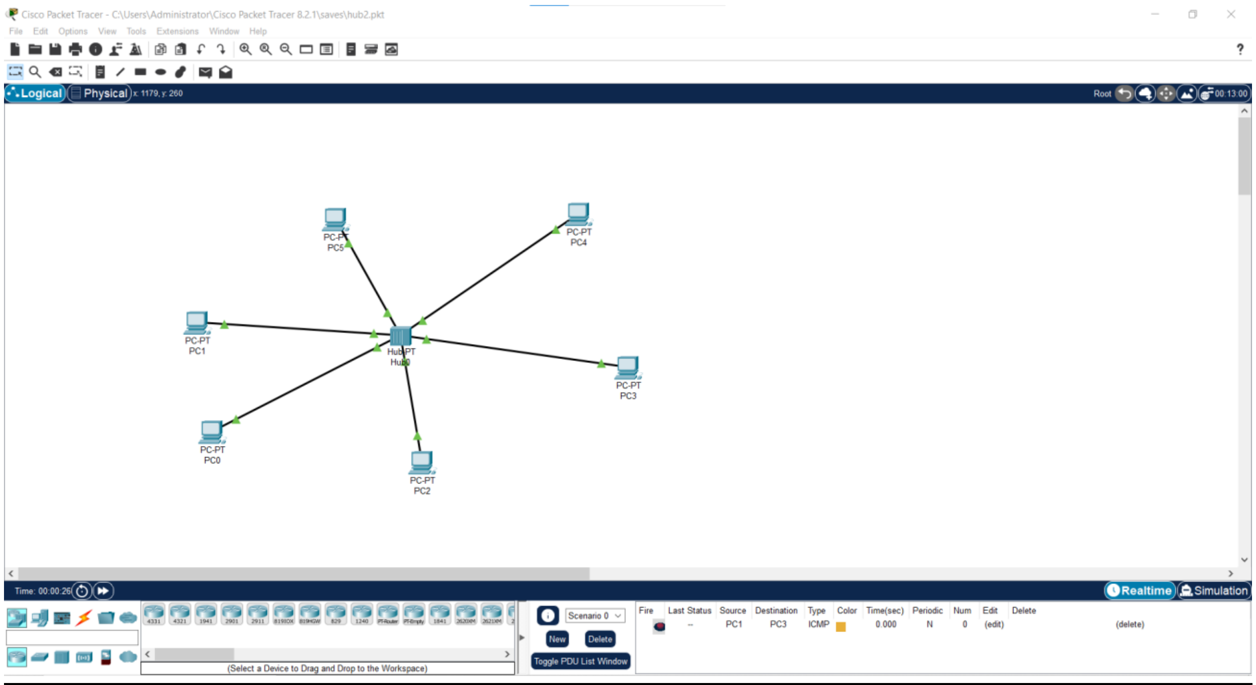
HUB


Cisco Packet Tracer - C:\Users\Administrator\Cisco Packet Tracer 8.2.1\saves\hub2.pkt
File Edit Options View Tools Extensions Window Help
Logical
Physical
x: 1179, y: 260
Root
00:13:00
PC-PT PC5
PC-PT PC4
PC-PT PC1
Hub-PT Hub0
PC-PT PC3
PC-PT PC0
PC-PT PC2
Time: 00:00:26
Realtime
Simulation
(Select a Device to Drag and Drop to the Workspace)
Scenario 0
New
Delete
Toggle PDU List Window

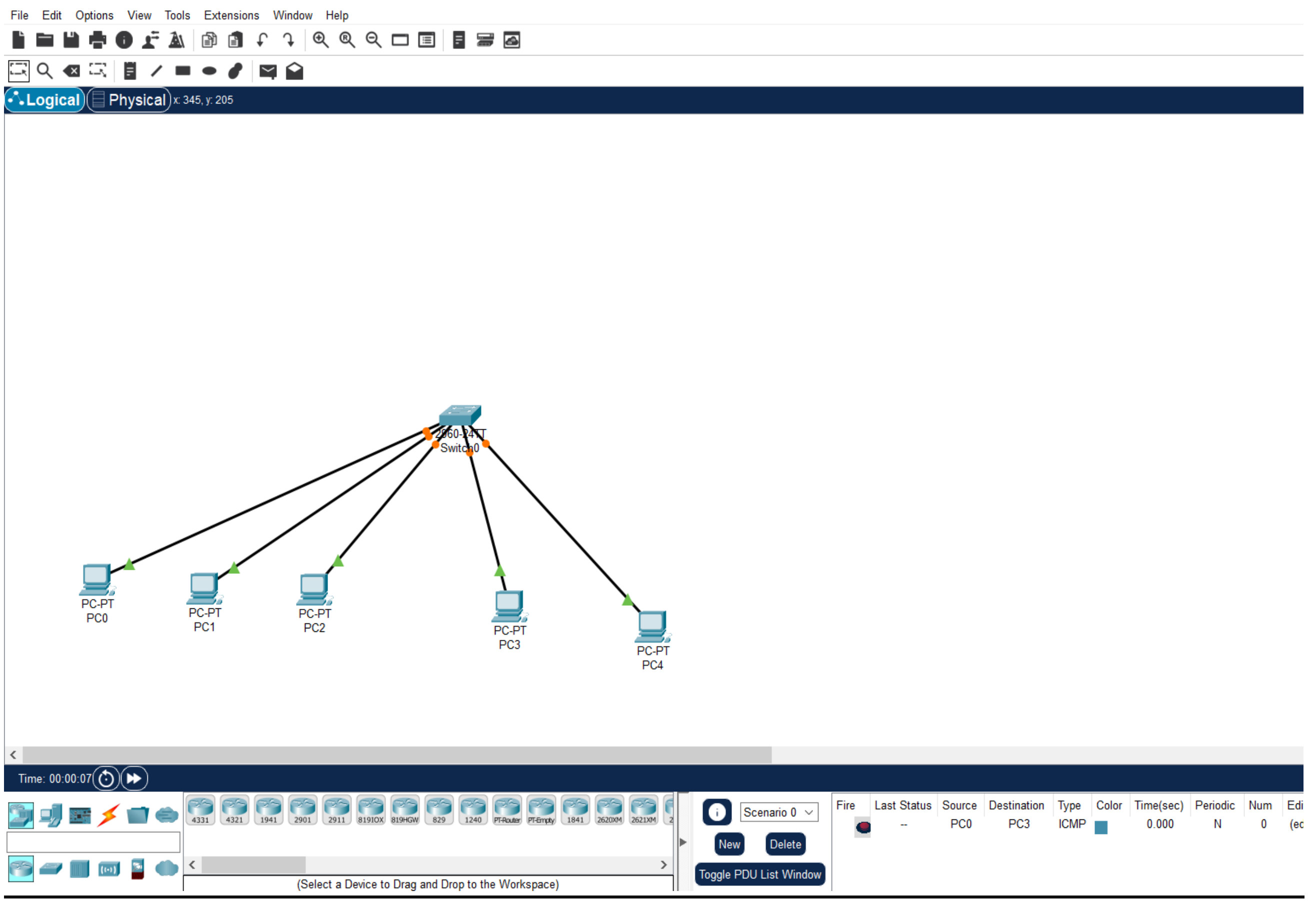
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	--	PC1	PC3	ICMP	🟧	0.000	N	0	(edit)	(delete)

A **hub** is a networking device that functions as a central connection point for devices in a network. It operates at the physical layer of the OSI model. Hubs are often used to connect segments of a Local Area Network (LAN). The primary role of a hub is to transmit data packets to all devices connected to its ports, without any filtering or data routing capabilities. This means that when a data packet arrives at one port, it is copied to all other ports, so every segment of the LAN can see all packets.

There are three main types of hubs:

- **Active Hub**: These have their own power supply and can regenerate and relay the signal along with the network. They are used to extend the maximum distance between nodes.
- **Passive Hub**: These collect wiring from nodes and rely on an active hub for the power supply. They cannot regenerate signals and therefore cannot extend the distance between nodes.
- **Intelligent Hub**: These function like active hubs but also include remote management capabilities, allowing network administrators to monitor traffic and configure each port in the hub

SWITCH


File Edit Options View Tools Extensions Window Help
Logical
Physical
x: 345, y: 205
2960-24TT
Switch0
PC-PT PC0
PC-PT PC1
PC-PT PC2
PC-PT PC3
PC-PT PC4
Time: 00:00:07
4331 4321 1941 2901 2911 819IOX 819HGW 829 1240 PT-Router PT-Empty 1841 2620XM 2621XM
(Select a Device to Drag and Drop to the Workspace)
Scenario 0
New
Delete
Toggle PDU List Window

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edi
●	--	PC0	PC3	ICMP	🟦	0.000	N	0	(ec

A **network switch** is a fundamental device in computer networking that operates at the **Data Link Layer (Layer 2)** of the OSI model. It connects multiple devices within a network, such as computers, printers, and servers, into a single LAN (Local Area Network). Here's a brief overview:

Functionality:

- **Filters and Forwards Data**: Switches are responsible for filtering and forwarding packets between LAN segments based on the MAC address.
- **Error Checking**: Before forwarding data, switches perform error checking to ensure data integrity.
- **Full Duplex Communication**: Operates in full duplex mode, allowing simultaneous bidirectional communication, which reduces the chance of data collisions.

Types of Switches:

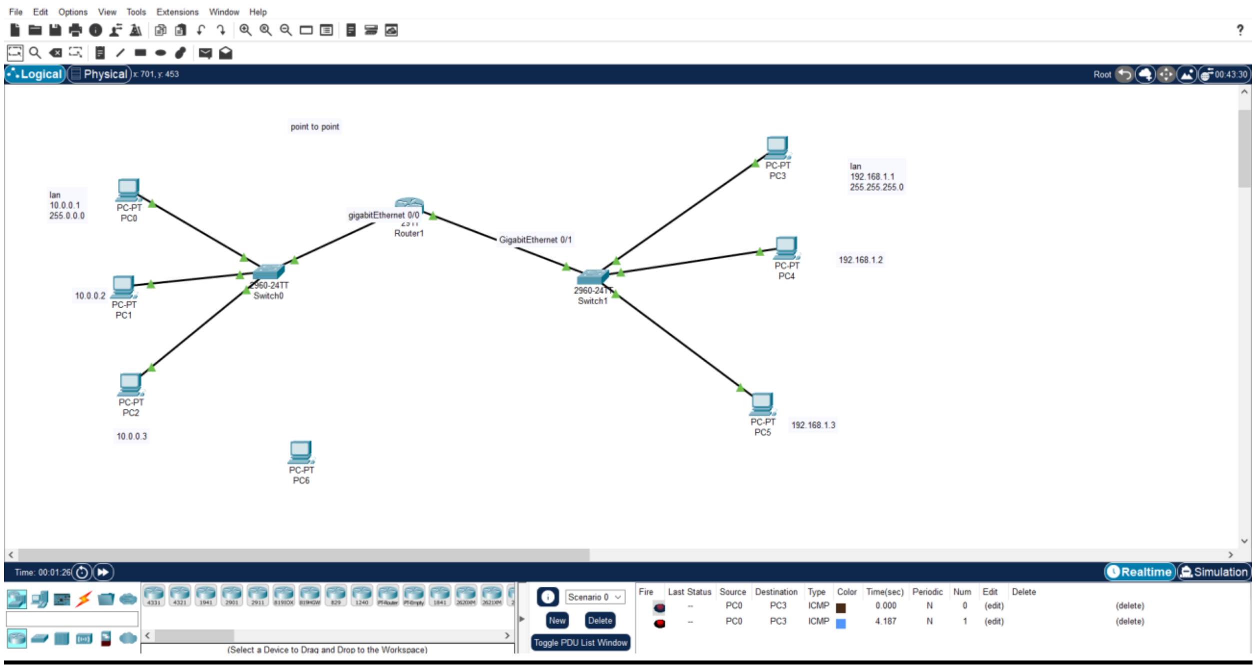
- **Unmanaged Switches**: Simple plug-and-play devices used for basic network connectivity, often in home networks or small businesses.

- **Managed Switches**: Offer advanced features like VLANs, QoS, and network management, suitable for larger organizations with complex networks.
- **LAN Switches**: Specifically designed to reduce network congestion by efficiently allocating bandwidth.
- **PoE Switches**: Power over Ethernet switches that can transmit both power and data over a single network cable, useful for powering devices like IP cameras and wireless access points.

Benefits:

- **Efficient Bandwidth Usage**: By operating in full duplex, switches make effective use of available bandwidth.
- **Segmentation**: Helps in segmenting the network into subnets, which can improve performance and security.
- **Smart Environment Integration**: Facilitates the connection of IoT devices, contributing to smarter surroundings through data sharing.

ROUTER

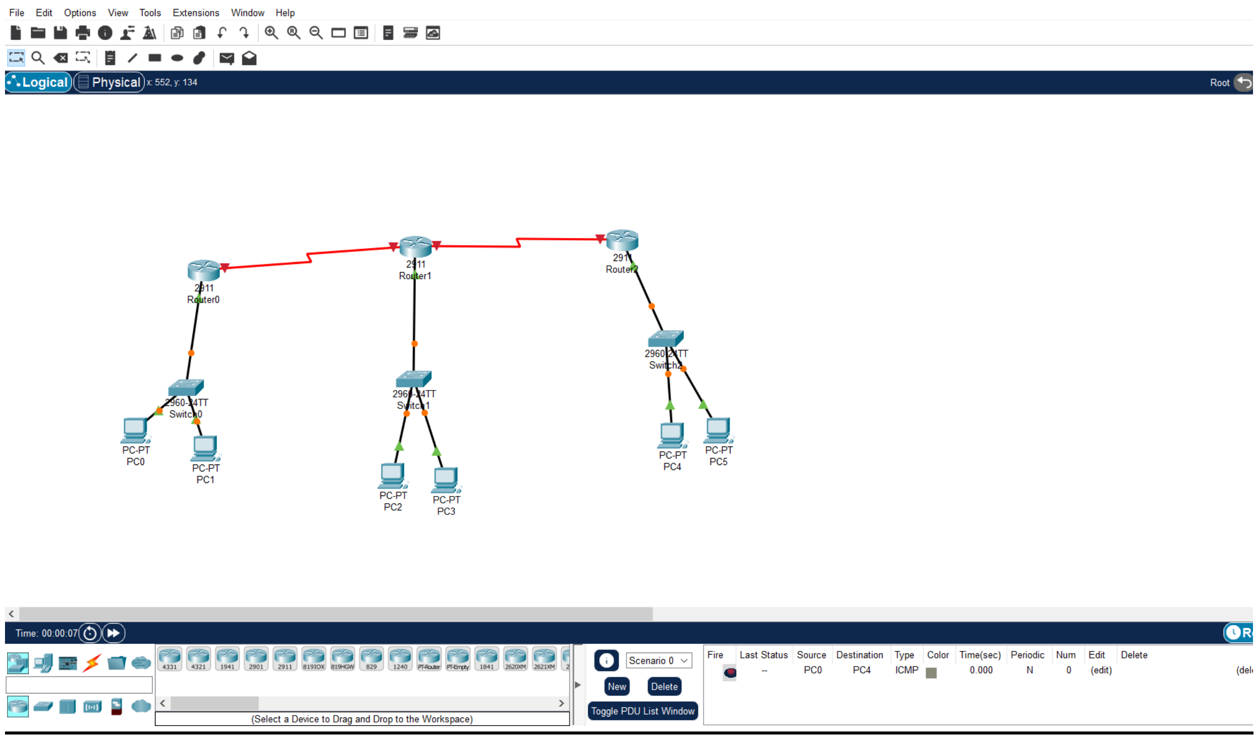


Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	--	PC0	PC3	ICMP	🟫	0.000	N	0	(edit)	(delete)
●	--	PC0	PC3	ICMP	🟦	4.187	N	1	(edit)	(delete)

A router is a critical device in a computer network that directs data packets between different networks. It operates on the Network Layer of the OSI Model and uses routing tables to determine the best path for data transmission. Here's a brief overview:

- **Function**: Routers forward data packets between computer networks, managing traffic and enabling multiple devices to share an internet connection.
- **Operation**: They analyze the destination IP address in data packets and use routing tables to send the data along the most efficient path.
- **Types**: There are various types of routers, including broadband, wireless, and wired routers, each serving different networking needs.
- **Dynamic and Static Routing**: Routers can use dynamic routing tables that update based on network activity or static tables set up manually.
- **Network Layer Device**: As network layer devices, routers are responsible for packet forwarding based on IP addresses.

ROUTING INFORMATION PROTOCOL

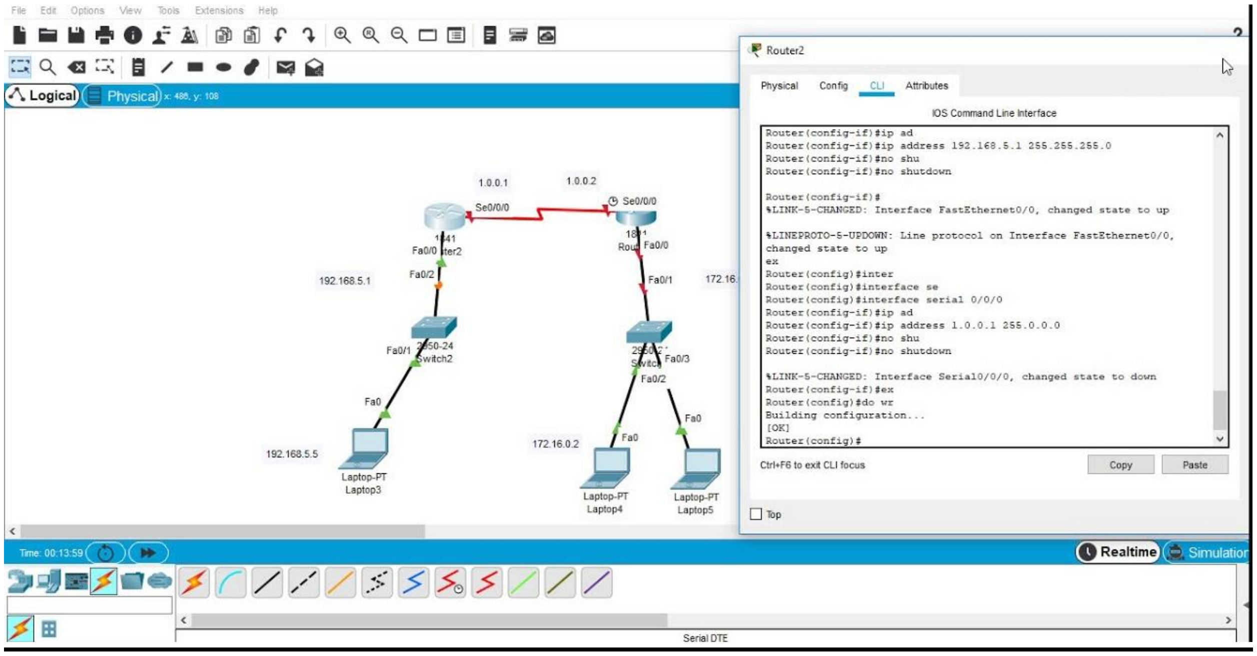


Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete
●	--	PC0	PC4	ICMP	■	0.000	N	0	(edit)	(del

Routing Information Protocol (RIP) version 1 is a dynamic routing protocol used in computer networks. It employs a distance-vector routing algorithm and uses hop count as the routing metric to determine the best path between the source and the destination network. Here are some key points about RIP version 1:

- **Hop Count**: This is the number of routers that a packet must pass through to reach its destination. The path with the lowest hop count is considered the best route and is placed in the routing table.
- **Maximum Hop Count**: RIP version 1 limits the number of hops to 15, and a hop count of 16 is considered unreachable.
- **Periodic Updates**: RIP version 1 routers exchange their entire routing tables with adjacent routers every 30 seconds.
- **Broadcast Updates**: The updates are broadcasted to all devices on the network.
- **Classful Protocol**: RIP version 1 does not support subnetting or supernetting as it is a classful protocol, meaning it doesn't send subnet mask information in its routing updates.

ROUTING INFORMATION PROTOCOL



```
Router(config-if)#ip ad
Router(config-if)#ip address 192.168.5.1 255.255.255.0
Router(config-if)#no shu
Router(config-if)#no shutdown

Router(config-if)#
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0,
changed state to up
ex
Router(config)#inter
Router(config)#interface se
Router(config)#interface serial 0/0/0
Router(config-if)#ip ad
Router(config-if)#ip address 1.0.0.1 255.0.0.0
Router(config-if)#no shu
Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial0/0/0, changed state to down
Router(config-if)#ex
Router(config)#do wr
Building configuration...
[OK]
Router(config)#
```

Routing Information Protocol version 2 (RIP v2) is an enhancement over its predecessor, RIP v1, which was a distance-vector routing protocol used in local and wide area networks. As a classless routing protocol, RIP v2 supports Variable Length Subnet Masking (VLSM), allowing for more efficient use of IP addresses. Unlike RIP v1, which only supported classful routing, RIP v2 can carry subnet information, enabling more flexible network design and addressing.

Key features of RIP v2 include:

- **Use of multicast**: RIP v2 sends updates using multicast address 224.0.0.9, which reduces unnecessary traffic compared to RIP v1's broadcast updates.
- **Support for authentication**: It can authenticate update messages to enhance security.
- **Route tagging**: RIP v2 allows routes to be tagged, which helps in distinguishing between different types of routes.
- **Triggered updates**: It sends updates more efficiently by triggering them only when there are changes in the network topology, rather than at regular intervals.

RIP v2 is suitable for small to medium-sized networks due to its simplicity and ease of configuration. However, it has limitations, such as a maximum hop count of 15, which restricts its use in larger networks. Additionally, it's less preferred in modern networks due to the availability of more advanced routing protocols that offer better scalability and performance.

Write a program for error detecting code using CRC-CCITT (16- bits)

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check.

To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The CRC calculation or cyclic redundancy check was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it.

Algorithm:- 1. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.

2. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.

3. Define T(x) = B(x) R(x) (T(x)/G(x) => remainder 0)

4. Transmit T, the bit string corresponding to T(x). error occurred otherwise, the receiver concludes an error occurred and requires a retransmission

PROGRAM:

```
/* CRC */
import java.util.*;
public class Crc
{
 void div(int a[],int k)
{
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
int count=0;
for(int i=0;i<k;i++)
{
```

```
if(a[i]==gp[0])
{
for(int j=i;j<17+i;j++)
{
a[j]=a[j]^gp[count++];
}
count=0;
}
}
}
public static void main(String args[])
{
int a[]=new int[100];
int b[]=new int[100];
int len,k;
Crc ob=new Crc();
System.out.println("Enter the length of Data Frame:");
Scanner sc=new Scanner(System.in);
len=sc.nextInt();
int flag=0;
System.out.println("Enter the Message:");
for(int i=0;i<len;i++)
{
a[i]=sc.nextInt();
}
for(int i=0;i<16;i++)
{
```

```
a[len++]=0;
}
k=len-16;
for(int i=0;i<len;i++)
{
b[i]=a[i];
}
ob.div(a,k);
for(int i=0;i<len;i++)
a[i]=a[i]^b[i];
System.out.println("Data to be transmitted: ");
for(int i=0;i<len;i++)
{
 System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("Enter the Reveived Data: ");
for(int i=0;i<len;i++)
{
a[i]=sc.nextInt();
}
ob.div(a, k);
for(int i=0;i<len;i++)
{
if(a[i]!=0)
{
flag=1; break;
}
}
if(flag==1)
System.out.println("error in data");
else
System.out.println("no error");
} }
```

Output1

```
Enter the length of Data Frame:
4
Enter the Message:
1 0 1 1
Data to be transmitted:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
Enter the Reveived Data:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
no error
```

Output2

```
Enter the length of Data Frame:
4
Enter the Message:
1 0 1 1
Data to be transmitted:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
Enter the Reveived Data:
1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1
 error in data
```

Write a program to find the shortest path between vertices using bellman-ford algorithm.

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Implementation Algorithm:

```
1. send my routing table to all my neighbors whenever my link table changes
2. when I get a routing table from a neighbor on port P with link metric M:
        a. add L to each of the neighbor's metrics
        b. for each entry (D, P', M') in the updated neighbor's table:
                i. if I do not have an entry for D, add (D, P, M') to my routing table
                ii. if I have an entry for D with metric M", add (D, P, M') to my routing
                table if M' < M"
3. if my routing table has changed, send all the new entries to all my neighbor
```

PROGRAM:

```
/* Bellman-Ford */
import java.util.*;
public class Belmanford
 {
 private int D[];
 private int n;
 public static final int max_value=999;
 public Belmanford(int n)
 {
 this.n=n;
 D=new int[n+1];
```

```
}
public void shortest(int s,int a[][])
{
for(int i=1;i<=n;i++)
{
D[i]=max_value;
}
D[s]=0;
for(int k=1;k<=n-1;k++)
{
for(int i=1;i<=n;i++)
{
for(int j=1;j<=n;j++)
{
if(a[i][j]!=max_value)
{
if(D[j]>D[i]+a[i][j])
D[j]=D[i]+a[i][j];
}
}
}
}
for (int i=1;i<=n;i++)
{
for (int j=1;j<=n;j++)
{
if(a[i][j]!=max_value)
```

```
 {
 if(D[j]>D[i]+a[i][j])
 {
 System.out.println("the graph contains -ve edge cycle");
 return;
 }
 }
 }
 }
 for (int i=1;i<=n;i++)
 {
 System.out.println("distance of source"+s+"to"+i+"is"+D[i]);
 }
 }
public static void main(String[] args)
{
 int n=0,s;
 Scanner sc=new Scanner(System.in);
 System.out.println("enter the no.of values");
 n=sc.nextInt();
 int a[][]=new int [n+1][n+1];
 System.out.println("enter the weighted matrix:");
 for (int i=1;i<=n;i++)
 {
 for (int j=1;j<=n;j++)
 {
 a[i][j]=sc.nextInt();
```

```
if(i==j)
{
a[i][j]=0;
continue;
}
if(a[i][j]==0)
a[i][j]=max_value;
}
}
System.out.println("enter the source vertex:");
s=sc.nextInt();
Belmanford b=new Belmanford(n);
b.shortest(s,a);
sc.close();
} }
```

Output1

```
enter the no.of values
4
enter the weighted matrix:
0 999 999 999
5 0 3 4
999 999 0 2
999 999 999 0
enter the source vertex:
2
distance of source 2 to1 is 5
distance of source 2 to 2 is 0
distance of source 2 to 3 is 3

distance of source 2 to 4 is 4

Output2:

enter the no.of values

4

enter the weighted matrix:

0 4 999 5

999 0 999 999

999 -10 0 999

999 999 3 0

enter the source vertex:

1

distance of source 1to 1 is 0

distance of source 1 to 2 is-2

distance of source 1 to 3 is 8

distance of source 1 to 4 is 5

Output3

enter the no.of values

4

enter the weighted matrix:

0 4 5 999

999 0 999 7

999 7 0 999

999 999 -15 0

enter the source vertex:

1

the graph contains -ve edge cycle
```

Write a program for congestion control using leaky bucket algorithm.

The leaky-bucket algorithm:

The algorithm can be conceptually understood as follows:

Consider a bucket with a hole in the bottom.

- The empty space of the bucket represents an amount of credit available measured in

  Bytes.
- The size of the bucket is b bytes. This means that if the bucket is empty, b bytes of

  credit is available.
- If a packet arrives and its size is less than the available credit, the packet can be forwarded. Otherwise, it is discarded or queued depending on the application.
- The bucket leaks through the hole in its bottom at a constant rate of r bytes per second, this indicates credit accumulation.

PROGRAM:

```
/* Leaky Bucket */
public class LeakyBucket
{
 static int min(int x,int y)
{
if(x<y)
return x;
else
return y;
}
```

```
public static void main(String[] args)
 {
 int drop=0,mini,nsec,cap,count=0,i,process;
int inp[]=new int[25];
Scanner sc=new Scanner(System.in);
System.out.println("Enter The Bucket Size\n");
cap= sc.nextInt();
System.out.println("Enter The Operation Rate\n");
process= sc.nextInt();
System.out.println("Enter The No. Of Seconds You Want To Stimulate\n");
nsec=sc.nextInt();
for(i=0;i<nsec;i++)
{
System.out.print("Enter The Size Of The Packet Entering At "+ i+1+"sec");
inp[i] = sc.nextInt();
}
System.out.println("\nSecond | Packet Recieved | Packet Sent | Packet Left | Packet
Dropped|\n");
//System.out.println("------------------------------------------------------------------\n");
for(i=0;i<nsec;i++)
{
count+=inp[i];
if(count>cap)
{
drop=count-cap;
 count=cap;
 }
System.out.print(i+1);
System.out.print("\t\t"+inp[i]);
mini=min(count,process);
System.out.print("\t\t"+mini);
count=count-mini;
System.out.print("\t\t"+count);
System.out.print("\t\t"+drop);
drop=0;
System.out.println();
}
for(;count!=0;i++)
{
if(count>cap)
{
drop=count-cap;
count=cap;
}
System.out.print(i+1);
System.out.print("\t\t0");
mini=min(count,process);
System.out.print("\t\t"+mini);
count=count-mini;
System.out.print("\t\t"+count);
System.out.print("\t\t"+drop);
System.out.println();
}
}
}
```

```
Output1

Enter The Bucket Size

6

Enter The output Rate

2

Enter The No. of Seconds You Want To Stimulate

2

Enter The Size of Packet entering at 01sec

8

Enter The Size of Packet entering at 11sec

6

Second | Packet Recieved | Packet Sent | Packet Left | Packet Dropped|

1 8 2 4 2

2 6 2 4 4

3 0 2 2 0

4 0 2 0 0

Output2

Enter The Bucket Size

5

Enter The output Rate

2

Enter The No. of Seconds You Want To Stimulate

3

Enter The Size of Packet entering at 01sec

5

Enter The Size of Packet entering at 11sec

4
```

```
Enter The Size of Packet entering at 21sec

3

Second | Packet Recieved | Packet Sent | Packet Left | Packet Dropped|

1                 5                    2                3        0

2                 4                    2                3        2

3                 3                    2                3        1

4                 0                    2                1        0

5                 0                    1                0        0
```

CONCLUSION

In this lab, we explored various aspects of computer networks. We learned about network topologies, protocols, and communication models. Through practical exercises, we configured routers, switches, and analyzed network traffic. Overall, this lab provided valuable insights into network design, troubleshooting, and security. Moving forward, understanding these concepts will be crucial for building robust and efficient networks.

THANK YOU