***Computer Network Experiments***

***Experiment – 1***

**Aim: -** To study and understand different networking devices and Topologies.

**Network Devices: -**

* The devices which are used for [communication](https://www.elprocus.com/satellite-communication-system/) between different hardware used in the computer network are known as network devices.
* These devices transfer data in a fast, secure, and correct way over the same or different networks.
* Network devices may be inter-network or intra-network.
* These devices are also known as physical devices, networking hardware, and network equipment otherwise computer networking devices. In a computer network, each network device plays a key role based on its functionality, and also works for different purposes at different segments.
* Some devices are installed on the device, like NIC card or RJ45 connector, whereas some are part of the network, like a router, or switch.

**Types of Network devices: -**

* **HUB**
* **Switch**
* **Gateway**
* **Router**
* **Bridge**
* **Repeater**

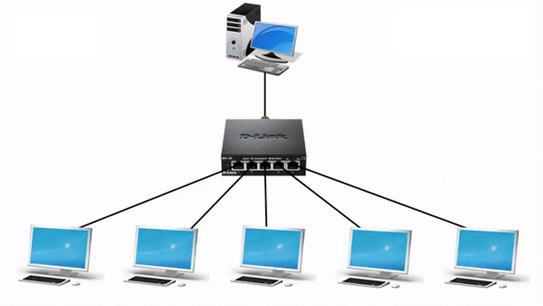
1. **HUB: -**
2. The transferring of data in a computer network can be done in the form

of packets. Whenever the [data processing](https://www.elprocus.com/data-processing-types-and-its-applications/) can be done from a host to a network hub, then the data can transmit to all the connected ports.

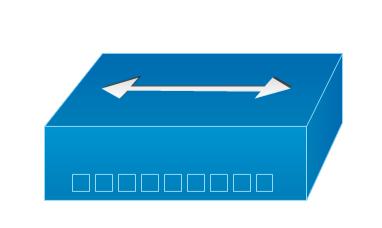
1. Similarly, all the ports identify the data path which leads to inefficiencies & wastage. Because of this working, a network hub cannot be so safe and secure.
2. Network hubs are classified into two types active hub & passive hub.

* **Active Hub: -** These are the hubs that have their own power supply and can clean, boost, and relay the signal along with the network. It serves both as a repeater as well as a wiring center. These are used to extend the maximum distance between nodes.
* **Passive Hub: -** These are the hubs that collect wiring from nodes and power supply from the active hub. These hubs relay signals onto the network without cleaning and boosting them and can’t be used to extend the distance between nodes

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

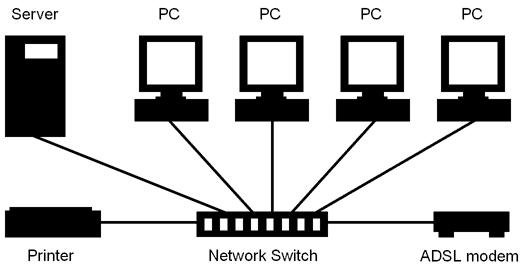
* **Advantages: -**
* It provides support for different types of Network Media.
* It can be used by anyone as it is very cheap.
* It can easily connect many different media types.
* The use of a hub does not impact on the network performance.
* Additionally, it can expand the total distance of the network.
* **Disadvantages: -**
* It has no ability to choose the best path of the network.
* It does not include mechanisms such as collision detection.
* It does not operate in full-duplex mode and cannot be divided into the Segment.
* It cannot reduce the network traffic as it has no mechanism.
* It is not able to filter the information as it transmits packets to all the connected segments.
* Furthermore, it is not capable of connecting various network architectures like a ring, token, and ethernet, and more.

1. **Switch: -**

o When a user accesses the internet or another computer network outside their immediate location, messages are sent through the network of transmission media. This technique of transferring the information from one computer network to another network is known as switching.

1. Network switches operate at Data link layer in the OSI model.
2. A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance.
3. The switch can perform error checking before forwarding data, which makes it very efficient as it does not forward packets that have errors and forward good packets selectively to the correct port only.

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* Switch increases the bandwidth of the network.
* It reduces the workload on individual PCs as it sends the information to only that device that has been addressed.
* It increases the overall performance of the network by reducing the traffic on the network.
* There will be less frame collision as the switch creates the collision domain for each connection.
* **Disadvantages: -**
* A Switch is more expensive than network bridges.
* A Switch cannot determine the network connectivity issues easily.
* Proper designing and configuration of the switch are required to handle multicast packets.

1. **Gateway: -**

o A gateway, as the name suggests, is a passage to connect two networks together that may work upon different networking models.

1. They basically work as the messenger agents that take data from one

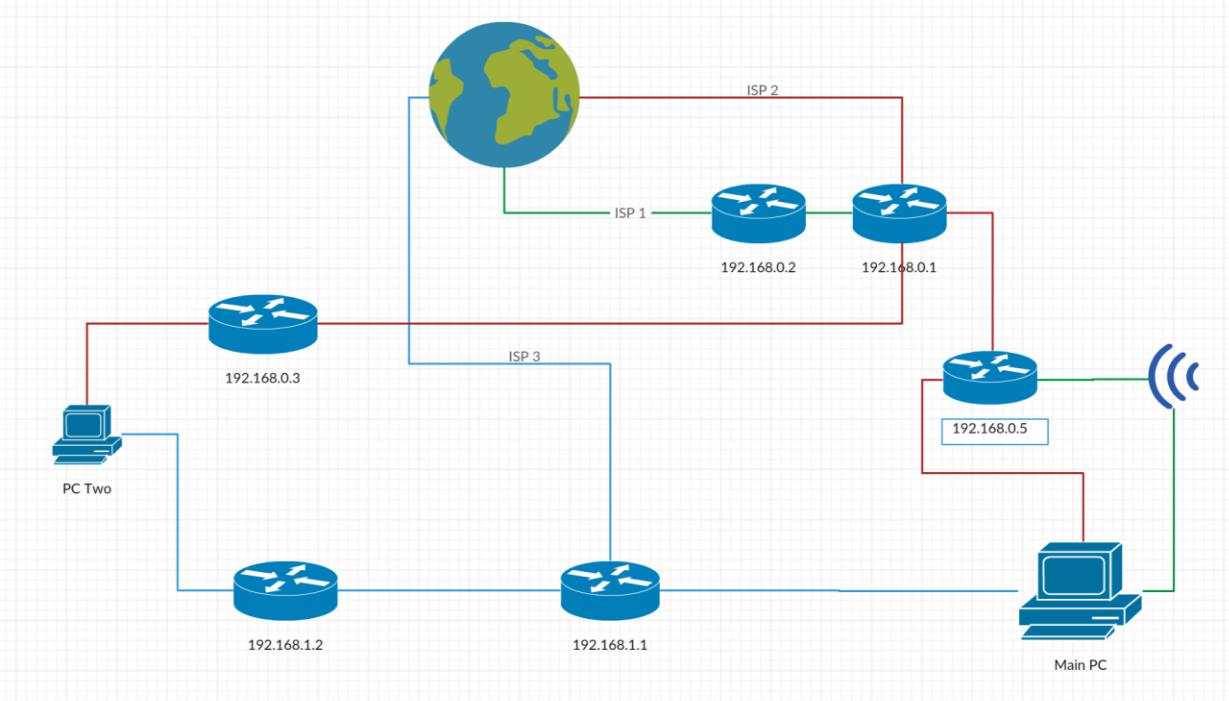
The system, interpret it, and transfer it to another system.

1. Gateways are also called protocol converters and can operate at any

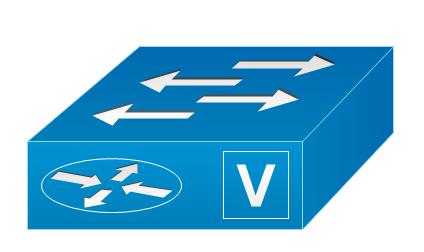
network layer.

1. Gateways are generally more complex than switches or routers. o Gateway is also called a protocol converter.

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

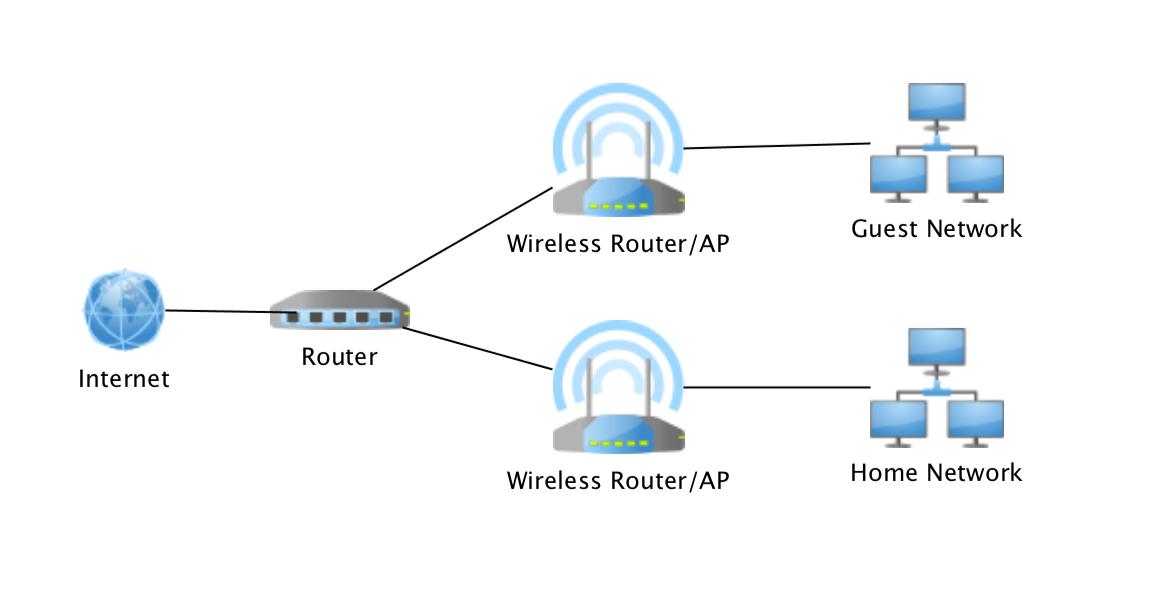
* **Advantages: -**
* It can connect the devices of two different networks having dissimilar structures.
* It is an intelligent device with filtering capabilities.
* It has control over both collisions as well as a broadcast domain.
* It uses a full-duplex mode of communication.
* It has the fastest data transmission speed amongst all network connecting devices.
* It can perform data translation and protocol conversion of the data packet as per the destination network's need.
* It can encapsulate and decapsulate the data packets.
* It has improved security than any other network connecting device.
* **Disadvantages: -**
* It is complex to design and implement.
* The implementation cost is very high.
* It requires a special system administration configuration.

1. **Router: -**
2. A Router is a process of selecting a path along which the data can be

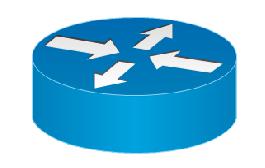
transferred from source to the destination. Routing is performed by a special device known as a router.

1. A Router works at the network layer in the OSI model and the internet layer in TCP/IP model
2. A router is a device like a switch that routes data packets based on their IP addresses. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

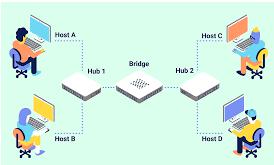
* **Advantages: -**
* It can choose the best path across the internetwork using dynamic routing algorithms
* It can reduce network traffic by creating collision domains and also by creating broadcast domains
* It provides sophisticated routing, flow control, and traffic isolation
* Can connect different network architecture, such as Ethernet and token ring
* **Disadvantages: -**
* A router is more expensive than bridge or repeaters
* The router is slower than bridge or repeaters because they must analyze data transmission from the physical to the network layer
* Dynamic router communication causes additional network traffic
* Are relatively complex device
* They are protocol-dependent devices that must understand the protocol they are forwarding.

1. **Bridge: -**
2. A bridge operates at the data link layer.
3. A bridge is a repeater, with add-on the functionality of filtering content by reading the MAC addresses of source and destination.
4. It is also used for interconnecting two LANs working on the same

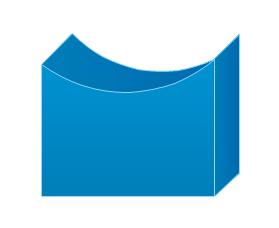
protocol.

1. It has a single input and single output port, thus making it a 2-port device.

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

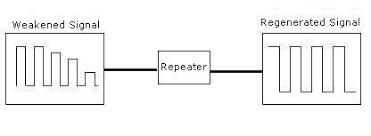
* **Advantages: -**
* It helps in the extension of the physical network.
* It reduces network traffic with minor segmentation.
* It creates separate collision domains. Hence it increases available bandwidth to individual nodes as fewer nodes share a collision domain.
* It reduces collisions.
* Some bridges connect networks having different architectures and media types.
* **Disadvantages: -**
* It is slower compared to repeaters due to filtering.
* It does not filter broadcasts.
* It is more expensive compared to repeaters.

1. **Repeaters: -**
2. The operating of a repeater can be done at the physical layer.
3. The main function of this device is to reproduce the signal on a similar network before the signal gets weak or otherwise damaged.
4. The significant point to be noted regarding these devices is that they do not strengthen the signal. Whenever the signal gets weak, then they

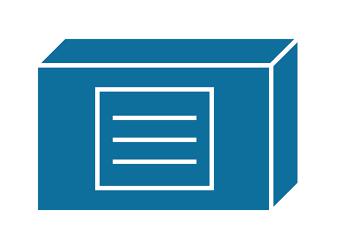
reproduce it at the actual strength.

1. A repeater is a two-port device

**Diagram: -**



**Logical Symbol: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* Repeaters are simple to install and can easily extend the length of the coverage area of networks.
* They are cost-effective.
* Repeaters don’t require any processing overhead. The only time they need to be investigated is in case of degradation of performance.
* They can connect signals using different types of cables.
* **Disadvantages: -**
* Repeaters cannot connect dissimilar networks.
* They cannot differentiate between actual signal and noise.
* They cannot reduce network traffic or congestion.
* Most networks have limitations upon the number of repeaters that can be deployed.

**Conclusion: -**

We have studied different networking devices and their advantages and disadvantages and also get the knowledge of their uses in detail. we also studied the different logical symbols of the networking devices. we get to know which device is used in the networking.

**Network Topologies: -**

* A Network Topology is the arrangement with which computer systems or network devices are connected to each other.
* Topologies may define both physical and logical aspects of the network. Both logical and physical topologies could be the same or different in the same network
* A network topology diagram is a visual representation of a network’s devices, connections, and paths, allowing you to picture how devices are interconnected and how they communicate with one another.
* Network diagrams are typically made to represent one or all of the first three-network layers (physical, data link, and network) according to the Open Systems Interconnection (OSI) model, collectively known as the media layers.

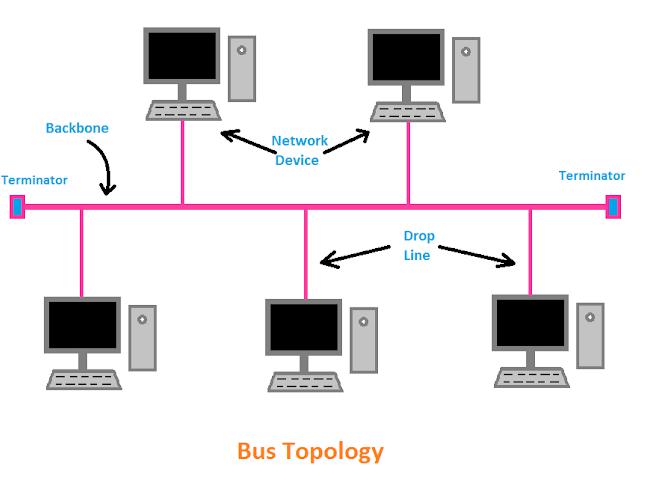
**Types of Network Topologies: -**

* **Bus Topology**
* **Mesh Topology**
* **Start Topology**
* **Ring Topology**
* **Hybrid Topology**
* **Tree Topology**

1. **Bus Topology: -**

* In the case of Bus topology, all devices sha are single communication lines or cables.
* Bus topology may have a problem while multiple hosts sending data at the same time. Therefore, Bus topology either uses CSMA (Carrier Sense Multiple Access) technologies or recognizes one host as Bus Master to solve the issue.
* Both ends of the shared channel have a line terminator. The data is sent in only one direction and as soon as it reaches the extreme end, the terminator removes the data from the line.
* The bus topology is mainly used in 802.3 (ethernet) and 802.4 standard networks.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Low-cost cable:** In a bus topology, nodes are directly connected to the cable without passing through a hub. Therefore, the initial cost of installation is low.
* **Moderate data speeds:** Coaxial or twisted pair cables are mainly used in bus-based networks that support up to 10 Mbps.
* **Familiar technology:** Bus topology is a familiar technology as the installation is easy.
* **Disadvantages: -**
* **Extensive cabling:** A bus topology is quite simpler, but still it requires a lot of cabling.
* **Difficult troubleshooting:** It requires specialized test equipment to determine the cable faults. If any fault occurs in the cable, then it would disrupt the communication for all the nodes.
* **Signal interference:** If two nodes send the messages simultaneously, then the signals of both the nodes collide with each other.
* **Reconfiguration difficult:** Adding new devices to the network would slow down the network.
* **Attenuation:** Attenuation is a loss of signal that leads to communication issues. Repeaters are used to regenerate the signal.

**Applications: -**

* Small workgroup local area networks (LANs) whose computers are connected using a thinnet cable.
* Trunk cables connecting hubs or switches of departmental LANs to form a larger LAN.
* Backboning, by joining switches and routers to form campus-wide networks.

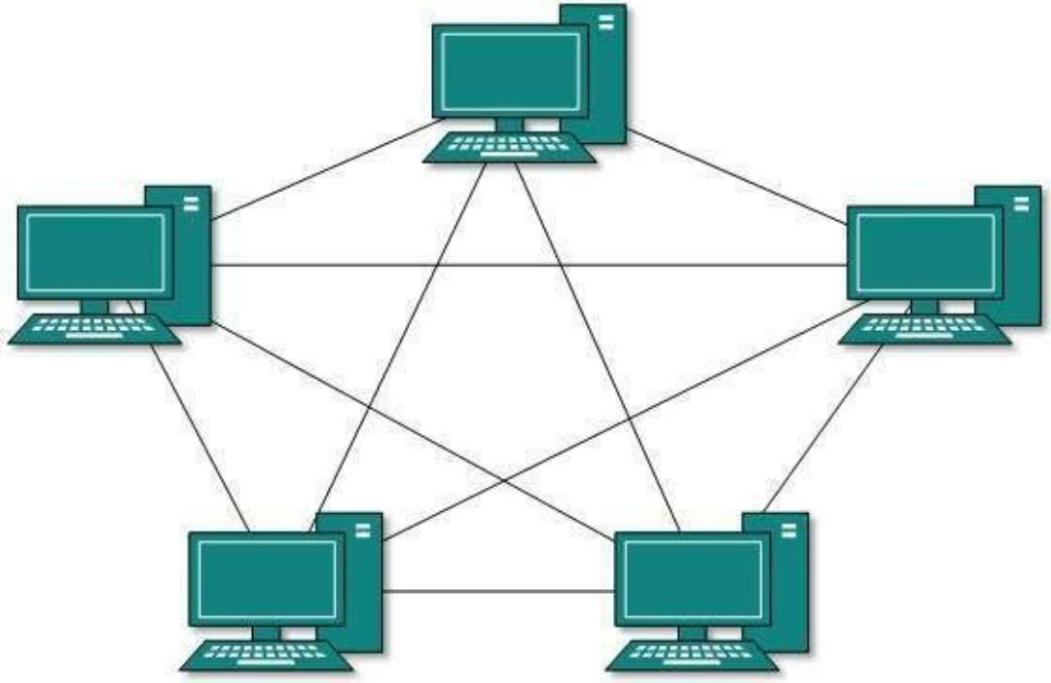
1. **Mesh Topology: -**

* In mesh, all the computers are interconnected to every other during a network.
* Each computer not only sends its own signals but also relays data from other computers.
* The nodes are connected to every other completely via a dedicated link during which information is travel from node to node.
* Every node features a point-to-point connection to the opposite node.
* Mesh topology is mainly used for WAN implementations where communication failures are a critical concern.
* Mesh topology is mainly used for wireless networks.
* Mesh topology can be formed by using the formula:

**Number of cables = (n\*(n-1))/2;**

Where n is the number of nodes that represents the network.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Reliable:** The mesh topology networks are very reliable as if any link breakdown will not affect the communication between connected computers.
* **Fast Communication:** Communication is very fast between the nodes.
* **Easier Reconfiguration:** Adding new devices would not disrupt the communication between other devices.
* **Disadvantages: -**
* **Cost:** A mesh topology contains a large number of connected devices such as a router and more transmission media than other topologies.
* **Management:** Mesh topology networks are very large and very difficult to maintain and manage. If the network is not monitored carefully, then the communication link failure goes undetected.
* **Efficiency:** In this topology, redundant connections are high which reduces the efficiency of the network.

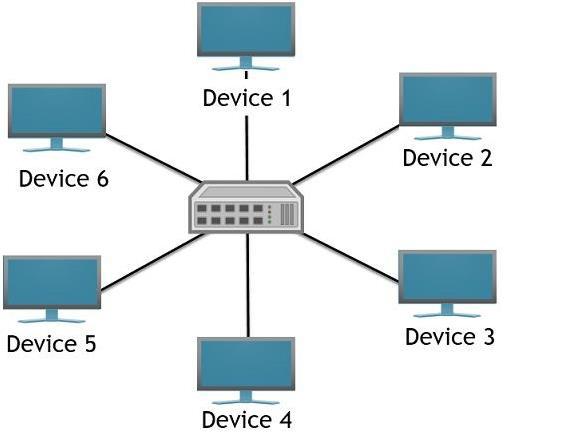
**Applications: -**

* Optical Mesh Networks enable the transport networks with the dynamic quality of service, bandwidth on demand, and managing bandwidth with peer nodes and applying policies.
* Rural or Village communication systems work with wireless nodes in every village connected to each other and provide communication services without depending on the operator infrastructure.
* Wireless Ad-hoc networks are decentralized networks with no predefined structure.

1. **Star Topology: -**

* A star may be a topology for a Local Area Network (LAN) during which all nodes are individually connected to a central connection point, sort of a hub, or a switch.
* A star takes more cable than e.g., a bus, but the benefit is that if a cable fails, just one node is going to be brought down.
* Each device within the network is connected to a central device called hub. If one device wants to send data to another device, it’s to first send the info to hub then the hub transmits that data to the designated device.
* The central computer is known as a server, and the peripheral devices attached to the server are known as clients.
* Coaxial cable or RJ-45 cables are used to connect the computers.
* Star topology is the most popular topology in network implementation.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Network control:** Complex network control features can be easily implemented in the star topology. Any changes made in the star topology are automatically accommodated.
* **Limited failure:** As each station is connected to the central hub with its own cable, therefore failure in one cable will not affect the entire network.
* **Familiar technology:** Star topology is a familiar technology as its tools are cost-effective.
* **Easily expandable:** It is easily expandable as new stations can be added to the open ports on the hub.
* **Cost-effective:** Star topology networks are cost-effective as it uses inexpensive coaxial cable.
* **High data speeds:** It supports a bandwidth of approx. 100Mbps. Ethernet 100BaseT is one of the most popular Star topology networks.
* **Disadvantages: -**
* **A Central point of failure:** If the central hub or switch goes down, then all the connected nodes will not be able to communicate with each other.
* **Cable:** Sometimes cable routing becomes difficult when a significant amount of routing is required.

**Applications: -**

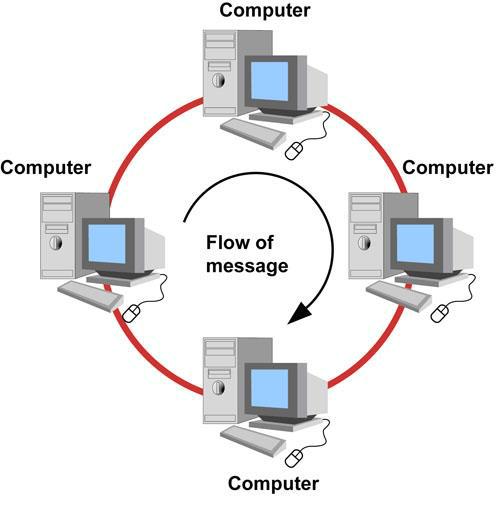
* Star topology is used in Local Area Network (LAN)
* High-speeded LAN often uses the star topology.
* Star topology is often used in homes and offices.
* Star topology is also used to transmit data along with the central hub between the network nodes.

1. **Ring Topology: -**

* In ring topology, it forms a ring connecting devices with exactly two neighbouring devices.
* A number of repeaters are used for Ring topology with a large number of nodes, because if someone wants to send some data to the last node in the ring topology with 100 nodes, then the data will have to pass through 99 nodes to reach the 100th node.
* Hence to prevent data loss repeaters are used in the network.
* The transmission is unidirectional, but it can be made bidirectional by having 2 connections between each Network Node, it is called Dual Ring Topology.
* The data in a ring topology flow in a clockwise direction.
* The most common access method of the ring topology is **token passing**.

1. **Token passing:** It is a network access method in which token is passed from one node to another node.
2. **Token:** It is a frame that circulates around the network.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Network Management:** Faulty devices can be removed from the network without bringing the network down.
* **Product availability:** Many hardware and software tools for network operation and monitoring are available.
* **Cost:** Twisted pair cabling is inexpensive and easily available. Therefore, the installation cost is very low.
* **Reliable:** It is a more reliable network because the communication system is not dependent on a single host computer.
* **Disadvantages: -**
* **Difficult troubleshooting:** It requires specialized test equipment to determine the cable faults. If any fault occurs in the cable, then it would disrupt the communication for all the nodes.
* **Failure:** The breakdown in one station leads to the failure of the overall network.
* **Reconfiguration difficult:** Adding new devices to the network would slow down the network.
* **Delay:** Communication delay is directly proportional to the number of nodes. Adding new devices increases the communication delay.

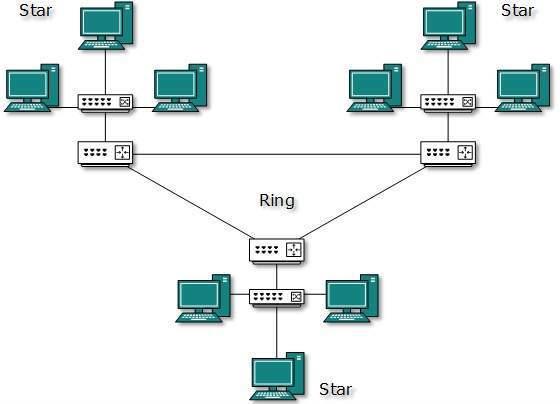
**Applications: -**

* Ring network topologies are used when a simple network is needed.
* They are suitable for locations that do not rely on very high data-transfer speeds, and where the network is unlikely to alter in size or structure.
* For example, a small office with only a few nodes may use a ring network topology.

1. **Hybrid Topology: -**

* The combination of various different topologies is known as **Hybrid topology**.
* A Hybrid topology is a connection between different links and nodes to transfer the data.
* When two or more different topologies are combined together is termed as Hybrid topology and if similar topologies are connected with each other will not result in Hybrid topology.
* For example, if there exists a ring topology in one branch of ICICI bank and bus topology in another branch of ICICI bank, connecting these two topologies will result in Hybrid topology.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Reliable:** If a fault occurs in any part of the network will not affect the functioning of the rest of the network.
* **Scalable:** The size of the network can be easily expanded by adding new devices without affecting the functionality of the existing network.
* **Flexible:** This topology is very flexible as it can be designed according to the requirements of the organization.
* **Effective:** Hybrid topology is very effective as it can be designed in such a way that the strength of the network is maximized and the weakness of the network is minimized.
* **Disadvantages: -**
* **Complex design:** The major drawback of the Hybrid topology is the design of the Hybrid network. It is very difficult to design the architecture of the Hybrid network.
* **Costly Hub:** The Hubs used in the Hybrid topology are very expensive as these hubs are different from usual Hubs.
* **Costly infrastructure:** The infrastructure cost is very high as a hybrid network requires a lot of cabling, network devices, etc.

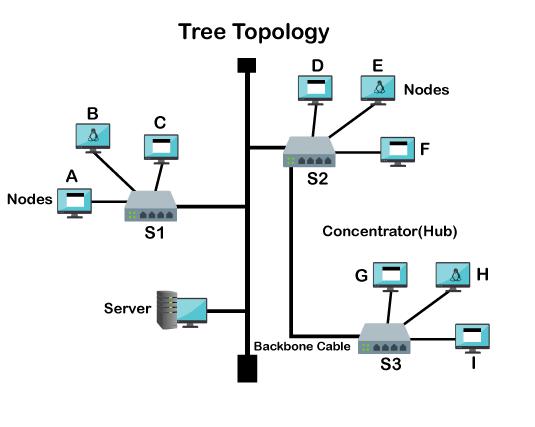
**Applications: -**

* Network growth is when more network nodes are added to an existing network. A hybrid network eases the addition of new nodes to the network as changes can be done at the basic network levels as well as on the main network.
* The hybrid topology is more useful when you need to fulfil diversity in Computer Network. In this topology, all network sections can include the configuration of different Network Topology.

1. **Tree Topology: -**

* Tree topology also known as Hierarchical Topology, is the most common form of network topology in use presently.
* This topology imitates an extended Star topology and inherits properties of bus topology.
* This topology divides the network into multiple levels/layers of the network. Mainly in LANs, a network is bifurcated into three types of network devices.
* The lowermost is the access layer where computers are attached. The middle layer is known as the distribution layer, which works as a mediator between the upper layer and the lower layer. The highest layer is known as the core layer and is the central point of the network, i.e., the root of the tree from which all nodes fork.
* It is not the single point of failure. Every connection serves as point of failure, failing of which divides the network into unreachable segment.

**Diagram: -**



**Advantages & Disadvantages: -**

* **Advantages: -**
* **Support for broadband transmission:** Tree topology is mainly used to provide broadband transmission, i.e., signals are sent over long distances without being attenuated.
* **Easily expandable:** We can add the new device to the existing network. Therefore, we can say that tree topology is easily expandable.
* **Easily manageable:** In tree topology, the whole network is divided into segments known as star networks which can be easily managed and maintained.
* **Error detection:** Error detection and error correction are very easy in a tree topology.
* **Limited failure:** The breakdown in one station does not affect the entire network.
* **Point-to-point wiring:** It has point-to-point wiring for individual segments.
* **Disadvantages: -**
* **Difficult troubleshooting:** If any fault occurs in the node, then it becomes difficult to troubleshoot the problem.
* **High cost:** Devices required for broadband transmission are very costly.
* **Failure:** A tree topology mainly relies on the main bus cable and failure in the main bus cable will damage the overall network.
* **Reconfiguration difficult:** If new devices are added, then it becomes difficult to reconfigure.

**Applications: -**

* When you have a multi-story building and wish to establish clusters at each section of the network, you can utilise tree topology.
* If you have departments and sub-departments, you can segregate the whole Tree Network with the help of several switches that makes the entire network easy to maintain and more manageable.

**Conclusion: -**

We studied different topologies in detail. We studied the advantages and disadvantages and its application in the networking. Selection of topology to be done on the requirement of the network and users’ capacity.

***Experiment – 2***

**Aim: -** To study different networking commands.

**Networking Commands:**

1. **IPCONFIG**

It stands for **Internet Protocol Configuration**. This is a command-line application that displays all the current TCP/IP (Transmission Control Protocol/Internet Protocol) network configuration, refreshes the DHCP (Dynamic Host Configuration Protocol) and DNS (Domain Name Server). It also displays IP address, subnet mask, and default gateway for all adapters.

**Example:**

Windows IP Configuration

Ethernet adapter Ethernet:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection\* 3:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection\* 4:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

|  |  |
| --- | --- |
| Link-local IPv6 Address . . | . . . : fe80::e482:2a01:d8f4:fdfd%16 |
| IPv4 Address. . . . . . . . . . . | : 192.168.0.107 |
| Subnet Mask . . . . . . . . . . | . : 255.255.255.0 |
| Default Gateway . . . . . . . | . . : 192.168.0.1 |

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

1. **IPCONFIG/all**

Displays the full TCP/IP configuration for all adapters. Adapters can represent physical interfaces, such as installed network adapters, or logical interfaces, such as dial-up connections.

**Example:**

Windows IP Configuration

Host Name . . . . . . . . . . . . : DESKTOP-N65E7CV

Primary Dns Suffix . . . . . . . :

Node Type . . . . . . . . . . . . : Hybrid

IP Routing Enabled. . . . . . . . : No

WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Realtek PCIe FE Family Controller

Physical Address. . . . . . . . . : C0-3E-BA-36-F9-43

DHCP Enabled. . . . . . . . . . . : Yes

Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection\* 3:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #3

Physical Address. . . . . . . . . : 2A-CD-C4-58-15-6B

DHCP Enabled. . . . . . . . . . . : Yes

Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Local Area Connection\* 4:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Microsoft Wi-Fi Direct Virtual Adapter #4

Physical Address. . . . . . . . . : 3A-CD-C4-58-15-6B

DHCP Enabled. . . . . . . . . . . : No

Autoconfiguration Enabled . . . . : Yes

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :

|  |  |  |
| --- | --- | --- |
| Description . . . . . . . . . . . | : Qualcomm QCA9377 802.11ac Wireless Adapter | |
| Physical Address. . . . . . . | . . | : 28-CD-C4-58-15-6B |
| DHCP Enabled. . . . . . . . . | . . | : Yes |
| Autoconfiguration Enabled | | . . . . : Yes |
| Link-local IPv6 Address . | . . | . . : fe80::e482:2a01:d8f4:fdfd%16(Preferred) |
| IPv4 Address. . . . . . . . . . | . : 192.168.0.107(Preferred) | |
| Subnet Mask . . . . . . . . . . | . : 255.255.255.0 | |
| Lease Obtained. . . . . . . . | . . | : 30 March 2022 17:14:06 |
| Lease Expires . . . . . . . . . . | : 30 March 2022 19:14:05 | |
| Default Gateway . . . . . . . . . | | : 192.168.0.1 |
| DHCP Server . . . . . . . . . . | . : 192.168.0.1 | |
| DHCPv6 IAID . . . . . . . . . . | . : 153669060 | |
| DHCPv6 Client DUID. . . . | . . | . . : 00-01-00-01-26-D3-64-06-C0-3E-BA-36-F9-43 |
| DNS Servers . . . . . . . . . . . | : 192.168.0.1 | |
| NetBIOS over Tcpip. . . . . . . | | . : Enabled |

Ethernet adapter Bluetooth Network Connection:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . :

Description . . . . . . . . . . . : Bluetooth Device (Personal Area Network)

Physical Address. . . . . . . . . : 28-CD-C4-58-15-6C

DHCP Enabled. . . . . . . . . . . : Yes

Autoconfiguration Enabled . . . . : Yes

1. **Ping(IP address/web address)**

Verifies IP-level connectivity to another TCP/IP computer by sending Internet Control Message Protocol (ICMP) echo Request messages. The receipt of corresponding echo Reply messages are displayed, along with round-trip times. ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution. Used without parameters, this command displays Help content.

**Example:**

Pinging www.google.com [172.217.160.196] with 32 bytes of data:

Reply from 172.217.160.196: bytes=32 time=84ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Reply from 172.217.160.196: bytes=32 time=4ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Ping statistics for 172.217.160.196:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 4ms, Maximum = 84ms, Average = 25ms

1. **Ping -t(IP address/web address)**

Specifies ping continue sending echo Request messages to the destination until interrupted. To interrupt and display statistics, press CTRL+ENTER. To interrupt and quit this command, press CTRL+C.

**Example:**

Pinging www.google.com [172.217.160.196] with 32 bytes of data:

Reply from 172.217.160.196: bytes=32 time=4ms TTL=120

Reply from 172.217.160.196: bytes=32 time=8ms TTL=120

Reply from 172.217.160.196: bytes=32 time=4ms TTL=120

Reply from 172.217.160.196: bytes=32 time=7ms TTL=120

Reply from 172.217.160.196: bytes=32 time=8ms TTL=120

Reply from 172.217.160.196: bytes=32 time=8ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Reply from 172.217.160.196: bytes=32 time=7ms TTL=120

Reply from 172.217.160.196: bytes=32 time=15ms TTL=120

Reply from 172.217.160.196: bytes=32 time=8ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Reply from 172.217.160.196: bytes=32 time=4ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Reply from 172.217.160.196: bytes=32 time=7ms TTL=120

Reply from 172.217.160.196: bytes=32 time=7ms TTL=120

Reply from 172.217.160.196: bytes=32 time=5ms TTL=120

Reply from 172.217.160.196: bytes=32 time=6ms TTL=120

Ping statistics for 172.217.160.196:

Packets: Sent = 17, Received = 17, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 4ms, Maximum = 15ms, Average = 6ms

Control-C

^C

1. **Netstat**

Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols). Used without parameters, this command displays active TCP connections.

**Example:**

Active Connections

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Proto Local Address | | Foreign Address | | State | |
| TCP | 127.0.0.1:49671 | DESKTOP-N65E7CV:49672 ESTABLISHED | | | |
| TCP | 127.0.0.1:49672 | DESKTOP-N65E7CV:49671 ESTABLISHED | | | |
| TCP | 127.0.0.1:49673 | DESKTOP-N65E7CV:49674 ESTABLISHED | | | |
| TCP | 127.0.0.1:49674 | DESKTOP-N65E7CV:49673 ESTABLISHED | | | |
| TCP | 192.168.0.107:54371 | | 23.98.104.196:https | | ESTABLISHED |
| TCP | 192.168.0.107:54374 | | 8.241.155.126:http | | CLOSE\_WAIT |
| TCP | 192.168.0.107:61183 | | 20.197.71.89:https | | ESTABLISHED |
| TCP | 192.168.0.107:61214 | | 52.98.63.18:https | | ESTABLISHED |
| TCP | 192.168.0.107:61342 | | 40.70.161.7:https | | CLOSE\_WAIT |

TCP 192.168.0.107:61359 bom12s21-in-f13:https TIME\_WAIT

|  |  |  |  |
| --- | --- | --- | --- |
| TCP | 192.168.0.107:61362 | bom12s13-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61364 | si-in-f188:5228 | ESTABLISHED |
| TCP | 192.168.0.107:61365 | bom12s18-in-f14:https ESTABLISHED | |
| TCP | 192.168.0.107:61368 | bom07s27-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61372 | bom12s16-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61375 | se-in-f101:https TIME\_WAIT | |
| TCP | 192.168.0.107:61376 | bom12s21-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61385 | bom07s31-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61386 | bom12s12-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61388 | bom07s36-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61389 | hkg12s10-in-f46:https TIME\_WAIT | |

TCP 192.168.0.107:61391 bom07s45-in-f5:https ESTABLISHED

|  |  |  |  |
| --- | --- | --- | --- |
| TCP | 192.168.0.107:61393 | bom12s17-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61399 | bom12s04-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61401 | bom07s30-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61402 | bom12s03-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61403 | bom12s19-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61405 | bom05s15-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61406 | bom12s14-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61407 | bom12s12-in-f14:https TIME\_WAIT | |
| TCP | 192.168.0.107:61408 | bom07s16-in-f10:https TIME\_WAIT | |
| TCP 192.168.0.107:61409 52.98.123.226:https ESTABLISHED | | | |
| TCP 192.168.0.107:61412 52.98.123.226:https ESTABLISHED | | | |
| TCP | 192.168.0.107:61414 | 219:https | ESTABLISHED |
| TCP | 192.168.0.107:61416 | bom07s45-in-f10:https TIME\_WAIT | |
| TCP | 192.168.0.107:61417 | bom12s18-in-f14:https TIME\_WAIT | |

TCP 192.168.0.107:61418 bom07s36-in-f3:https TIME\_WAIT

TCP 192.168.0.107:61419 bom07s16-in-f3:https TIME\_WAIT TCP 192.168.0.107:61463 bom07s16-in-f3:https ESTABLISHED

|  |  |  |  |
| --- | --- | --- | --- |
| TCP | 192.168.0.107:61575 | hkg12s09-in-f14:https ESTABLISHED | |
| TCP | 192.168.0.107:61576 | e2a:https | ESTABLISHED |
| TCP | 192.168.0.107:61577 | bom07s28-in-f17:https ESTABLISHED | |

TCP 192.168.0.107:61586 bom07s45-in-f5:https ESTABLISHED TCP 192.168.0.107:61595 52.231.199.126:https TIME\_WAIT TCP 192.168.0.107:61596 52.231.199.126:https TIME\_WAIT TCP 192.168.0.107:61608 bom07s18-in-f3:https ESTABLISHED TCP 192.168.0.107:61614 bom07s33-in-f22:https TIME\_WAIT TCP 192.168.0.107:61616 bom12s12-in-f10:https TIME\_WAIT TCP 192.168.0.107:61617 bom07s36-in-f1:https TIME\_WAIT TCP 192.168.0.107:61620 bom07s15-in-f3:https ESTABLISHED

TCP 192.168.0.107:61622 a104-120-79-78:https ESTABLISHED

TCP 192.168.0.107:61623 a104-120-79-78:https ESTABLISHED

1. **Netstat -an**

Display all active TCP connections and the TCP and UDP ports on which the computer is listening.

**Example:**

Active Connections

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proto Local Address | | Foreign Address | | State |
| TCP | 0.0.0.0:135 | 0.0.0.0:0 | LISTENING | |
| TCP | 0.0.0.0:445 | 0.0.0.0:0 | LISTENING | |
| TCP | 0.0.0.0:3306 | 0.0.0.0:0 | LISTENING | |
| TCP | 0.0.0.0:5040 | 0.0.0.0:0 | LISTENING | |
| TCP | 0.0.0.0:5700 | 0.0.0.0:0 | LISTENING | |
| TCP | 0.0.0.0:6646 | 0.0.0.0:0 | LISTENING | |

|  |  |  |  |
| --- | --- | --- | --- |
| TCP | 0.0.0.0:33060 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49664 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49665 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49666 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49667 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49668 | 0.0.0.0:0 | LISTENING |
| TCP | 0.0.0.0:49670 | 0.0.0.0:0 | LISTENING |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TCP | 127.0.0.1:8884 | 0.0.0.0:0 | LISTENING | |
| TCP | 127.0.0.1:9012 | 0.0.0.0:0 | LISTENING | |
| TCP | 127.0.0.1:49671 | 127.0.0.1:49672 | | ESTABLISHED |
| TCP | 127.0.0.1:49672 | 127.0.0.1:49671 | | ESTABLISHED |
| TCP | 127.0.0.1:49673 | 127.0.0.1:49674 | | ESTABLISHED |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TCP | 127.0.0.1:49674 | | 127.0.0.1:49673 | | ESTABLISHED | |
| TCP | 127.0.0.1:55982 | | 0.0.0.0:0 | | LISTENING | |
| TCP | 192.168.0.107:139 | | | 0.0.0.0:0 | LISTENING | |
| TCP | 192.168.0.107:55664 | | | 20.197.71.89:443 | | ESTABLISHED |
| TCP | 192.168.0.107:55681 | | | 23.98.104.196:443 | | ESTABLISHED |
| TCP | 192.168.0.107:55691 | | | 52.109.124.47:443 TIME\_WAIT | | |
| TCP | 192.168.0.107:55692 | | | 52.109.124.47:443 TIME\_WAIT | | |
| TCP | 192.168.0.107:55693 | | | 52.109.124.71:443 TIME\_WAIT | | |
| TCP | 192.168.0.107:55695 | | | 52.109.56.76:443 | | TIME\_WAIT |
| TCP | 192.168.0.107:55696 52.182.143.208:443 TIME\_WAIT | | | | | |
| TCP | 192.168.0.107:55698 | | | 13.107.213.48:443 TIME\_WAIT | | |
| TCP | 192.168.0.107:55699 | | | 35.247.144.219:443 | | ESTABLISHED |
| TCP | 192.168.0.107:55701 | | | 52.109.124.18:443 | | ESTABLISHED |
| TCP | 192.168.0.107:55702 | | | 52.109.8.19:443 | | ESTABLISHED |
| TCP | [::]:135 | [::]:0 | | LISTENING | |  |
| TCP | [::]:445 | [::]:0 | | LISTENING | |  |
| TCP | [::]:3306 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:5700 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:33060 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49664 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49665 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49666 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49667 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49668 |  | [::]:0 | LISTENING | |  |
| TCP | [::]:49670 |  | [::]:0 | LISTENING | |  |

|  |  |  |  |
| --- | --- | --- | --- |
| TCP | [::1]:49669 | [::]:0 | LISTENING |
| UDP | 0.0.0.0:123 | \*:\* |  |
| UDP | 0.0.0.0:500 | \*:\* |  |
| UDP | 0.0.0.0:4500 | \*:\* |  |

|  |  |  |  |
| --- | --- | --- | --- |
| UDP | 0.0.0.0:5050 | \*:\* | |
| UDP | 0.0.0.0:5353 | \*:\* | |
| UDP | 0.0.0.0:5355 | \*:\* | |
| UDP | 0.0.0.0:6646 | \*:\* | |
| UDP | 0.0.0.0:52782 | \*:\* | |
| UDP | 127.0.0.1:1900 | \*:\* | |
| UDP | 127.0.0.1:49666 | | 127.0.0.1:49666 |
| UDP | 127.0.0.1:50154 | | \*:\* |
| UDP | 192.168.0.107:137 | | \*:\* |
| UDP | 192.168.0.107:138 | | \*:\* |
| UDP | 192.168.0.107:1900 | | \*:\* |
| UDP | 192.168.0.107:2177 | | \*:\* |
| UDP | 192.168.0.107:50153 \*:\* | | |
| UDP | [::]:123 | \*:\* |  |
| UDP | [::]:500 | \*:\* |  |
| UDP | [::]:4500 | \*:\* |  |
| UDP | [::]:5353 | \*:\* |  |
| UDP | [::]:5355 | \*:\* |  |
| UDP | [::]:52782 | \*:\* |  |
| UDP | [::1]:1900 | \*:\* |  |
| UDP | [::1]:50152 | \*:\* | |
| UDP | [fe80::e482:2a01:d8f4:fdfd%16]:1900 \*:\* | | |
| UDP | [fe80::e482:2a01:d8f4:fdfd%16]:2177 \*:\* | | |
| UDP | [fe80::e482:2a01:d8f4:fdfd%16]:50151 \*:\* | | |

1. **Pathping**

Provides information about network latency and network loss at intermediate hops between a source and destination. This command sends multiple echo Request messages to each router between a source and destination, over a period of time, and then computes results based on the packets returned from each router. Because this command displays the degree of packet loss at any given router or link, you can determine which routers or subnets might be having network problems. Used without parameters, this command displays help.

**Example:**

Usage: pathping [-g host-list] [-h maximum\_hops] [-i address] [-n]

[-p period] [-q num\_queries] [-w timeout]

[-4] [-6] target\_name

Options:

-g host-list Loose source route along host-list.

-h maximum\_hops Maximum number of hops to search for target.

-i address

Use the specified source address.

-n

Do not resolve addresses to hostnames.

-p period

Wait period milliseconds between pings.

-q num\_queries Number of queries per hop.

|  |  |  |
| --- | --- | --- |
| -w timeout | | Wait timeout milliseconds for each reply. |
| -4 | Force using IPv4. | |
| -6 | Force using IPv6. | |

1. **Arp-a**

Displays current arp cache tables for all interfaces. The **/n** parameter is case-sensitive. To display the arp cache entry for a specific IP address, use **arp /a** with the **inetaddr** parameter, where **inetaddr** is an IP address. If **inetaddr** is not specified, the first applicable interface is used. To display the arp cache table for a specific interface, use the **/n ifaceaddr** parameter in conjunction with the **/a** parameter where **inetaddr** is the IP address assigned to the interface.

**Example:**

Interface: 192.168.0.107 --- 0x10

Internet Address Physical Address Type

192.168.0.1 e8-48-b8-c2-73-55 dynamic

192.168.0.102 2c-d9-74-5c-60-d2 dynamic

224.0.0.22 01-00-5e-00-00-16 static

224.0.0.251 01-00-5e-00-00-fb static

224.0.0.252 01-00-5e-00-00-fc static

239.255.255.250 01-00-5e-7f-ff-fa static

255.255.255.255 ff-ff-ff-ff-ff-ff static

**9.Nslookup**

Displays information that you can use to diagnose Domain Name System (DNS) infrastructure. Before using this tool, you should be familiar with how DNS works. The nslookup command-line tool is available only if you have installed the TCP/IP protocol.

**Example:**

Default Server: UnKnown

Address: 192.168.0.1

**Conclusion: -**

Learn and execute various networking commands on command prompt.

***Experiment – 3***

**Aim: -** To implement CRC and hamming code as error detection and correction codes.

**CRC:**

CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in the communication channel.   
CRC uses **Generator Polynomial**which is available on both sender and receiver side. An example generator polynomial is of the form like x3 + x + 1. This generator polynomial represents key 1011. Another example is x2 + 1 that represents key 101.

Code: -

import java.util.Scanner;

class CRC {

    String xor(String div, String temp) {

        int i;

        StringBuilder result = new StringBuilder();

        for (i = 0; i < temp.length(); i++) {

            if (div.charAt(i) == temp.charAt(i)) {

                result.append("0");

            } else {

                result.append("1");

            }

        }

        return result.substring(1).toString();

    }

    String getDividend(String data, String divisor) {

        int i, pick = divisor.length();

        StringBuilder dividend = new StringBuilder();

        dividend.append(data);

        for (i = 0; i < pick - 1; i++) {

            dividend.append("0");

        }

        return dividend.toString();

    }

    String getRemainder(String dividend, String divisor) {

        int i, pick = divisor.length();

        StringBuilder zeros = new StringBuilder();

        String temp = dividend.substring(0, pick);

        while (pick < dividend.length()) {

            if (temp.charAt(0) == '1') {

                temp = xor(divisor, temp) + dividend.charAt(pick);

            } else {

                for (i = 0; i < pick; i++) {

                    zeros.append("0");

                }

                temp = xor(zeros.toString(), temp) + dividend.charAt(pick);

                zeros.setLength(0);

            }

            pick += 1;

        }

        if (temp.charAt(0) == '1') {

            temp = xor(divisor, temp);

        } else {

            for (i = 0; i < pick; i++) {

                zeros.append("0");

            }

            temp = xor(zeros.toString(), temp);

        }

        return temp;

    }

*void* checkCodeword(*String* *codeword*, *String* *divisor*) {

*int* remainder = Integer.parseInt(getRemainder(codeword, divisor));

        if (remainder > 0) {

            System.out.println("Corrupted data received");

        } else {

            System.out.println("Data received without any error");

        }

    }

}

public class ErrorDetectionCRC {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        CRC crc = new CRC();

        String data, divisor, dividend, codeword, userCodeword;

        System.out.println("Enter the data: ");

        data = in.next();

        System.out.println("Enter the divisor: ");

        divisor = in.next();

        dividend = crc.getDividend(data, divisor);

        codeword = data + crc.getRemainder(dividend, divisor);

        System.out.println("The codeword is: " + codeword);

        System.out.println("Enter the codeword to check its correctness: ");

        userCodeword = in.next();

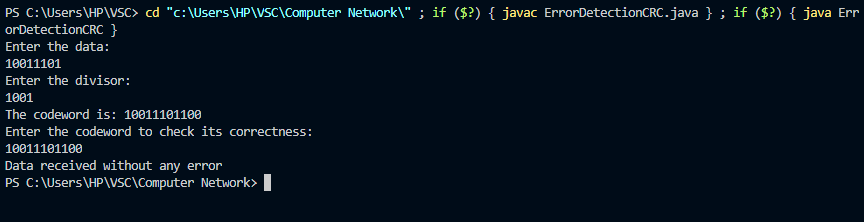
        crc.checkCodeword(userCodeword, divisor);

        in.close();

    }

}

Output: -



**Hamming:**

[Hamming code](https://www.geeksforgeeks.org/computer-network-hamming-code/) is a set of **error-correction codes** that can be used to detect and correct the errors that can occur when the data is moved or stored from the sender to the receiver. It is a technique developed by **R.W. Hamming** for error correction.

**Example:**

**Input:**

message bit = 0101

r1 r2 m1 r4 m2 m3 m4

0 1 0 1

**Output:**

Generated codeword:

r1 r2 m1 r4 m2 m3 m4

0 1 0 0 1 0 1

Code: -

import java.util.Scanner;

class Hamming {

*int*[] getHammingCode(*int*[] *data*) {

*int*[] returnData;

*int* size;

*int* i = 0, j = 0, k = 0, parityBits = 0;

        size = *data*.length;

        while (i < size) {

            if (Math.pow(2, parityBits) == (i + parityBits + 1)) {

                parityBits++;

            } else {

                i++;

            }

        }

        returnData = new *int*[size + parityBits];

        for (i = 1; i <= returnData.length; i++) {

            if (Math.pow(2, j) == i) {

                returnData[(i - 1)] = 2;

                j++;

            } else {

                returnData[(k + j)] = *data*[k++];

            }

        }

        for (i = 0; i < parityBits; i++) {

            returnData[((*int*) Math.pow(2, i)) - 1] = getParityBit(returnData, i);

        }

        return returnData;

    }

*int* getParityBit(*int*[] *returnData*, *int* *pow*) {

*int* parityBit = 0;

*int* size = *returnData*.length;

        for (*int* i = 0; i < size; i++) {

            if (*returnData*[i] != 2) {

*int* k = (i + 1);

                String str = Integer.toBinaryString(k);

*int* temp = ((Integer.parseInt(str)) / ((*int*) Math.pow(10, *pow*))) % 10;

                if (temp == 1) {

                    if (*returnData*[i] == 1) {

                        parityBit = (parityBit + 1) % 2;

                    }

                }

            }

        }

        return parityBit;

    }

*void* receiveData(*int*[] *data*, *int* *parityBits*) {

*int* pow;

*int* size = *data*.length;

*int*[] parityArray = new *int*[*parityBits*];

        StringBuilder errorLoc = new StringBuilder();

        for (pow = 0; pow < *parityBits*; pow++) {

            for (*int* i = 0; i < size; i++) {

*int* j = i + 1;

                String str = Integer.toBinaryString(j);

*int* bit = ((Integer.parseInt(str)) / ((*int*) Math.pow(10, pow))) % 10;

                if (bit == 1) {

                    if (*data*[i] == 1) {

                        parityArray[pow] = (parityArray[pow] + 1) % 2;

                    }

                }

            }

            errorLoc.insert(0, parityArray[pow]);

        }

        int finalLoc = Integer.parseInt(errorLoc.toString(), 2);

        if (finalLoc != 0) {

            System.out.println("Error is found at location " + finalLoc + ".");

*data*[finalLoc - 1] = (*data*[finalLoc - 1] + 1) % 2;

            System.out.println("After correcting the error, the code is:");

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

                System.out.print(*data*[size - i - 1]);

            }

            System.out.println();

        } else {

            System.out.println("There is no error in the received data.");

        }

        System.out.println("The data sent from the sender:");

        pow = *parityBits* - 1;

        for (int k = size; k > 0; k--) {

            if (Math.pow(2, pow) != k) {

                System.out.print(*data*[k - 1]);

            } else {

                pow--;

            }

        }

        System.out.println();

    }

}

public class ErrorDetectionHamming {

    public static void main(String[] *args*) {

        Scanner in = new Scanner(System.in);

        Hamming hamming = new Hamming();

        int i, j, k, size, hammingCodeSize, errorPosition;

        int[] arr, hammingCode;

        System.out.println("Enter the bits size for the data: ");

        size = in.nextInt();

        arr = new int[size];

        for (j = 0; j < size; j++) {

            System.out.println("Enter " + (size - j) + "-bit of the data: ");

            arr[size - j - 1] = in.nextInt();

        }

        System.out.println("The data which you enter is: ");

        for (k = 0; k < size; k++) {

            System.out.print(arr[size - k - 1]);

        }

        System.out.println();

        hammingCode = hamming.getHammingCode(arr);

        hammingCodeSize = hammingCode.length;

        System.out.println("The hamming code generated for your data is: ");

        for (i = 0; i < hammingCodeSize; i++) {

            System.out.print(hammingCode[(hammingCodeSize - i - 1)]);

        }

        System.out.println();

        System.out.println(

                "For detecting error at the receiver end, enter position of a bit to alter original data (0 for no error): ");

        errorPosition = in.nextInt();

        if (errorPosition != 0) {

            hammingCode[errorPosition - 1] = (hammingCode[errorPosition - 1] + 1) % 2;

        }

        System.out.println("Sent Data is: ");

        for (k = 0; k < hammingCodeSize; k++) {

            System.out.print(hammingCode[hammingCodeSize - k - 1]);

        }

        System.out.println(); // for next line

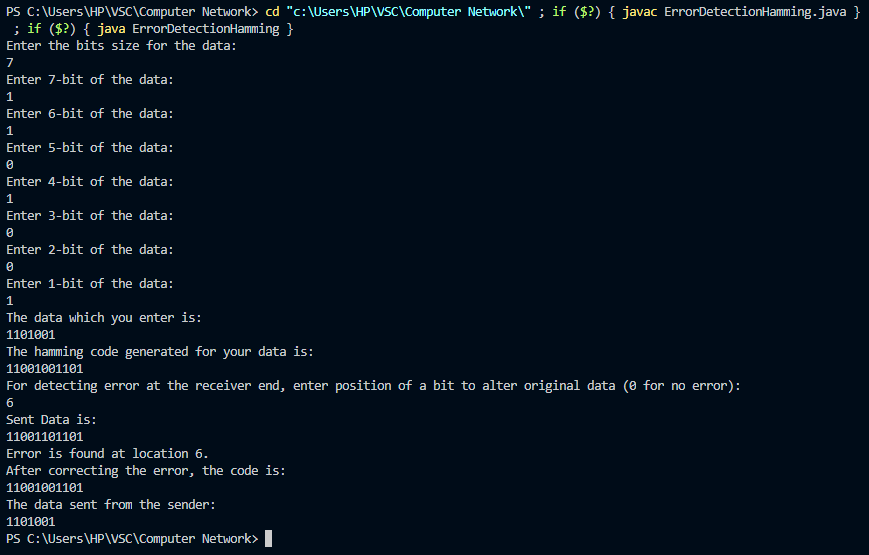
        hamming.receiveData(hammingCode, hammingCodeSize - arr.length);

        in.close();

    }

}

Output: -



**Conclusion: -**

Learnt about the two error detection techniques and implemented them in program.

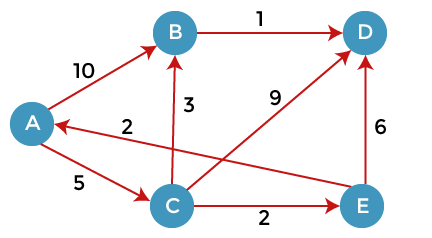
***Experiment – 4***

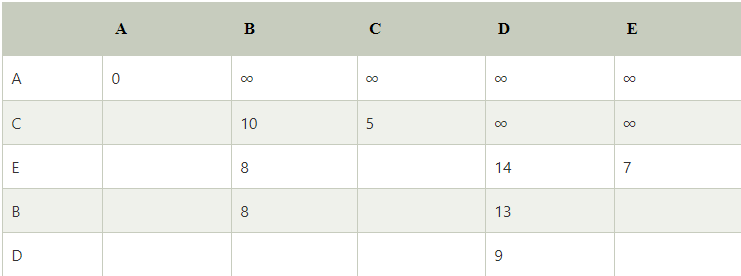
**Aim: -** To Study & implement Dijkstra’s Algorithm.

**Theory: -**

Dijkstra’s algorithm is very similar to [Prim’s algorithm for minimum spanning tree](https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/). Like Prim’s MST, we generate a*SPT (shortest path tree)* with a given source as a root. We maintain two sets, one set contains vertices included in the shortest-path tree, other set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.  
Below are the detailed steps used in Dijkstra’s algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Example:





**Code: -**import java.util.Scanner;

public class Dijkstra {

    static *int* V;

*int* minDistance(*int*[] *dist*, Boolean[] *sptSet*)

    {

*int* min = Integer.MAX\_VALUE, min\_index = -1;

        for (*int* v = 0; v < V; v++) {

            if (!*sptSet*[v] && *dist*[v] <= min) {

                min = *dist*[v];

                min\_index = v;

            }

        }

        return min\_index;

    }

*int*[] dijkstra(*int*[][] *graph*, *int* *src*)

    {

*int*[] dist = new *int*[V];

        Boolean[] sptSet = new Boolean[V];

        for (*int* i = 0; i < V; i++) {

            dist[i] = Integer.MAX\_VALUE;

            sptSet[i] = false;

        }

        dist[*src*] = 0;

        for (*int* count = 0; count < V - 1; count++) {

*int* u = minDistance(dist, sptSet);

            sptSet[u] = true;

            for (*int* v = 0; v < V; v++) {

                if (!sptSet[v] && *graph*[u][v] != 0 && dist[u] != Integer.MAX\_VALUE && dist[u] + *graph*[u][v] < dist[v]) {

                    dist[v] = dist[u] + *graph*[u][v];

                }

            }

        }

        return dist;

    }

    public static *void* main(String[] *args*)

    {

        Scanner in = new Scanner(System.in);

        Dijkstra d = new Dijkstra();

*int* i, j;

*int* [][] graph;

*int*[] dist;

        System.out.println("Enter the number of nodes: ");

        V = in.nextInt();

        graph = new *int*[V][V];

        System.out.println("Enter the graph: ");

        for(i = 0; i < V; i ++) {

            System.out.println("Enter row " + (i + 1) + ": ");

            for(j = 0; j < V; j ++) {

                graph[i][j] = in.nextInt();

            }

        }

        dist = d.dijkstra(graph, 0);

        System.out.println("Vertex \t\t Distance from Source");

        for (i = 0; i < V; i++) {

            System.out.println(i + " \t\t " + dist[i]);

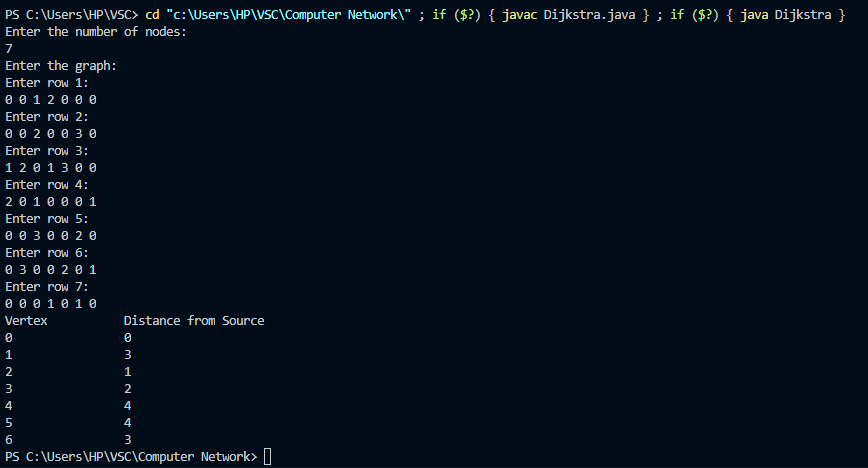
        }

        in.close();

    }

}

**Output: -**

****

**Application: -**

* **For map applications**, it is hugely deployed in measuring the least possible distance and check direction amidst two geographical regions like Google Maps, discovering map locations pointing to the vertices of a graph, calculating traffic and delay-timing, etc.
* **For telephone networks**, this is also extensively implemented in the conducting of data in networking and telecommunication domains for decreasing the obstacle taken place for transmission.
* Wherever addressing the need for shortest path explications either in the domain of robotics, transport, embedded systems, laboratory or production plants, etc, this algorithm is applied.
* Besides that, other applications are road conditions, road closures, construction, and IP routing to detect[Open Shortest Path First](https://www.metaswitch.com/knowledge-center/reference/what-is-open-shortest-path-first-ospf).

**Conclusion: -**

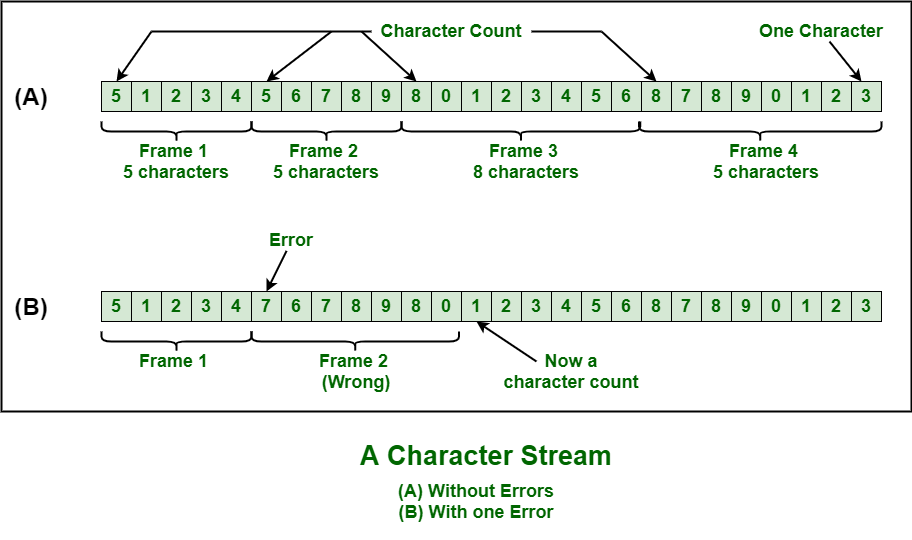
Learned about finding the shortest path using dijkstra’s algorithm and implemented it in program.

***Experiment – 5***

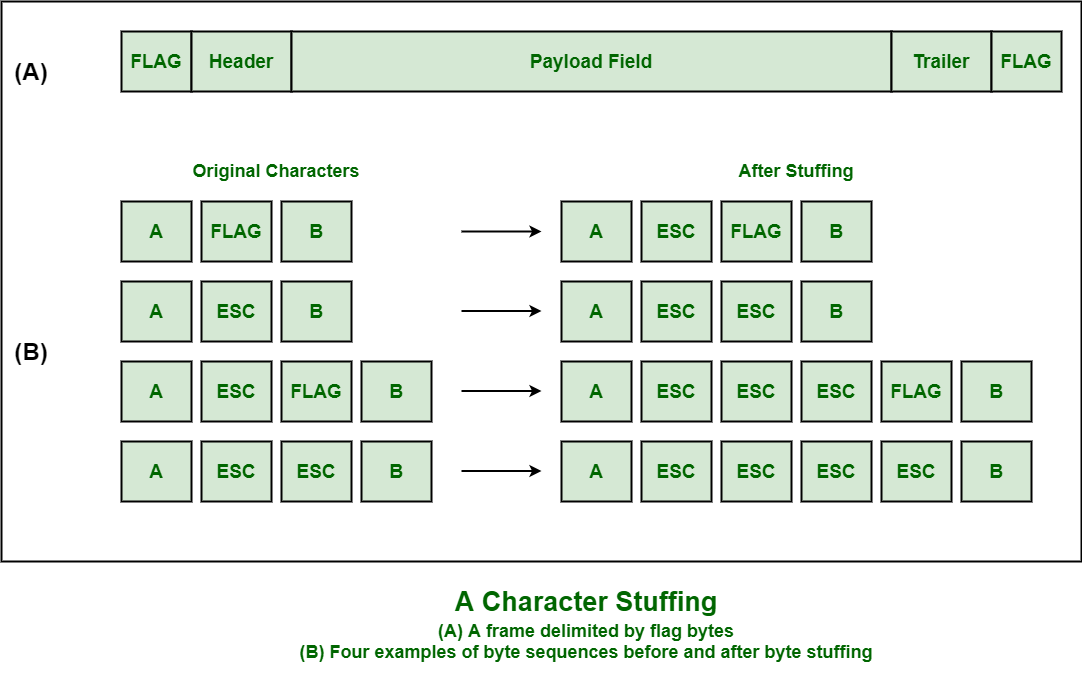
**Aim: -** To Study & implement different framing techniques.

**Theory: -**

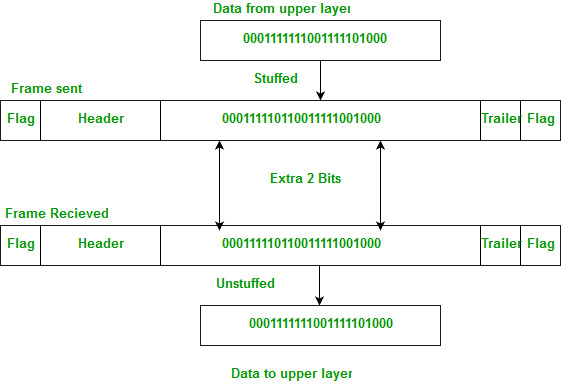
1. **Character Count:**  
   This method is rarely used and is generally required to count the total number of characters that are present in the frame. This is done by using a field in the header. The character count method ensures data link layer at the receiver or destination about the total number of characters that follow, and about where the frame ends. There is a disadvantage also of using this method i.e., if anyhow character count is disturbed or distorted by an error occurring during transmission, then the destination or receiver might lose synchronization. The destination or receiver might also be not able to locate or identify beginning of next frame.



1. [**Character Stuffing**](https://www.geeksforgeeks.org/difference-between-byte-stuffing-and-bit-stuffing/)**:**Character stuffing is also known as byte stuffing or character-oriented framing and is the same as that of bit stuffing but byte stuffing actually operates on bytes whereas bit stuffing operates on bits. In byte stuffing, a special byte that is basically known as ESC (Escape Character) that has a predefined pattern is generally added to the data section of the data stream or frame when there is a message or character that has the same pattern as that of the flag byte. But the receiver removes this ESC and keeps the data part that causes some problems or issues. In simple words, we can say that character stuffing is the addition of 1 additional byte if there is the presence of ESC or flag in text.



1. [**Bit Stuffing**](https://www.geeksforgeeks.org/bit-stuffing-in-computer-network/)**:**Bit stuffing is also known as bit-oriented framing or bit-oriented approach. In bit stuffing, extra bits are being added by network protocol designers to data streams. It is generally insertion or addition of extra bits into transmission unit or message to be transmitted as simple way to provide and give signalling information and data to receiver and to avoid or ignore appearance of unintended or unnecessary control sequences. It is type of protocol management simply performed to break up bit pattern that results in transmission to go out of synchronization. Bit stuffing is very essential part of transmission process in network and communication protocol. It is also required in USB.



**Code: -**

import java.util.ArrayList;

import java.util.Scanner;

class CharacterCount {

*void* getFrames(*int*[] *a*) {

        ArrayList<ArrayList<Integer>> frames = new ArrayList<>();

*int* i, j, k, val;

        i = 0;

        k = 0;

        while (i < *a*.length) {

            val = *a*[i];

            frames.add(new ArrayList<>());

            for (j = 0; j < val && i < *a*.length; j++, i++) {

                frames.get(k).add(j, *a*[i]);

            }

            k++;

        }

        System.out.println("The frames are: ");

        for (i = 0; i < k; i++) {

            System.out.println("Frame " + (i + 1) + ": " + frames.get(i));

        }

    }

}

class CharacterStuffing {

*void* getStuffedString(String[] *s*) {

*int* i;

        String start = "STX DLE";

        String end = "DLE ETX";

        StringBuilder stuffedString = new StringBuilder();

        stuffedString.append(start).append(" ");

        for (i = 0; i < *s*.length; i++) {

            if (*s*[i].equalsIgnoreCase("DLE")) {

                stuffedString.append("DLE").append(" ");

            }

            stuffedString.append(*s*[i]).append(" ");

        }

        stuffedString.append(end);

        System.out.println("The stuffed string is: " + stuffedString);

    }

}

class ByteStuffing {

*void* getStuffedByte(String[] *s*) {

*int* i;

        String start = "flag header";

        String end = "trailer flag";

        StringBuilder stuffedByte = new StringBuilder();

        stuffedByte.append(start).append(" ");

        for (i = 0; i < *s*.length; i++) {

            if (*s*[i].equalsIgnoreCase("esc") || *s*[i].equalsIgnoreCase("flag")) {

                stuffedByte.append("esc").append(" ");

            }

            stuffedByte.append(*s*[i]).append(" ");

        }

        stuffedByte.append(end);

        System.out.println("The stuffed byte is: " + stuffedByte);

    }

}

class BitStuffing {

*void* getStuffedBit(String *s*) {

        String flag = "01111110";

*int* i, count = 0;

        StringBuilder stuffedBit = new StringBuilder();

        stuffedBit.append(flag);

        for (i = 0; i < *s*.length(); i++) {

            if (*s*.charAt(i) == '1' && count == 5) {

                stuffedBit.append("0");

                count = 0;

            } else if (*s*.charAt(i) == '1') {

                count++;

            } else {

                count = 0;

            }

            stuffedBit.append(*s*.charAt(i));

        }

        stuffedBit.append(flag);

        System.out.println("The stuffed bit is: " + stuffedBit);

    }

}

public class Framing {

    public static *void* main(String[] *args*) {

        Scanner in = new Scanner(System.in);

*int*[] a;

*int* i, n, option;

        String[] str;

        String s;

        while (true) {

            System.out.println("""

                    The menu options are:\s

                    1. Character Count\s

                    2. Character Stuffing\s

                    3. Byte Stuffing\s

                    4. Bit Stuffing\s

                    5. Exit""");

            System.out.println("Enter your choice: ");

            option = in.nextInt();

            switch (option) {

                case 1 *->* {

                    CharacterCount cc = new CharacterCount();

                    System.out.println("Enter the size of the data: ");

                    n = in.nextInt();

                    a = new *int*[n];

                    System.out.println("Enter the data: ");

                    for (i = 0; i < n; i++) {

                        a[i] = in.nextInt();

                    }

                    cc.getFrames(a);

                }

                case 2 *->* {

                    CharacterStuffing cs = new CharacterStuffing();

                    System.out.println("Enter the size of the data: ");

                    n = in.nextInt();

                    str = new String[n];

                    System.out.println("Enter the data: ");

                    for (i = 0; i < n; i++) {

                        str[i] = in.next();

                    }

                    cs.getStuffedString(str);

                }

                case 3 *->* {

                    ByteStuffing bs = new ByteStuffing();

                    System.out.println("Enter the size of the data: ");

                    n = in.nextInt();

                    str = new String[n];

                    System.out.println("Enter the data: ");

                    for (i = 0; i < n; i++) {

                        str[i] = in.next();

                    }

                    bs.getStuffedByte(str);

                }

                case 4 *->* {

                    BitStuffing bs = new BitStuffing();

                    System.out.println("Enter the string: ");

                    s = in.next();

                    bs.getStuffedBit(s);

                }

                case 5 *->* {

                    System.out.println("Thank you for using!!");

                    in.close();

                    System.exit(0);

                }

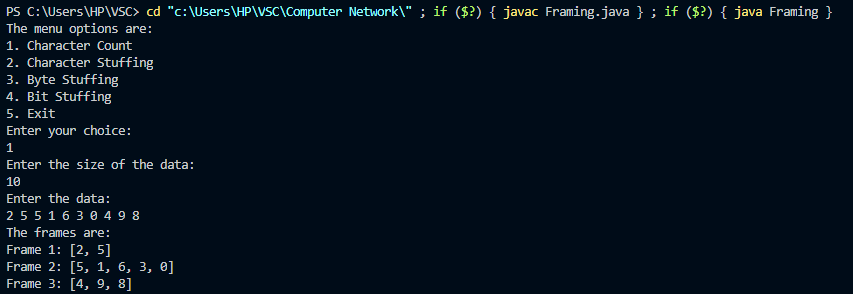
            }

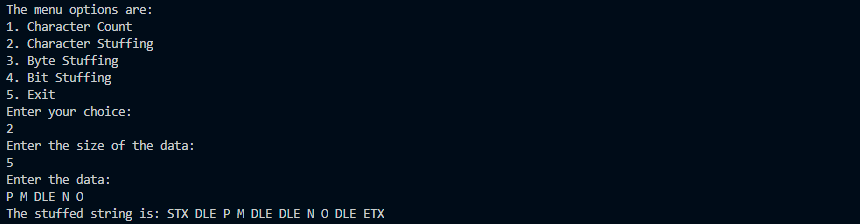
        }

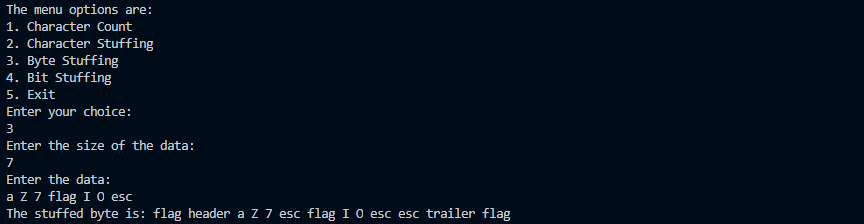
    }

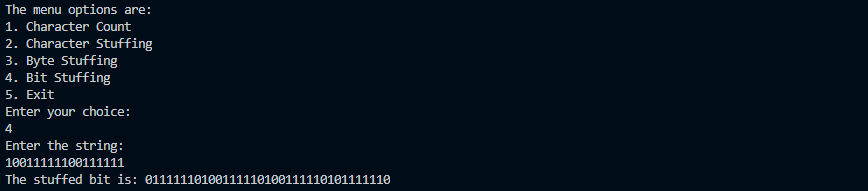
}

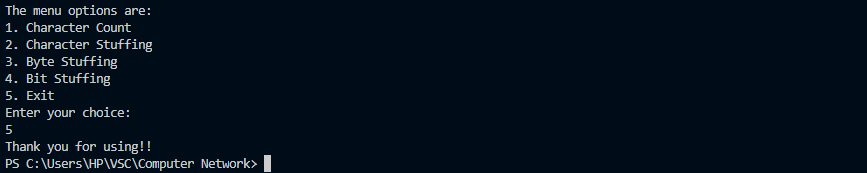
**Output: -**

****

****

****

****

****

**Conclusion: -**

Learned about various framing techniques and implemented them in the program.

***Experiment – 6***

**Aim: -** To implement socket communication in Java.

**Theory: -**

A **socket** is one endpoint of a **two-way** communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication takes place.

Like ‘Pipe’ is used to create pipes and sockets are created using the **‘socket’** system call. The socket provides a bidirectional **FIFO** Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.

Socket are generally employed in client-server applications. The server creates a socket, attaches it to a network port address then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, the transfer of data takes place.

There are two types of Sockets: the **datagram** socket and the **stream** socket.

1. **Datagram Socket:**  
   This is a type of network which has connection-less points for sending and receiving packets. It is similar to a mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).
2. **Stream Socket:**  
   In Computer operating system, a stream socket is a type of [interprocess communications](https://www.geeksforgeeks.org/inter-process-communication-ipc/) socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well-defined mechanisms for creating and destroying connections and for detecting errors. It is similar to a phone. A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.
   1. **TCP Socket:**

**Code: -**

1. Server

import java.io.\*;

import java.net.\*;

class ServerTCP

{

    public static *void* main(String[] *args*) throws Exception

    {

        String msg;

        ServerSocket ss = new ServerSocket(80);

        while(true)

        {

            Socket s1 = ss.accept();

            String[] m = {"M", "T", "W", "Th", "F", "Sa", "Su"};

*int* i = (*int*) (Math.random() \* m.length);

            msg = m[i];

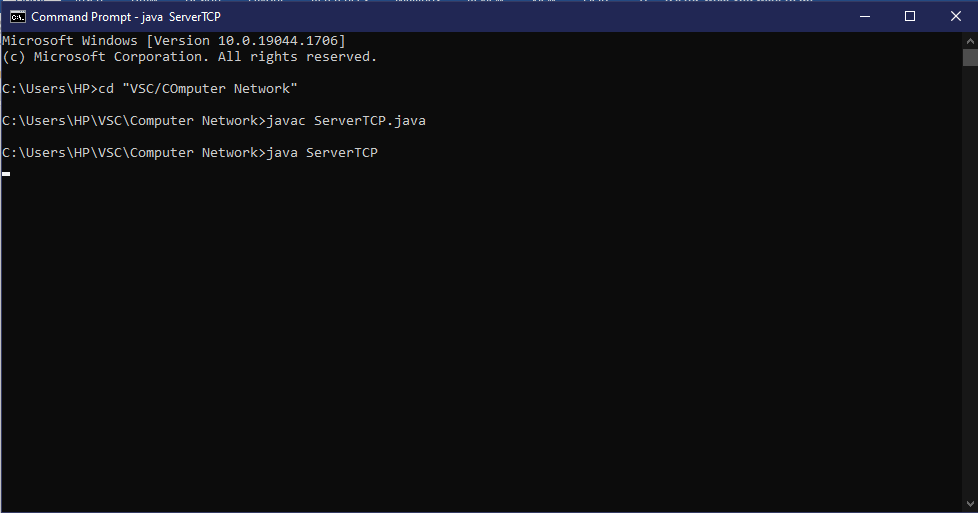
            PrintStream pr = new PrintStream(s1.getOutputStream());

            pr.println(msg);

        }

    }

}



* + - 1. Client

import java.io.\*;

import java.net.\*;

class ClientTCP

{

    public static *void* main(String[] *args*) throws Exception

    {

        Socket cs = new Socket("localhost", 80);

        BufferedReader br = new BufferedReader(new InputStreamReader(cs.getInputStream()));

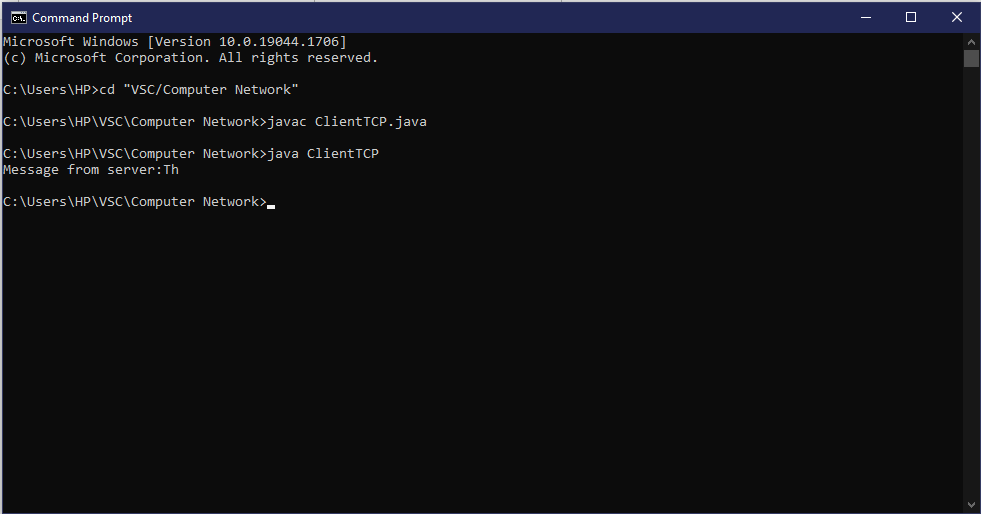
        String m = br.readLine();

        System.out.println("Message from server:" + m);

        cs.close();

    }

}



* 1. **UDP Socket:**

**Code: -**

1. Server

import java.io.\*;

import java.net.\*;

class ServerUDP

{

    public static *void* main(String[] *args*) throws Exception

    {

*byte*[] rData = new *byte*[1024];

*byte*[] sData = new *byte*[1024];

        String msg;

        DatagramSocket ss =new DatagramSocket(9876);

        while(true)

        {

            DatagramPacket rPacket = new DatagramPacket(rData, rData.length);

            ss.receive(rPacket);

            String[] m = {"Sun", "Mon", "Tue", "Wed", "Thurs", "Fri", "Sat"};

*int* i = (*int*) (Math.random() \* m.length);

            msg = m[i];

            sData = msg.getBytes();

            InetAddress ipadd = rPacket.getAddress();

*int* port = rPacket.getPort();

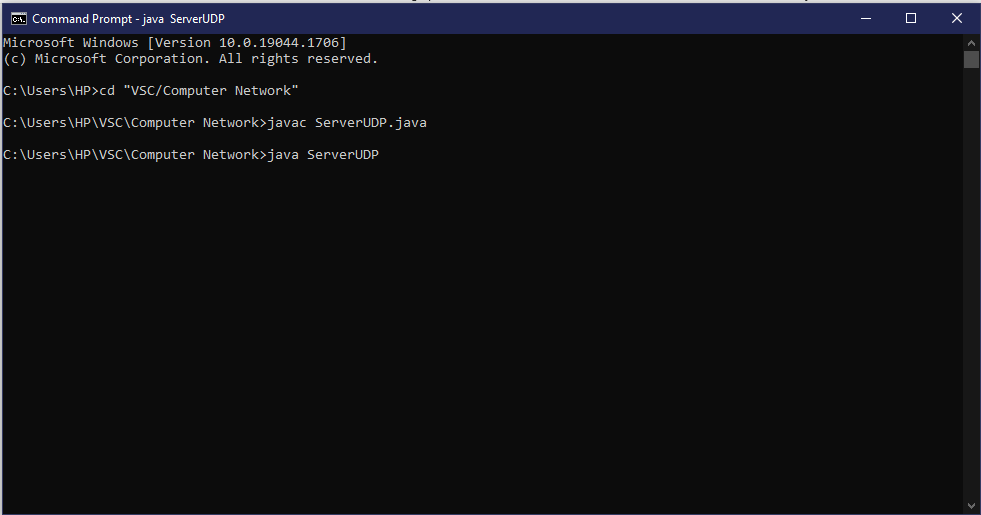
            DatagramPacket sPacket = new DatagramPacket(sData, sData.length, ipadd, port);

            ss.send(sPacket);

        }

    }

}



1. Client:

import java.io.\*;

import java.net.\*;

class ClientUDP

{

    public static *void* main(String[] *args*) throws Exception

    {

*byte*[] rData = new *byte*[1024];

*byte*[] sData = new *byte*[1024];

        String msg = " ";

        DatagramSocket cs = new DatagramSocket();

        sData = msg.getBytes();

        InetAddress ipadd = InetAddress.getByName("localhost");

        DatagramPacket sPacket = new DatagramPacket(sData, sData.length, ipadd, 9876);

        cs.send(sPacket);

        DatagramPacket rPacket = new DatagramPacket(rData, rData.length);

        cs.receive(rPacket);

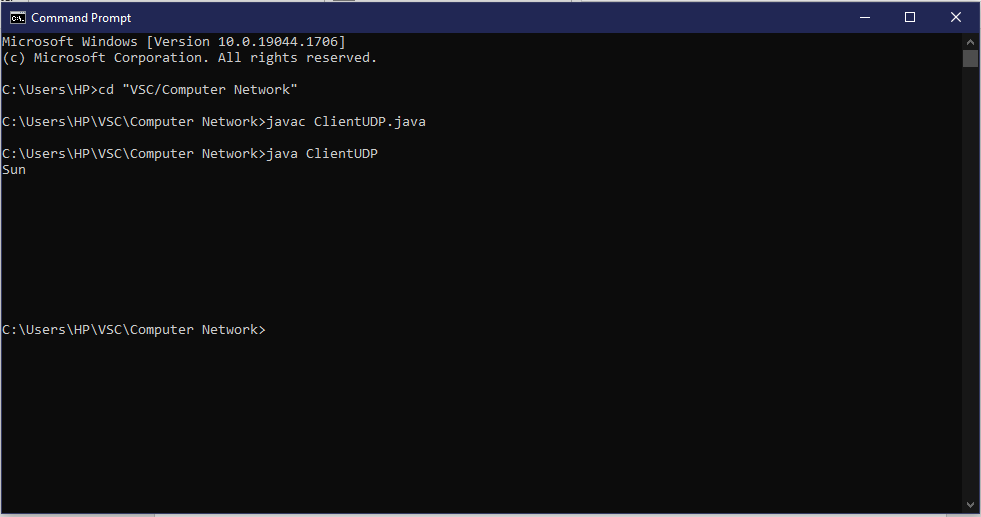
        msg = new String(rPacket.getData());

        System.out.println(msg);

        cs.close();

    }

}



**Conclusion: -**

Learned different types of socket programming in java and implemented them using a simple example.

***Experiment – 7***

**Aim: -** Creation of Duplex links in ns2 between two nodes.

**Code: -**

#===================================

# Simulation parameters setup

#===================================

set val(stop) 10.0 ;# time of simulation end

#===================================

# Initialization

#===================================

#Create a ns simulator

set ns [new Simulator]

#Open the NS trace file

set tracefile [open out.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open out.nam w]

$ns namtrace-all $namfile

#===================================

# Nodes Definition

#===================================

#Create 2 nodes

set n0 [$ns node]

set n1 [$ns node]

#===================================

# Links Definition

#===================================

#Createlinks between nodes

$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 50

#Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right

#===================================

# Agents Definition

#===================================

#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n1 $sink1

$ns connect $tcp0 $sink1

$tcp0 set packetSize\_ 1500

#===================================

# Applications Definition

#===================================

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"

#===================================

# Termination

#===================================

#Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile

exec nam out.nam &

exit 0

}

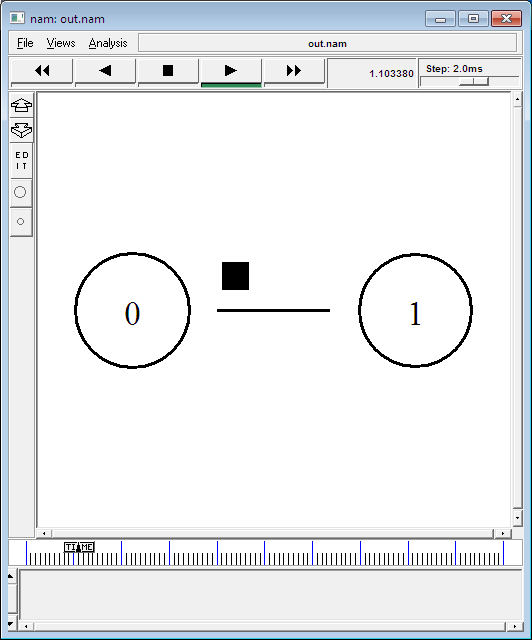
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run

**Output: -**



**Conclusion: -**

Learned how to create duplex links in NS2 and got to know how it works with the help of simulation.

***Experiment – 8***

**Aim: -** Creation of TCP and UDP in NS2.

**Code: -**

#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the NAM trace file

close $nf

#Execute NAM on the trace file

exec nam out.nam &

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 10

#Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient right-down

$ns duplex-link-op $n1 $n2 orient right-up

$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection

set tcp [new Agent/TCP]

$tcp set class\_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid\_ 1

#Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type\_ FTP

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid\_ 2

#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type\_ CBR

$cbr set packet\_size\_ 1000

$cbr set rate\_ 1mb

$cbr set random\_ false

#Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Print CBR packet size and interval

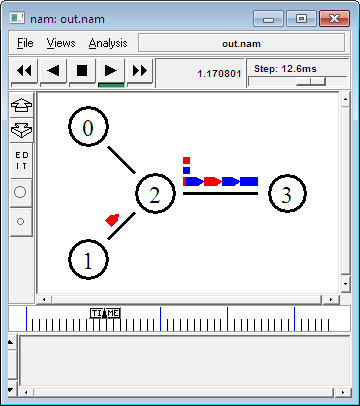
puts "CBR packet size = [$cbr set packet\_size\_]"

puts "CBR interval = [$cbr set interval\_]"

#Run the simulation

$ns run

**Output: -**



**Conclusion: -**

Learned how to create TCP and UDP in NS2 and got to know how it works with the help of simulation.

***Experiment – 9***

**Aim: -** Creation of Stop and Wait in NS2.

**Code: -**

# stop and wait protocol in normal situation

# features : labeling, annotation, nam-graph, and window size monitoring

set ns [new Simulator]

set n0 [$ns node]

set n1 [$ns node]

$ns at 0.0 "$n0 label Sender"

$ns at 0.0 "$n1 label Receiver"

set nf [open A1-stop-n-wait.nam w]

$ns namtrace-all $nf

set f [open A1-stop-n-wait.tr w]

$ns trace-all $f

$ns duplex-link $n0 $n1 0.2Mb 200ms DropTail

$ns duplex-link-op $n0 $n1 orient right

$ns queue-limit $n0 $n1 10

Agent/TCP set nam\_tracevar\_ true

set tcp [new Agent/TCP]

$tcp set window\_ 1

$tcp set maxcwnd\_ 1

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ns add-agent-trace $tcp tcp

$ns monitor-agent-trace $tcp

$tcp tracevar cwnd\_

$ns at 0.1 "$ftp start"

$ns at 3.0 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n1 $sink"

$ns at 3.5 "finish"

$ns at 0.0 "$ns trace-annotate \"Stop and Wait with normal operation\""

$ns at 0.05 "$ns trace-annotate \"FTP starts at 0.1\""

$ns at 0.11 "$ns trace-annotate \"Send Packet\_0\""

$ns at 0.35 "$ns trace-annotate \"Receive Ack\_0\""

$ns at 0.56 "$ns trace-annotate \"Send Packet\_1\""

$ns at 0.79 "$ns trace-annotate \"Receive Ack\_1\""

$ns at 0.99 "$ns trace-annotate \"Send Packet\_2\""

$ns at 1.23 "$ns trace-annotate \"Receive Ack\_2 \""

$ns at 1.43 "$ns trace-annotate \"Send Packet\_3\""

$ns at 1.67 "$ns trace-annotate \"Receive Ack\_3\""

$ns at 1.88 "$ns trace-annotate \"Send Packet\_4\""

$ns at 2.11 "$ns trace-annotate \"Receive Ack\_4\""

$ns at 2.32 "$ns trace-annotate \"Send Packet\_5\""

$ns at 2.55 "$ns trace-annotate \"Receive Ack\_5 \""

$ns at 2.75 "$ns trace-annotate \"Send Packet\_6\""

$ns at 2.99 "$ns trace-annotate \"Receive Ack\_6\""

$ns at 3.1 "$ns trace-annotate \"FTP stops\""

proc finish {} {

global ns nf

$ns flush-trace

close $nf

puts "filtering..."

exec tclsh ../ns-allinone-2.1b5/nam-1.0a7/bin/namfilter.tcl A1-stop-n-wait.nam

puts "running nam..."

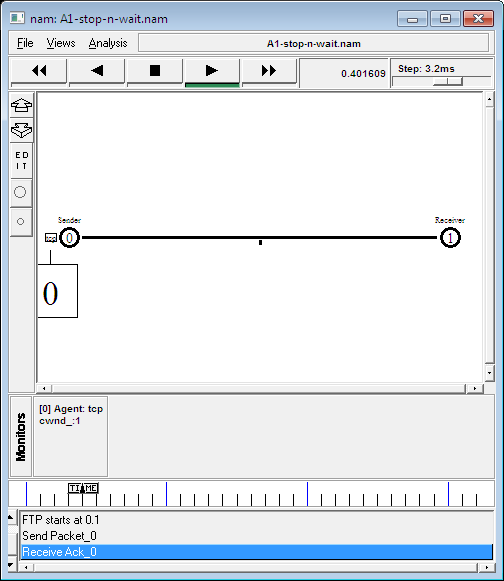
exec nam A1-stop-n-wait.nam &

exit 0

}

$ns run

**Output: -**



**Conclusion: -**

Learned how to create Stop and Wait in NS2 and got to know how it works with the help of simulation.

***Experiment – 10***

**Aim: -** To Implement RIP in packet tracer.

**Introduction: -**

Routing Information Protocol (RIP) is a dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance-vector routing protocol that has an AD value of 120 and works on the Network layer of the OSI model. RIP uses port number 520.

**Hop Count:** Hop count is the number of routers occurring in between the source and destination network. The path with the lowest hop count is considered as the best route to reach a network and therefore placed in the routing table. RIP prevents routing loops by limiting the number of hops allowed in a path from source and destination. The maximum hop count allowed for RIP is 15 and a hop count of 16 is considered as network unreachable.

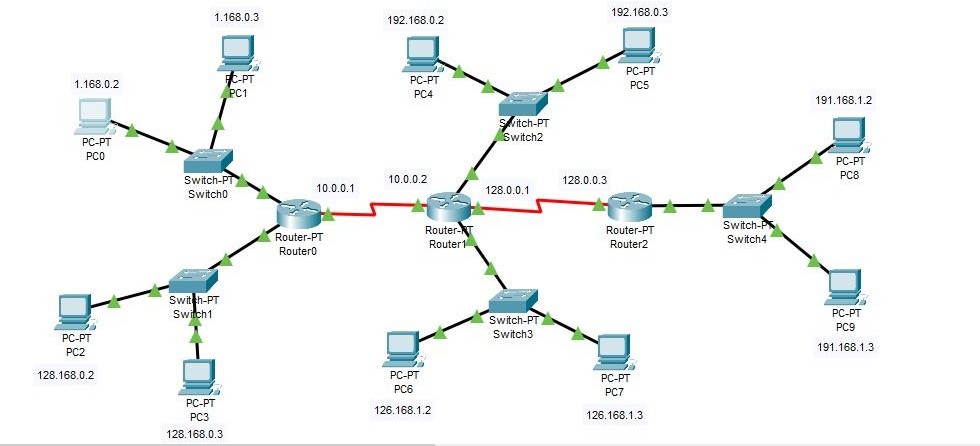
**Features of RIP**

1. Updates of the network are exchanged periodically.
2. Updates (routing information) are always broadcast.
3. Full routing tables are sent in updates.
4. Routers always trust routing information received from neighbour routers. This is also known as *Routing on*rumours.

**RIP versions: -**

There are three versions of routing information protocol – RIP Version1, RIP Version2, and RIPng.

**Diagram: -**



**Output: -**

Packet Tracer PC Command Line 1.0 C:\>ping 128.168.0.2

Pinging 128.168.0.2 with 32 bytes of data: Request timed out.

Reply from 128.168.0.2: bytes=32 time<1ms TTL=127 Reply from 128.168.0.2: bytes=32 time=16ms TTL=127 Reply from 128.168.0.2: bytes=32 time=1ms TTL=127 Ping statistics for 128.168.0.2:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 0ms, Maximum = 16ms, Average = 5ms

C:\>ping 128.168.0.3

Pinging 128.168.0.3 with 32 bytes of data: Request timed out.

Reply from 128.168.0.3: bytes=32 time=11ms TTL=127 Reply from 128.168.0.3: bytes=32 time<1ms TTL=127 Reply from 128.168.0.3: bytes=32 time<1ms TTL=127

Ping statistics for 128.168.0.3:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 0ms, Maximum = 11ms, Average = 3ms

C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data: Request timed out.

Reply from 192.168.0.2: bytes=32 time=1ms TTL=126 Reply from 192.168.0.2: bytes=32 time=1ms TTL=126 Reply from 192.168.0.2: bytes=32 time=2ms TTL=126 Ping statistics for 192.168.0.2:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\>ping 192.168.0.3

Pinging 192.168.0.3 with 32 bytes of data: Request timed out.

Reply from 192.168.0.3: bytes=32 time=1ms TTL=126 Reply from 192.168.0.3: bytes=32 time=15ms TTL=126 Reply from 192.168.0.3: bytes=32 time=1ms TTL=126 Ping statistics for 192.168.0.3:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 1ms, Maximum = 15ms, Average = 5ms

C:\>ping 126.168.1.2

Pinging 126.168.1.2 with 32 bytes of data: Request timed out.

Reply from 126.168.1.2: bytes=32 time=1ms TTL=126 Reply from 126.168.1.2: bytes=32 time=14ms TTL=126 Reply from 126.168.1.2: bytes=32 time=2ms TTL=126 Ping statistics for 126.168.1.2:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 1ms, Maximum = 14ms, Average = 5ms

C:\>ping 126.168.1.3

Pinging 126.168.1.3 with 32 bytes of data: Request timed out.

Reply from 126.168.1.3: bytes=32 time=1ms TTL=126 Reply from 126.168.1.3: bytes=32 time=24ms TTL=126 Reply from 126.168.1.3: bytes=32 time=15ms TTL=126 Ping statistics for 126.168.1.3:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds: Minimum = 1ms, Maximum = 24ms, Average = 13ms

C:\>ping 191.168.1.2

Pinging 191.168.1.2 with 32 bytes of data: Request timed out.

Reply from 191.168.1.2: bytes=32 time=2ms TTL=125 Reply from 191.168.1.2: bytes=32 time=13ms TTL=125

Reply from 191.168.1.2: bytes=32 time=2ms TTL=125 Ping statistics for 191.168.1.2:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds:

Minimum = 2ms, Maximum = 13ms, Average = 5ms

C:\>ping 191.168.1.3

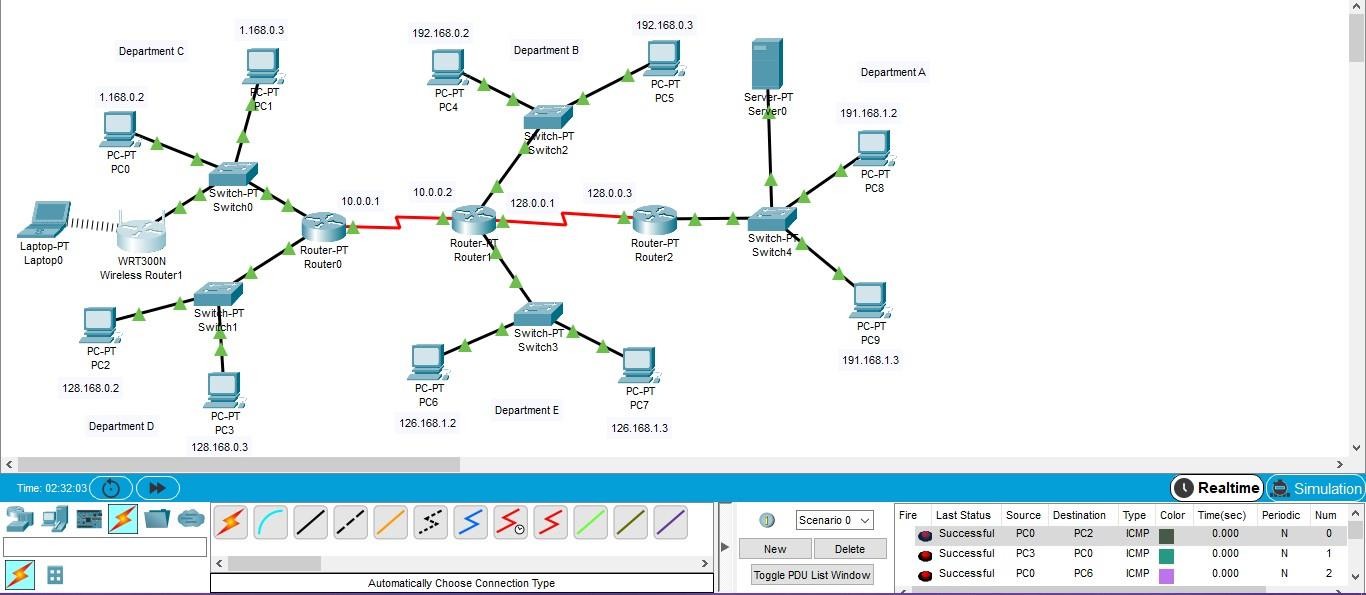
Pinging 191.168.1.3 with 32 bytes of data:

Request timed out.

Reply from 191.168.1.3: bytes=32 time=5ms TTL=125 Reply from 191.168.1.3: bytes=32 time=4ms TTL=125 Reply from 191.168.1.3: bytes=32 time=2ms TTL=125 Ping statistics for 191.168.1.3:

Packets: Sent = 4, Received = 3, Lost = 1 (25% loss), Approximate round trip times in milli-seconds:

Minimum = 2ms, Maximum = 5ms, Average = 3ms C:\>



**Conclusion: -**

Learned about RIP Packet Tracing and implemented using Cisco Packet Tracer.