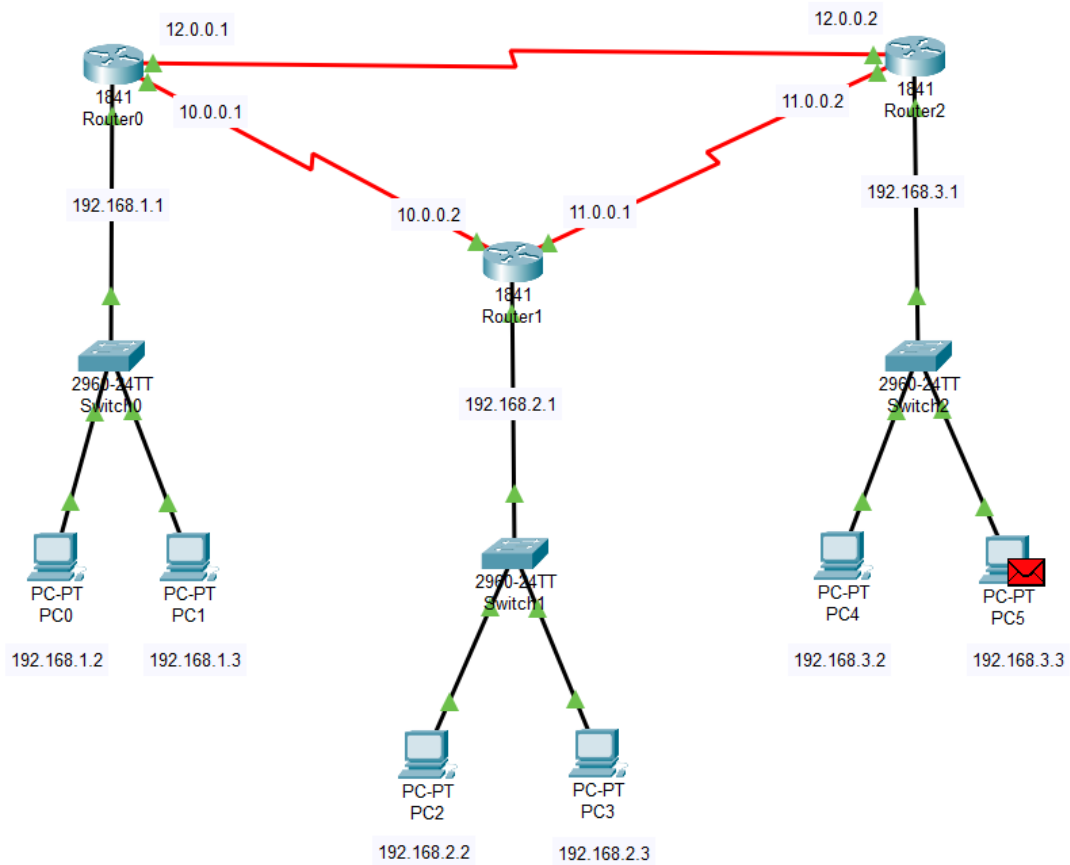
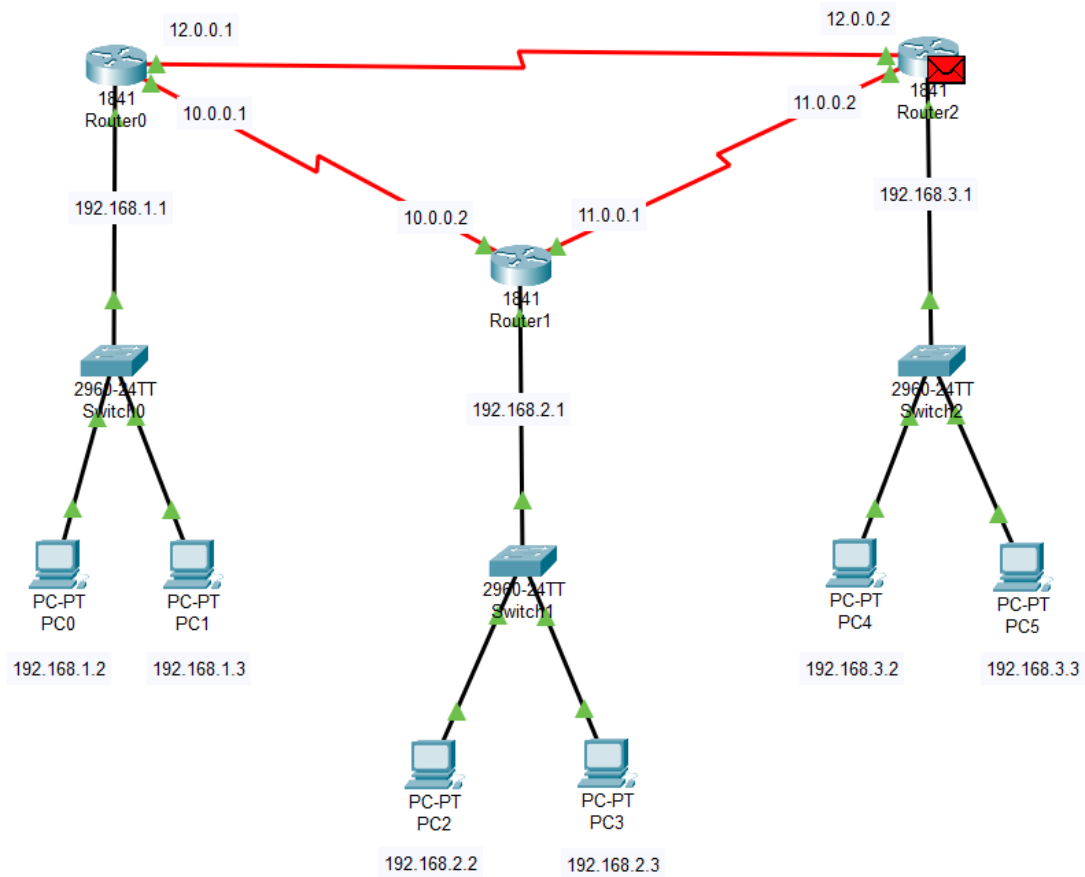
```
> output number 5
```

```
Optimal path f-a : the next hop is c and the cost is 4.0  
Optimal path f-b : the next hop is c and the cost is 4.0  
Optimal path f-c : the next hop is c and the cost is 1.0  
Optimal path f-d : the next hop is c and the cost is 3.0  
Optimal path f-e : the next hop is c and the cost is 2.0
```

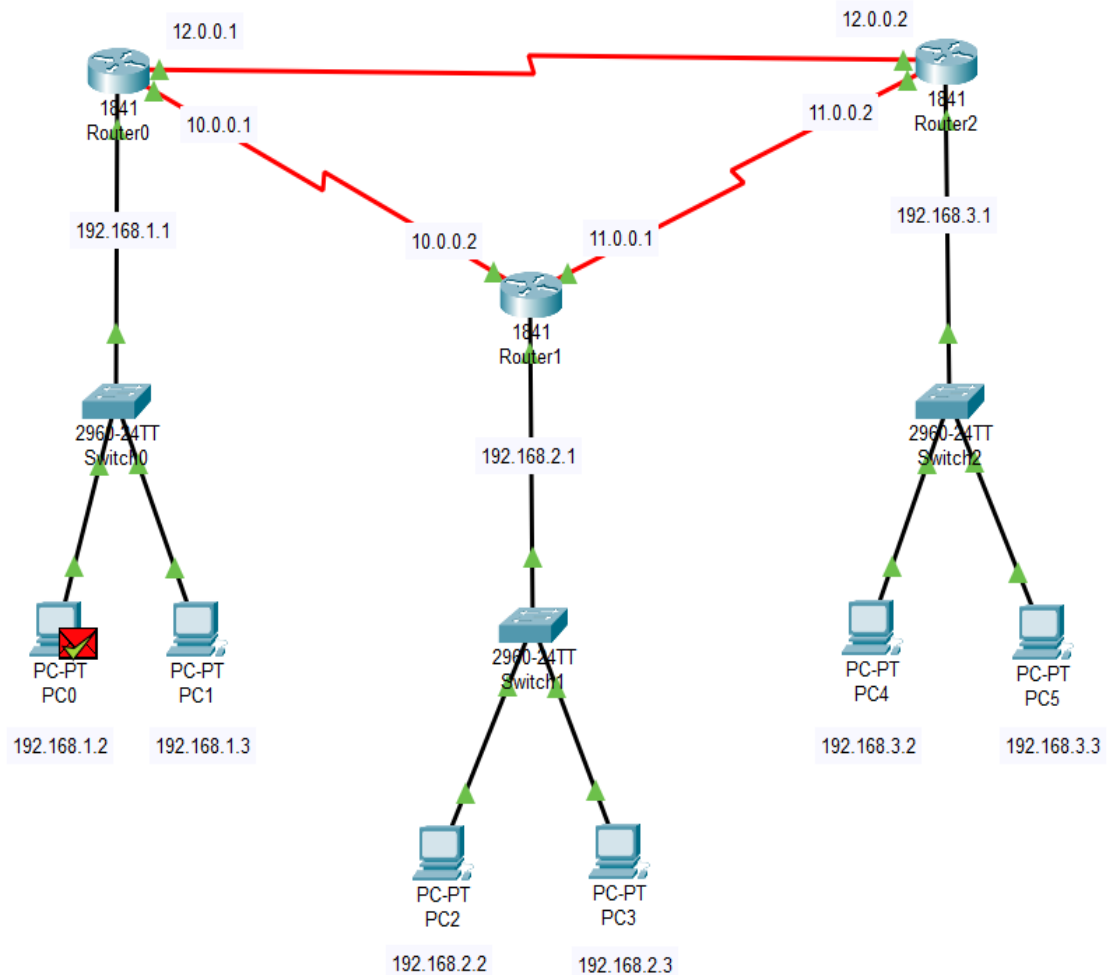
Initially, the route was unknown for some of the node pairs and the cost was higher in some of the node pairs. But with the Distance Vector Routing Protocol, we were able to find the optimal path from one node to all the other nodes. Since we use Bellman-Ford algorithm the output gives us single source shortest path unlike Dijkstra's algorithm where we get all source shortest path. The number of iterations that need to happen to determine the outcome is $(n-1)$ iterations, where n is the number of edges. In our example, we have used 6 edges so we iterate 5 times.

3.2. Simulation Output:





Again, the received packet will be re-transmitted to the sender along the same optimal path i.e., PC5 will re-transmit the packet to PC0 along the same path in which it was transmitted. Once the packet is received back, it will be acknowledged with a tick mark.



Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
	Successful	PC0	PC5	ICMP		0.000	N	0	(edit)	

Fig 3.1 – Packet Acknowledgment

Simulation Panel		
Event List		
Vis.	Time(sec)	Last Device
	0.000	--
	0.001	PC0
	0.002	Switch0
	0.003	Router0
	0.004	Router2
	0.005	Switch2
	0.006	PC5
	0.007	Switch2
	0.008	Router2
	0.009	Router0
Visible	0.010	Switch0

Fig 3.2 – Simulation Event List

Here the transmission of packets could have gone through Router-1, but the path was not taken as it is not the optimal path. The number of hops in the path via Router-1 is more than the number of hops in the path without Router-1. That is why the path was ignored and the packet was directly sent to Router-2 bypassing Router-1.

CHAPTER 4

CONCLUSION AND FUTURE PLANS

Distance Vector Routing algorithm is a very simple and effective algorithm. It does not require that all its nodes operate in the lock step with each other. It is better than Static Routing algorithms where we must mention the path manually. DVRP is very easy to implement and is very simple to debug. It allows us to compute the optimal path even without the knowledge of full network topography.

Enhanced and Advanced versions of Distance Vector Routing Protocols are being used in the current scenario. Some of the protocols that work based on DVRP are RIP, IGRP, BGP. The original ARPANET was also based on Distance Vector Routing Protocol. With proper knowledge about the Distance Vector Routing protocol, we can implement and simulate the above algorithms in the future.

CHAPTER 5

REFERENCES

James F. Kurose, and Keith W. Ross. Computer Networking: A Top-down Approach. Pearson Education, Seventh Edition, 2017.

www.geeksforgeeks.org

www.javatpoint.com

www.tutorialspoint.com

www.scaler.com