Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.15)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Prerna Sunil Jadhav SAP ID: 60004220127

Course: Computer Networks Course Code: DJ19CEL405

Year: S.Y. B.Tech. Sem: IV Batch: TT

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	1	1	2	2	4	3	4	4	5	4	6						
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5	5	5	5	5	5					5	5		5
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)	5	5	5	5	5	5	5	5	5					5	5		5
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)	5	5	4	5	5	5	4	5	4					5	5		4
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)	5	4	5	4	5	3	5	3	5					5	5		4
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–	–	–	–	–	–					–	–		
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)	6	5	4	4	5	3	3	3	3					5	5		4
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–	–	–	–	–	–					–	–		
Total	25	24	23	23	25	23	23	23	23					25	25		24
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member:

Signature of Head of the Department
Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with 'A' Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Prerna Sunil Jadhav SAP ID: 60004220127

Course: Computer Networks Course Code: DJ19CEL405

Year: S.Y. B.Tech. Sem: IV Batch: TT1

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	1	1	2	2	4	3	4	4	5	4	6						
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)																	
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)																	
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)																	
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)																	
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–	–	–	–	–	–					–	–		
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)																	
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–	–	–	–	–	–					–	–		
Total	25	24															
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member

Signature of Head of the Department
Date:

NAME: PRERNA SUNIL JADHAV
SAP ID: 60004220127

CLASS: S.Y. B.Tech (COMPUTER ENGINEERING)
COURSE: COMPUTER NETWORKS (DJ19CEL405)

DATE OF PERFORMANCE: 21 FEB 2023
DATE OF SUBMISSION: 22 FEB 2023

EXPERIMENT No.: 1

AIM: To study different networking devices and topologies.

Theory:

A) NETWORKING DEVICES:

1) Repeater:

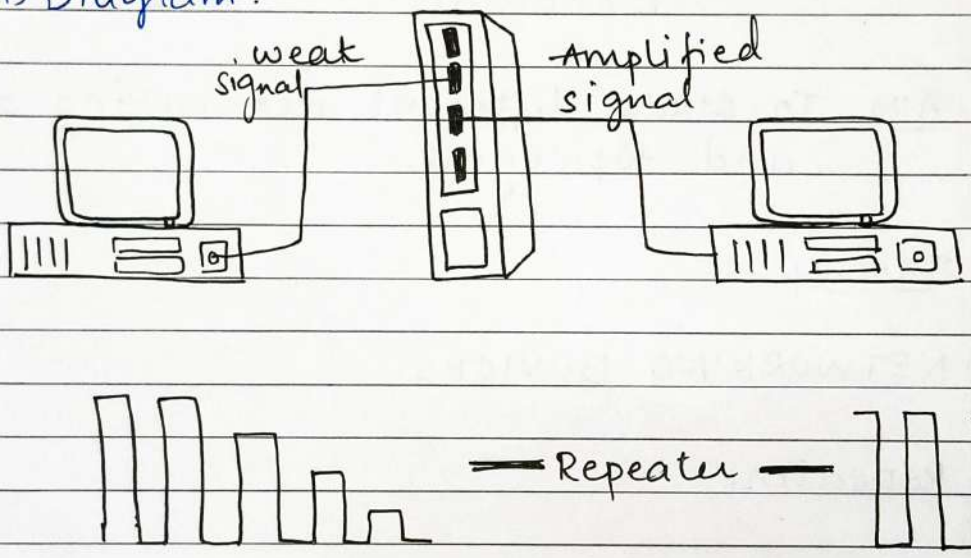
(i) Introduction

- A repeater is a network device that retransmits a received signal with more power and to an extended geographical or topological network boundary than what would be capable with the original signal.
- It is also known as a signal booster.
- The incoming data can be in optical, wireless

on electrical signals.

- It works at the physical layer of the OSI model
- Repeaters are two-port devices that are used for longer-distance data transmission without compromising data security or quality.
- when the incoming signals are attenuated it copies them bit by bit & retransmits them

(ii) Diagram:



(iii) Advantages:

- They provide signal strength
- They are cheap as well as easy to use
- They have no impact on the network's performance.
- They are capable of retransmitting data and boosting weak signals.

(iv) Disadvantages:

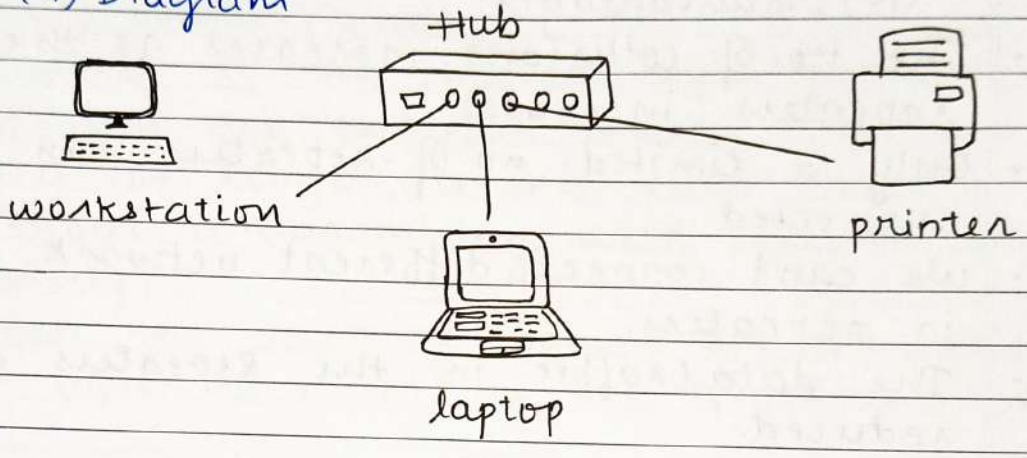
- The no. of collisions increases as the no. of repeaters increases.
- Only a limited no. of repeaters can be connected
- We cant connect different network architecture in repeaters
- The data traffic in the Repeaters cannot be reduced.

2) Hub:

(i) Introduction

- A hub sends data packets to all devices on a network, regardless of any MAC addresses contained in the data packet.
- The main purpose of a hub is to connect all present network devices together on a predefined internal network.
- Hub is not considered to be an intelligent one because it doesnt filter any data present or has no intelligence to assume as to where the data is actually supposed to be sent, & thats the reason because the only thing a hub knows is that when a device is actually connected to one of its port.

(ii) Diagram



(iii) Advantages

- Ability to connect to the network using various physical devices
- Cheaper compared to other devices
- Causes minimum delay
- Doesnt affect the network's performance

(iv) Disadvantages

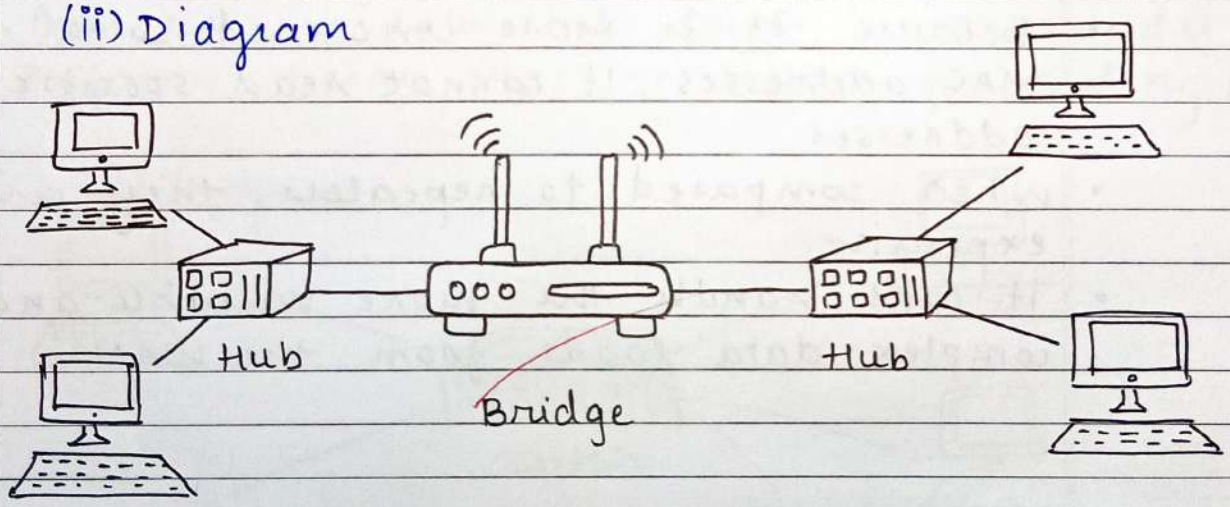
- Doesnt filter the data, hence wastage of bandwidth
- Makes the network insecure by sharing data to all devices present in the network
- Involves flooding of data.
- Cannot connect token ring.

3) Bridge:

(i) Introduction

- A bridge is a network device that connects two or more LANs (local area network) to form a larger LAN.
- Network bridging refers to the process of aggregating networks.
- A bridge connects various components, making them appear to be part of a single network.
- In the OSI Model, a bridge operates at level 2 or the data link layer.
- This is primarily to inspect incoming traffic and determine whether it should be filtered or forwarded.

(ii) Diagram



(iii) Advantages

- It serves as a network extender by acting as a repeater.
- subdividing network traffic into network communication can reduce network traffic on a segment.
- collisions can be reduced
- It reduces Bandwidth wastage, as fewer network nodes share a collision domain, bridges increase the available bandwidth to individual nodes.

(iv) Disadvantages

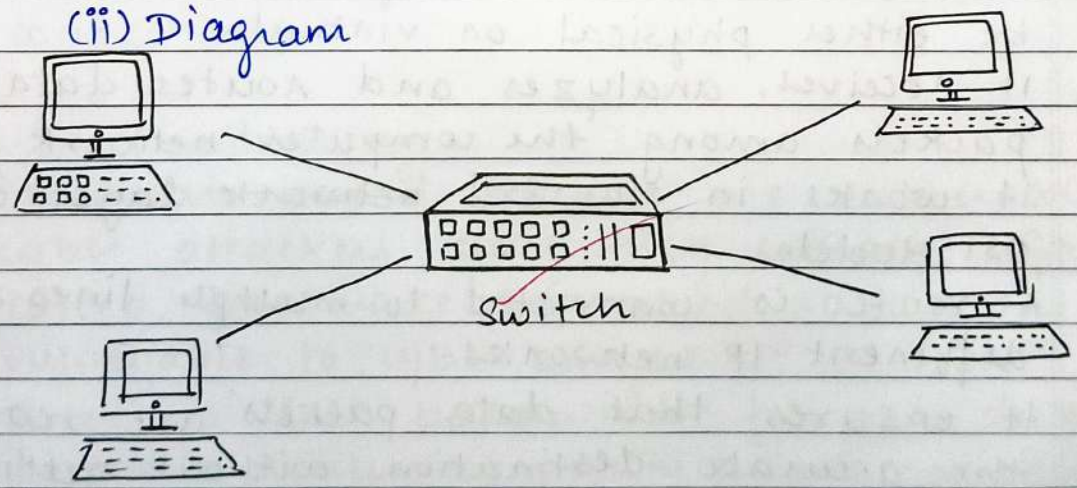
- It cannot assist in the construction of the network between the various network architectures.
- Because it is more concerned with the MAC addresses, it cannot read specific IP addresses.
- when compared to repeaters, they are expensive.
- It cant handle the more variable and complex data loads from the WAN.

4) Switch:

(i) Introduction

- It is used to segment the networks into different subnetworks called subnets or LAN segment.
- It is responsible for filtering and forwarding the packets between LAN segments based on MAC address.
- When a switch receives data, it determines the destination and sends it directly to that devices. It doesnt broadcast the packets to all computers as hub does, which means bandwidth is not shared & make network more efficient. That is why, switches are more intelligent and preferred than a hub
- Operates in data link layer in OSI Model
- Performs Error checking before forwarding it.

(ii) Diagram



(iii) Advantages

- Prevents traffic overloading in a network by segmenting the network into smaller subnet
- Increase the bandwidth of the network.
- Less frame collision as switch creates the collision domain for each connection.

(iv) Disadvantages

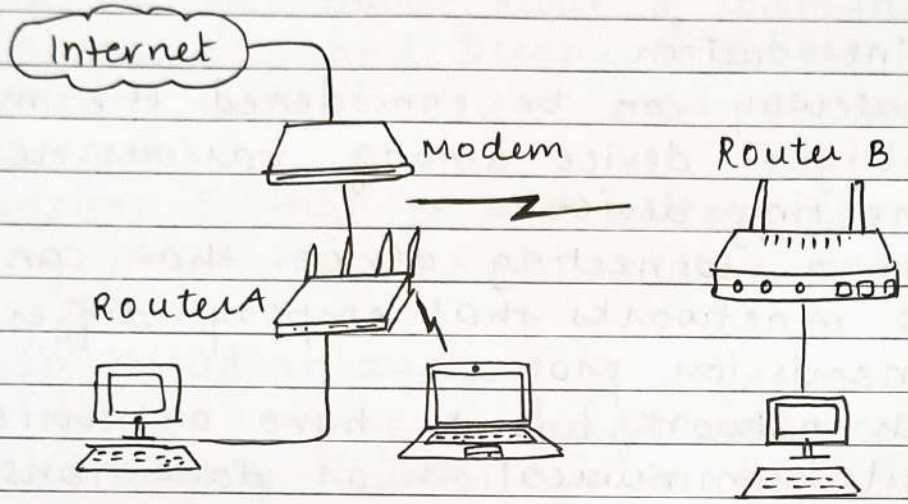
- They are more expensive than network bridges
- Network connectivity issues on network switches are difficult to trace.
- They do not work very well when used as a diversion to limit broadcasts.

5) Router:

(i) Introduction

- A router is a networking device that can be either physical or virtual
- It receives, analyzes and routes data packets among the computer network & devices.
- It works in the 3rd ~~network~~ layer of OSI Model.
- A router is connected to multiple lines from different IP networks.
- It ensures that data packets are reaching the accurate destination without getting lost.

(ii) Diagram



(iii) Advantages

- Determines the most efficient path between source and destination using dynamic routing algorithms such as OSPF, BGP, RIP etc.
- Provides connection among different network architecture
- Connects multiple users to a single network connection.

(iv) Disadvantages

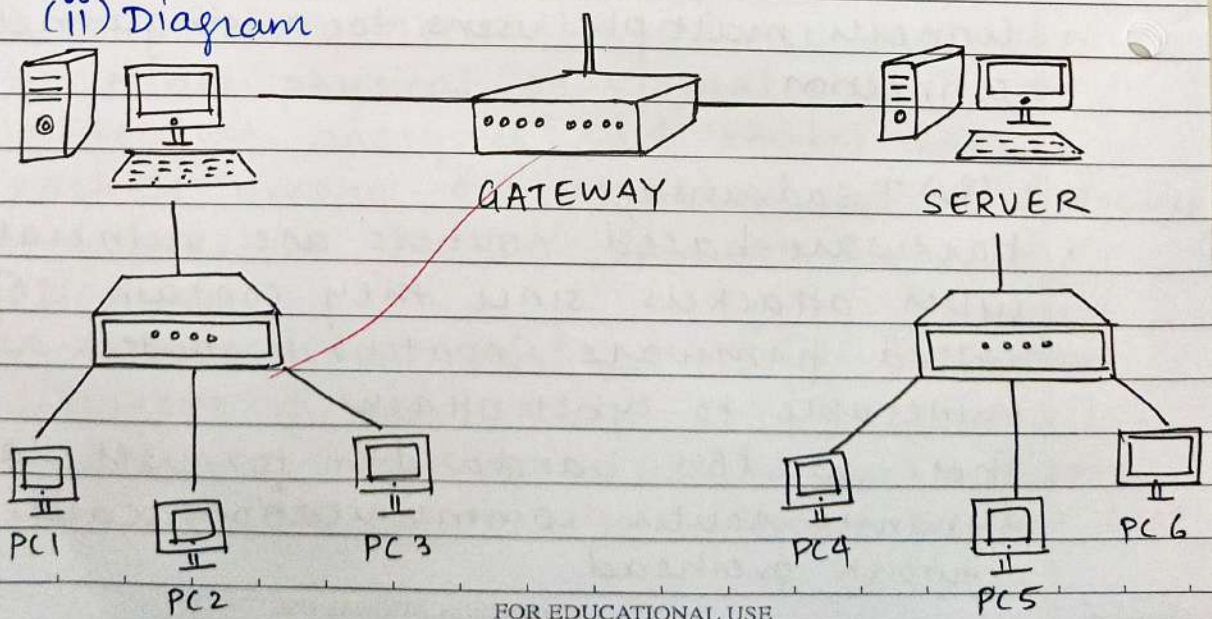
- Hardware-based routers are vulnerable to cyber attackers since they contain software called firmware. Unpatched routers are quite vulnerable to cyber attacks.
- There is less bandwidth for user data since dynamic router communications cause additional network overhead.

6) Gateway

(i) Introduction

- A Gateway can be considered the most intelligent device among various network connecting devices.
- It is a connecting device that can connect two ~~m~~ networks that employ different transmission protocols.
- Each network has to have a barrier that limits communication or data transmission to only connected devices. As a result, a gateway is required if a network needs to interact with network devices beyond its boundary.
- It is located at the network's edge & administers all traffic or data directed in/out

(ii) Diagram



(iii) Advantages

- Uses a full duplex mode of communication
- Encapsulates and Decapsulates the packets.
- controls both collisions and the broadcast domain
- Consists of tighter and better security than any other network connection device.

(iv) Disadvantages

- It is challenging to design and implement.
- It is highly costly due to high implementation cost.
- A particular system administration setup is required.

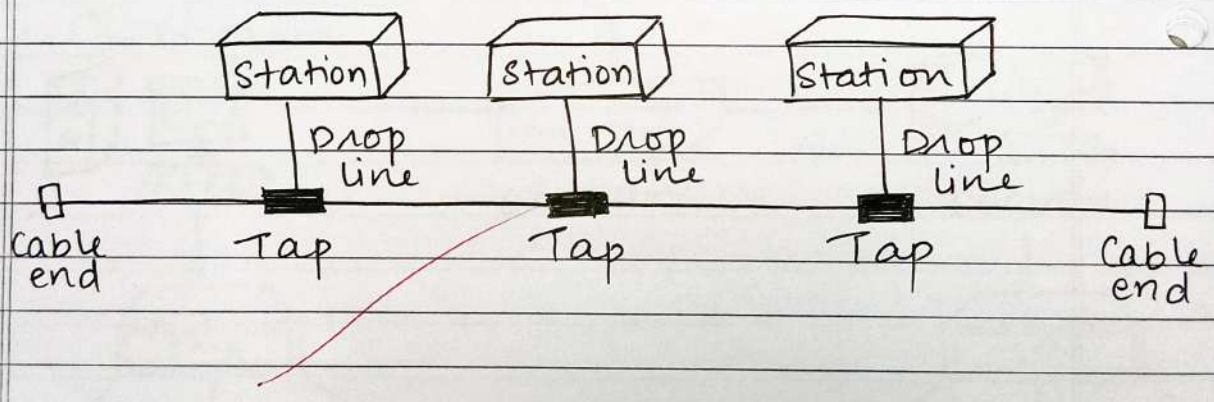
B) NETWORKING TOPOLOGY:

1) Bus:

(i) Introduction

- The bus topology is designed in such a way that all the stations are connected through a single cable known as 'backbone cable'.
- Each node is either connected to the backbone cable by the 'drop line' or directly connected to the backbone cable.
- When a node wants to send a message over network, it puts a message over network
- All stations available will be received the message whether its addressed or not.

(ii) Diagram



(iii) Advantages

- Its easy to install
- Because of backbone, less cable is required.
- No. of I/O ports required is less. Also the hardware is reduced.
- Backbone can be extended by using repeaters.
- cost is low as well.

(iv) Disadvantages:

- Heavy traffic can slow a bus considerably.
- Difficult for reconnection, fault isolation or troubleshooting.
- Difficult to add new node / device.
- Failure of backbone affects failure of all devices on network.

(v) Applications

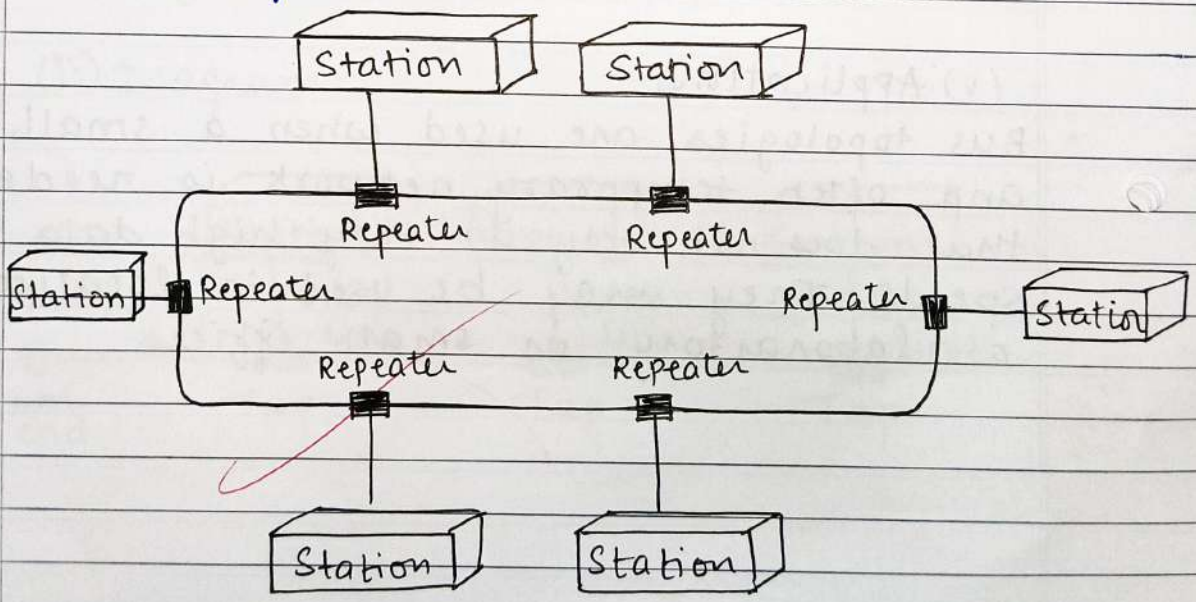
- Bus topologies are used when a small, cheap and often temporary network is needed that does not rely on very high data transfer speeds. They may be used in locations such as laboratory or small offices.

2) Ring:

(i) Introduction

- It is a topology in which each computer is linked to another on both the sides.
- The last computer is linked to first, forming a ring.
- This topology enables each computer to have exactly 2 neighbours.
- The most common access method of the ring topology is token passing.
- Token is a frame that circulates around the network.

(ii) Diagram



(iii) Advantages

- A ring topology is relatively easy to install and reconfigure
- Link failure can be easily found as each device is connected to its immediate neighbors only.
- Because every node is given equal access to the token no one node can monopolize the network.

(iv) Disadvantages

- Maximum ring length and the number of devices is limited.
- Failure of one node on the ring can affect the entire network
- Adding or Removing node disrupts the network.

(v) Applications

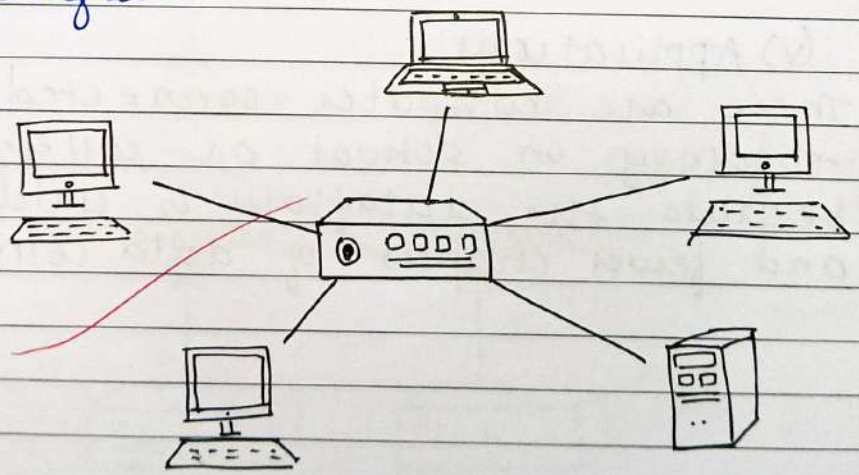
- There are computer connected in ring Topology on school or college campuses because the dataflow is unidirectional and fewer chances of data collision.

3) Star:

(i) Introduction

- Star topology is an arrangement of the network in which every node is connected to the central hub, switch or a central computer.
- The central computer is known as a server. The peripherals devices attached to it are known as clients.
- Coaxial cable or RJ-45 cables are used to connect the computers.
- Hubs or switches are mainly used as connection devices in a physical star topology.
- Star topology is the most popular one.

(ii) Diagram:



(iii) Advantages

- Each device needs only one link and one I/O port, which makes stars topology less expensive easy to install and easy to configure
- Robust topology. If any link fails it doesnt affect entire network.
- Easy fault identification and isolation
- It is easy to modify and add new nodes to star network without disturbing the rest of the network.

(iv) Disadvantages:

- If central hub fails the entire network fails to operate.
- Each device requires its own cable segment
- In heirarchical network, installation and configuration is difficult.

(v) Applications:

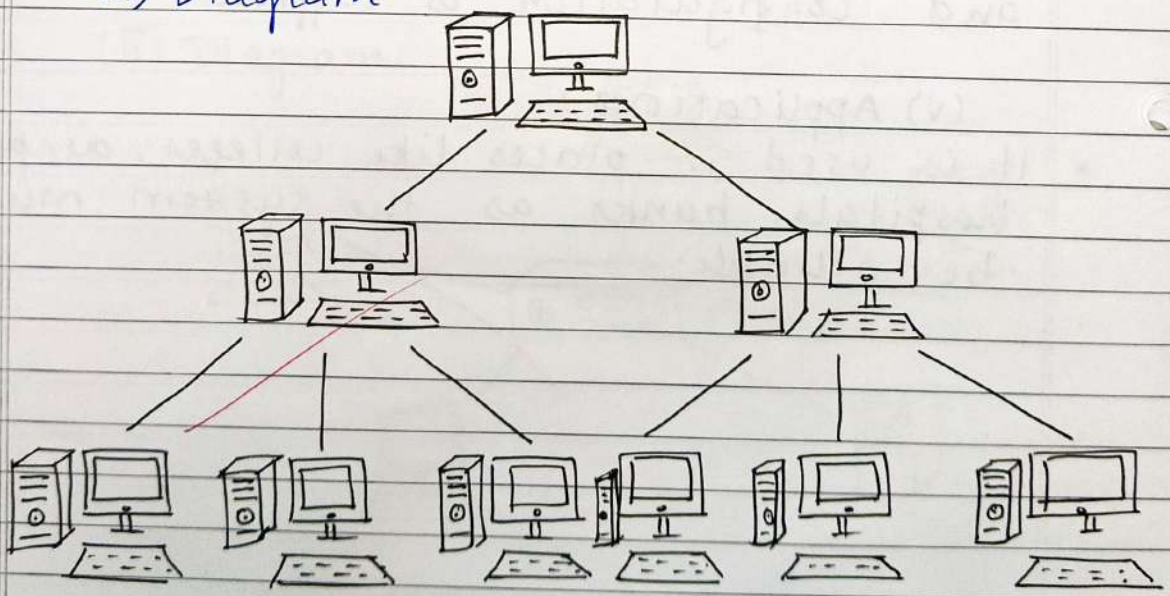
- It is used in places like colleges, airports, hospitals, banks as the system must be reliable.

4) Tree

(i) Introduction

- Tree topology are also known as hierarchical topology, as root node connects all other nodes to form a heirarchy.
- This topology is known as 'Star-Bus' topology because it combines several star topologies into a single bus.
- Data flows from top to bottom in this network topology, from the central hub to the secondary hub and then to the devices, or from bottom to top, from devices to the secondary hub, which then connects to the central hub.

(ii) Diagram



(iii) Advantages

- Support for broadband transmission, i.e., signals are sent over long distances without being attenuated.
- It is expandable i.e., we can add new devices to the existing network.
- Error detection and correction are very easy in a tree topology
- The breakdown in one station doesnt affect the entire network.

(iv) Disadvantages:

- If any fault occurs in the node, then it becomes difficult to troubleshoot the problem
- Devices required for broadband transmission are very costly.
- A tree topology mainly relies on main bus cable & failure in it will damage the overall network

(v) Applications

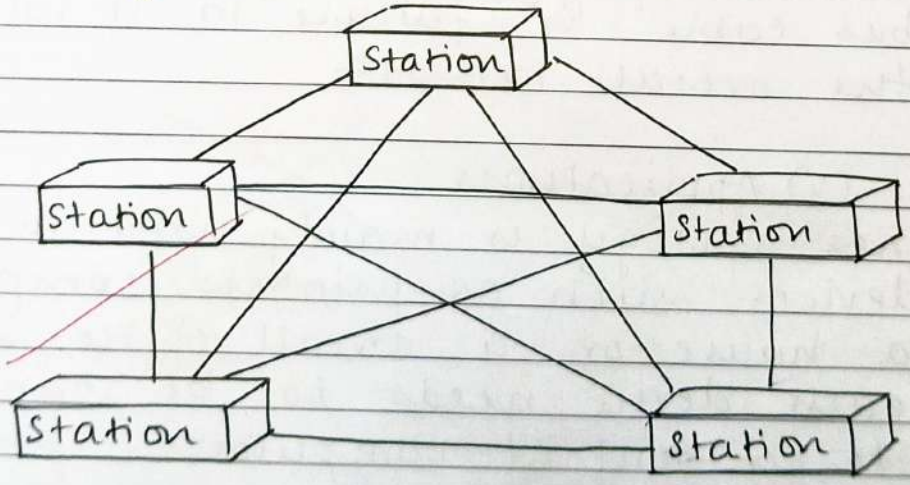
- Tree topology is mainly used to connect devices such as printers, computers in a house or a small office where each device needs to be connected to a central computer.

5) Mesh

(i) Introduction

- Mesh topology is an arrangement of the network in which computers are interconnected with each other through various redundant connections.
- There are multiple path from one computer to another computer.
- It can be formed by this formula:-
  Number of cables = $(n * (n-1))/2$,
  where n is the no. of nodes that represents the network.
- This indicates that each node must have $(n-1)$ I/O ports.

(ii) Diagram



(iii) Advantages

- No traffic because of dedicated link
- Robust because if one link fails, it does not affect the entire network.
- Privacy and security of data is achieved due to dedicated link.
- fault identification is easy.

(iv) Disadvantages

- Difficulty of installation and reconfiguration as every node is connected to every other node.
- Costly because of maintaining redundant links.
- The amount of cabling required is large

(v) Applications

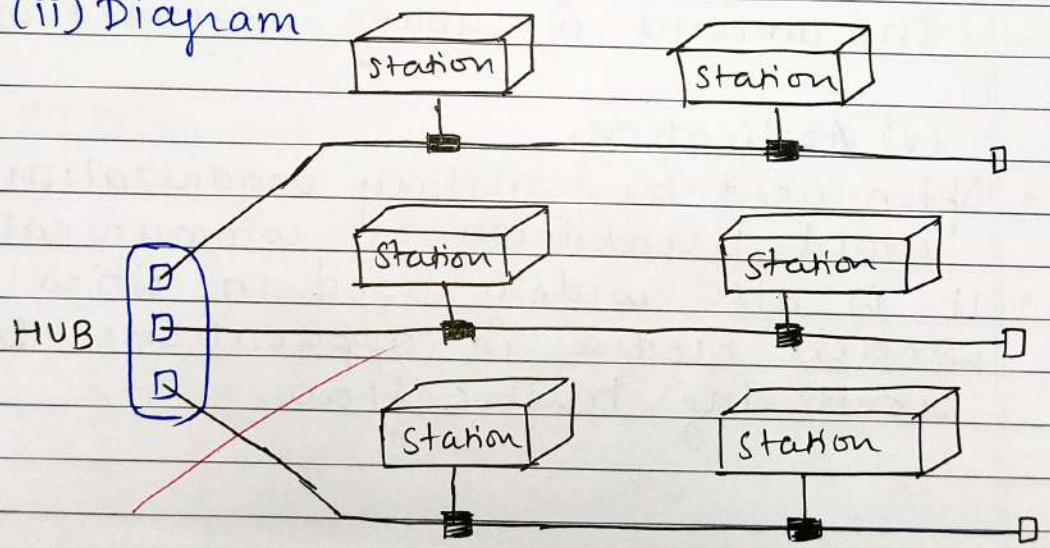
- Often used by military organization to avoid breakdown in communication
- It is also widely used in cities using wireless media in applications like monitoring traffic flow.

6) Hybrid

(i) Introduction

- When two or more different topologies are combined together is termed as Hybrid topology and if similar topologies are connected with each other it will not result in hybrid topology.
- For Example, if there exist a ring topology in one branch of HDFC bank and bus topology in another branch of HDFC bank, connecting these two topogies will result in Hybrid topology.

(ii) Diagram



(iii) Advantages

- Reliable: Fault in one part, wont affect functioning of entire network.
- Scalable: Add new elements without affecting the functionality of existing system.
- Flexible: It can be designed according to the requirements.
- Effective: The strength is maximized & the weakness of the network is minimized.

(iv) Disadvantages:

- Complex Design: It is very difficult to design the architecture of this network.
- Costly Hub: Hub used in this topology are expensive than the usual Hub
- Costly Infrastructure: As it require a lot of cabling, network devices etc.

(v) Applications

- Some of the major applications of the hybrid topology are -
  - financial and banking sector
  - automated industry
  - multinational companies
  - research organizations
  - educational institutes etc.

CONCLUSION:

Different types of networking devices and topologies are studied.

25

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with 'A' Grade (CGPA : 3.18)

Continuous Assessment for Laboratory / Assignment sessions

Academic Year 2022-23

Name: Prerna Sunil Jadhav SAP ID: 60004220127

Course: Computer Networks Course Code: DJ19CEL405

Year: S.Y. B.Tech. Sem: IV Batch: TT1

Department: Computer Engineering

Performance Indicators (Any no. of Indicators) (Maximum 5 marks per indicator)	1	2	3	4	5	6	7	8	9	10	11	Σ	Avg	A1	A2	Σ	Avg
Course Outcome	1	1	2	2	4	3	4	4	5	4	6						
1. Knowledge (Factual/Conceptual/Procedural/ Metacognitive)																	
2. Describe (Factual/Conceptual/Procedural/ Metacognitive)																	
3. Demonstration (Factual/Conceptual/Procedural/ Metacognitive)																	
4. Strategy (Analyse & / or Evaluate) (Factual/Conceptual/ Procedural/Metacognitive)																	
5. Interpret/ Develop (Factual/Conceptual/ Procedural/Metacognitive)	–	–	–	–	–	–	–	–	–					–	–		
6. Attitude towards learning (receiving, attending, responding, valuing, organizing, characterization by value)																	
7. Non-verbal communication skills/ Behaviour or Behavioural skills (motor skills, hand-eye coordination, gross body movements, finely coordinated body movements speech behaviours)	–	–	–	–	–	–	–	–	–					–	–		
Total	25	24															
Signature of the faculty member																	

Outstanding (5), Excellent (4), Good (3), Fair (2), Needs Improvement (1)

Laboratory marks Σ Avg. =	Assignment marks Σ Avg. =	Total Term-work (25) =
Laboratory Scaled to (15) =	Assignment Scaled to (10) =	Sign of the Student:

Signature of the Faculty member:
Name of the Faculty member

Signature of Head of the Department
Date:

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of performance:	28 – 02 – 2023
Date of Submission:	28 – 02 – 2023
Experiment No.:	02
Aim:	Study of Basic Networking Commands

1. ifconfig

Purpose

Configures or displays network interface parameters for a network by using TCP/IP.

Syntax

ifconfig -a [-l] [-d] [-u] [*protocolfamily*]

Description

- The ifconfig command to assign an address to a network interface and to configure or display the current network interface configuration information.
- The ifconfig command must be used at system startup to define the network address of each interface present on a system.
- After system startup, it can also be used to redefine an interfaces address and its other operating parameters.
- The network interface configuration is held on the running system and must be reset at each system restart.

Options

1. Ifconfig
   ifconfig command without any argument displays the details of all the active interfaces. This command also displays the assigned ip address of active interfaces. There could be interfaces that are active but they may not have been assigend an IP address.
2. Ifconfig –s
   display all network interfaces on the server even if the network interface is down.
3. Ifconfig –a
   display a short list, instead of all the details.
4. Ifconfig *interface_name*
   ifconfig command with interface name (wlan0) as an argument displays details of interface configuration.
5. Ifconfig *interface address*
   ifconfig command can be used with an interface name (eth0) and ip address to assign an IP address to the specific interface. Use this command with root permissions.

2. ping

Purpose

Sends an echo request to a network host.

Syntax

ping [-d] [-D] [-n] [-q] [-r] [-v] [-R] [-a addr_family] [-c Count] [-w timeout]

Description

- PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host.

- This command takes as input the IP address or the URL and sends a data packet to the specified address with the message "PING" and get a response from the server/host this time is recorded which is called latency.
- Fast ping low latency means faster connection. Ping uses ICMP(Internet Control Message Protocol) to send an ICMP echo message to the specified host if that host is available then it sends ICMP reply message.
- Ping is generally measured in millisecond every modern operating system has this ping pre-installed.

Options

1. Controlling the number of pings:
   Earlier we did not define the number of packets to send to the server/host by using -c option we can do so.
   ping -c 15 www.gmail.org
2. Controlling the size of packets send:
   Earlier a default sized packets were sent to a host but we can send light and heavy packet by using -s option.
   ping -s 40 -c 5 www.geeksforgeeks.org
3. Changing the time interval:
   By default ping wait for 1 sec to send next packet we can change this time by using -i option.
   ping -i 2 www.geeksforgeeks.org

3. Netstat

Purpose

The netstat command displays the contents of various network-related data structures for active connections.

Syntax

netstat [-a][-l]

Description

- The netstat command is used to show network status. Traditionally, it is used more for problem determination than for performance measurement.
- However, the netstat command can be used to determine the amount of traffic on the network to ascertain whether performance problems are due to network congestion.
- The netstat command displays information regarding traffic on the configured network interfaces, such as the following:
  - The address of any protocol control blocks associated with the sockets and the state of all sockets
  - The number of packets received, transmitted, and dropped in the communications subsystem
  - Cumulative statistics per interface
  - Routes and their status

Options

1. -a –all
   Show both listening and non-listening sockets. With the –interfaces option, show interfaces that are not up
2. –at
   To list all tcp ports.

3. –au
   To list all udp ports.
4. –pt
   To display the PID and program names.

4. Arp

Purpose

Displays and modifies address resolution, including ATM (Asynchronous Transfer Mode) interfaces.

Syntax

To Display ARP Entries

arp [-v] [-i if] [-H type] -a [hostname]

Description

- The arp command displays and modifies the Internet-to-adapter address translation tables used by the Address in Networks and communication management.
- The arp command displays the current ARP entry for the host specified by the HostName variable.
- The host can be specified by name or number, using Internet dotted decimal notation.

Options

1. -v, –verbose: This option shows the verbose information.
2. -n, –numeric: This option shows numerical addresses instead of symbolic host, port or usernames.
3. -H type, –hw-type type, -t type: This tells arp which class of entries it should check for. Default value is ether.

5. Nslookup

Purpose

Queries internet domain name servers interactively.

Syntax

nslookup [- option] [name | -] [server]

Description

- The nslookup command queries internet domain name servers in two modes. Interactive mode allows you to query name servers for information about various hosts and domains, or to print a list of the hosts in a domain.
- In non-interactive mode, the names and requested information are printed for a specified host or domain.

Options

1. nslookup -type=any gmail.com : Lookup for any record
   We can also view all the available DNS records using the -type=any option.
2. nslookup -type=txt ibm.com : Lookup for an txt record
   TXT records are useful for multiple types of records like DKIM, SPF, etc. You can find all TXT records configured for any domain using the below command.

6. Tracert

Purpose:

traceroute command in Linux prints the route that a packet takes to reach the host.

Syntax:

traceroute [options] host_Address [pathlength]

Description:

- We need to install it by the following command:
  sudo apt –get install traceroute
- This command is useful when you want to know about the route and about all the hops that a packet takes.
- The first column corresponds to the hop count.
- The second column represents the address of that hop and after that, you see three space-separated time in milliseconds.
- traceroute command sends three packets to the hop and each of the time refers to the time taken by the packet to reach the hop.

Options:

1. -F: Do not fragment packet
2. -m: Set the max number of hops for the packet to reach the destination. Default value is 30
3. -n: Do not resolve IP addresses to their domain names

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of performance:	28 – 02 – 2023
Date of Submission:	28 – 02 – 2023
Experiment No.:	02
Aim:	Study of Basic Networking Commands

1. Ifconfig

```
student@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e7:f5:24
          inet addr:192.168.64.128  Bcast:192.168.64.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee7:f524/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:236 errors:0 dropped:0 overruns:0 frame:0
          TX packets:214 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41319 (41.3 KB)  TX bytes:22716 (22.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:226 errors:0 dropped:0 overruns:0 frame:0
          TX packets:226 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:22126 (22.1 KB)  TX bytes:22126 (22.1 KB)

student@ubuntu:~$ ifconfig -s
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0       1500 0       240      0      0 0           218      0      0      0 BMRU
lo        65536 0       234      0      0 0           234      0      0      0 LRU
student@ubuntu:~$ ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e7:f5:24
          inet addr:192.168.64.128  Bcast:192.168.64.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee7:f524/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:240 errors:0 dropped:0 overruns:0 frame:0
          TX packets:218 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:41876 (41.8 KB)  TX bytes:23076 (23.0 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:234 errors:0 dropped:0 overruns:0 frame:0
          TX packets:234 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:22931 (22.9 KB)  TX bytes:22931 (22.9 KB)
```

```
student@ubuntu:~$ ifconfig  eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e7:f5:24
          inet addr:192.168.64.128  Bcast:192.168.64.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee7:f524/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:264 errors:0 dropped:0 overruns:0 frame:0
          TX packets:229 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:43950 (43.9 KB)  TX bytes:23976 (23.9 KB)

student@ubuntu:~$ sudo ifconfig eth0 192.162.72.228
[sudo] password for student:
student@ubuntu:~$ sudo ifconfig eth0 192.162.72.228
student@ubuntu:~$ ifconfig  eth0
eth0      Link encap:Ethernet  HWaddr 00:0c:29:e7:f5:24
          inet addr:192.162.72.228  Bcast:192.162.72.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fee7:f524/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:312 errors:0 dropped:0 overruns:0 frame:0
          TX packets:266 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:48380 (48.3 KB)  TX bytes:29247 (29.2 KB)

student@ubuntu:~$
```

2. Ping

```
student@ubuntu: ~

student@ubuntu:~$ ping www.gmail.com
PING www.gmail.com (142.250.194.5) 56(84) bytes of data.
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=1 ttl=128 time=75.8 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=2 ttl=128 time=153 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=3 ttl=128 time=147 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=4 ttl=128 time=141 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=5 ttl=128 time=137 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=6 ttl=128 time=137 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=7 ttl=128 time=125 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=8 ttl=128 time=225 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=9 ttl=128 time=139 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=10 ttl=128 time=107 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=11 ttl=128 time=180 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=12 ttl=128 time=127 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=13 ttl=128 time=176 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=14 ttl=128 time=107 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=15 ttl=128 time=137 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=16 ttl=128 time=98.4 ms
64 bytes from del12s01-in-f5.1e100.net (142.250.194.5): icmp_seq=17 ttl=128 time=165 ms
^C
--- www.gmail.com ping statistics ---
18 packets transmitted, 17 received, 5% packet loss, time 17030ms
rtt min/avg/max/mdev = 75.822/140.231/225.260/33.868 ms
student@ubuntu:~$ ping -c 10 www.ibm.com
PING e7817.dscx.akamaiedge.net (23.57.199.57) 56(84) bytes of data.
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=1 ttl=128 time=58.6 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=2 ttl=128 time=179 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=3 ttl=128 time=57.3 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=4 ttl=128 time=96.7 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=5 ttl=128 time=100 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=6 ttl=128 time=127 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=7 ttl=128 time=99.9 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=8 ttl=128 time=193 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=9 ttl=128 time=107 ms
64 bytes from a23-57-199-57.deploy.static.akamaitechnologies.com (23.57.199.57): icmp_seq=10 ttl=128 time=81.1 ms

--- e7817.dscx.akamaiedge.net ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 57.357/110.299/193.789/43.256 ms
```

```
student@ubuntu: ~
student@ubuntu:~$ ping -s 40 -i 2 -c 3 www.google.com
PING www.google.com (142.250.194.132) 40(68) bytes of data.
48 bytes from del12s05-in-f4.1e100.net (142.250.194.132): icmp_seq=1 ttl=128 time=172 ms
48 bytes from del12s05-in-f4.1e100.net (142.250.194.132): icmp_seq=2 ttl=128 time=91.0 ms
48 bytes from del12s05-in-f4.1e100.net (142.250.194.132): icmp_seq=3 ttl=128 time=102 ms

--- www.google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 91.095/122.291/172.827/36.062 ms
student@ubuntu:~$
```

3. Netstat

```
student@ubuntu:~$ netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp6       1      0 ip6-localhost:55958     ip6-localhost:ipp       CLOSE_WAIT
Active UNIX domain sockets (w/o servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  19     [ ]         DGRAM                    10461    /dev/log
unix  3      [ ]         STREAM     CONNECTED     14422    /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     14380
unix  3      [ ]         STREAM     CONNECTED     16003
unix  3      [ ]         STREAM     CONNECTED     15805    @/dbus-vfs-daemon/socket-mbI0qrfU
unix  3      [ ]         STREAM     CONNECTED     13866    /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     10914
unix  3      [ ]         STREAM     CONNECTED     14448
unix  3      [ ]         STREAM     CONNECTED     16084
unix  3      [ ]         STREAM     CONNECTED     14500    @/tmp/dbus-EQbggWQU89
unix  3      [ ]         STREAM     CONNECTED     15999    @/tmp/dbus-EQbggWQU89
unix  3      [ ]         STREAM     CONNECTED     15023    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     13865
unix  3      [ ]         STREAM     CONNECTED     10813    /var/run/dbus/system_bus_socket
unix  2      [ ]         STREAM     CONNECTED     16313    @/tmp/dbus-wfONIXQs
unix  3      [ ]         STREAM     CONNECTED     14577    @/tmp/dbus-EQbggWQU89
unix  3      [ ]         STREAM     CONNECTED     14230    /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     14072
unix  3      [ ]         STREAM     CONNECTED     15022
unix  3      [ ]         STREAM     CONNECTED     13936    @/tmp/dbus-EQbggWQU89
unix  3      [ ]         STREAM     CONNECTED     14993
unix  3      [ ]         STREAM     CONNECTED     14350    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     14199
unix  3      [ ]         STREAM     CONNECTED     15941
unix  3      [ ]         STREAM     CONNECTED     14445
unix  3      [ ]         STREAM     CONNECTED     14427
unix  3      [ ]         STREAM     CONNECTED     14360    @/tmp/dbus-EQbggWQU89
unix  3      [ ]         STREAM     CONNECTED     16263    @/tmp/.X11-unix/X0
unix  3      [ ]         STREAM     CONNECTED     15025
unix  3      [ ]         STREAM     CONNECTED     14100    @/tmp/dbus-EQbggWQU89
unix  2      [ ]         DGRAM                    11305
```

```
student@ubuntu:~$ netstat -at
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu:domain           *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
tcp6       1      0 ip6-localhost:55958     ip6-localhost:ipp       CLOSE_WAIT
student@ubuntu:~$ netstat -au
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 *:47308                 *:*
udp        0      0 *:ipp                   *:*
udp        0      0 *:mdns                  *:*
udp        0      0 *:40336                 *:*
udp        0      0 ubuntu:domain           *:*
udp        0      0 *:bootpc                *:*
udp6       0      0 [::]:mdns               [::]:*
udp6       0      0 [::]:11535              [::]:*
udp6       0      0 [::]:38806              [::]:*
student@ubuntu:~$ netstat -pt
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp6       1      0 ip6-localhost:55958     ip6-localhost:ipp       CLOSE_WAIT  -
student@ubuntu:~$
```

4. Arp

```
student@ubuntu:~$ arp
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.254           ether   00:50:56:eb:f5:22   C                     eth0
192.168.64.2             ether   00:50:56:f0:78:8f   C                     eth0
student@ubuntu:~$ arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.254           ether   00:50:56:eb:f5:22   C                     eth0
192.168.64.2             ether   00:50:56:f0:78:8f   C                     eth0
Entries: 2      Skipped: 0      Found: 2
student@ubuntu:~$ arp -n
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.254           ether   00:50:56:eb:f5:22   C                     eth0
192.168.64.2             ether   00:50:56:f0:78:8f   C                     eth0
student@ubuntu:~$ arp -H ether
Address                  HWtype  HWaddress           Flags Mask            Iface
192.168.64.254           ether   00:50:56:eb:f5:22   C                     eth0
192.168.64.2             ether   00:50:56:f0:78:8f   C                     eth0
student@ubuntu:~$
```

5. Nslookup

```
student@ubuntu:~$ nslookup www.google.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.192.4

student@ubuntu:~$ nslookup 209.132.183.181
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
181.183.132.209.in-addr.arpa    name = origin-www2.redhat.com.

Authoritative answers can be found from:

student@ubuntu:~$ nslookup -type=any ibm.com
;; Truncated, retrying in TCP mode.
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
ibm.com nameserver = asia3.akam.net.
ibm.com nameserver = usw2.akam.net.
ibm.com nameserver = usc2.akam.net.
ibm.com nameserver = usc3.akam.net.
ibm.com nameserver = ns1-99.akam.net.
ibm.com nameserver = eur2.akam.net.
ibm.com nameserver = ns1-206.akam.net.
ibm.com nameserver = eur5.akam.net.
ibm.com has AAAA address 2600:140f:d800:1ad::3831
ibm.com has AAAA address 2600:140f:d800:196::3831
ibm.com mail exchanger = 5 mx0b-001b2d01.pphosted.com.
ibm.com mail exchanger = 5 mx0a-001b2d01.pphosted.com.
Name:   ibm.com
Address: 104.85.150.168
ibm.com
        origin = asia3.akam.net
        mail addr = dnsadm.us.ibm.com
        serial = 1564135723
```

```
student@ubuntu:~$ nslookup -type=txt gmail.com
Server:         127.0.1.1
Address:        127.0.1.1#53

Non-authoritative answer:
gmail.com       text = "globalsign-smime-dv=CDYX+XFHUw2wml6/Gb8+59BsH31KzUr6c1l2BPvqKX8="
gmail.com       text = "v=spf1 redirect=_spf.google.com"

Authoritative answers can be found from:
```

6. Traceroute

```
student@ubuntu: ~
student@ubuntu:~$ traceroute www.ibm.com
traceroute to www.ibm.com (23.57.199.57), 30 hops max, 60 byte packets
 1  192.168.64.2 (192.168.64.2)  0.250 ms  0.240 ms  0.227 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  * * *
23  * * *
24  * * *
25  * * *
26  * * *
27  * * *
28  * * *
29  * * *
30  * * *
student@ubuntu:~$
```

```
student@ubuntu:~$ traceroute -F  www.ibm.com
traceroute to www.ibm.com (23.57.199.57), 30 hops max, 60 byte packets
 1  192.168.64.2 (192.168.64.2)  0.251 ms  0.306 ms  0.163 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
```

```
student@ubuntu:~$ traceroute -m 5 www.yahoo.com
traceroute to www.yahoo.com (202.165.107.50), 5 hops max, 60 byte packets
 1  192.168.64.2 (192.168.64.2)  0.245 ms  0.303 ms  0.172 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
student@ubuntu:~$ traceroute -n www.yahoo.com
traceroute to www.yahoo.com (202.165.107.49), 30 hops max, 60 byte packets
 1  192.168.64.2  0.182 ms  0.165 ms  0.083 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
```

NAME: PRERNA SUNIL JADHAV
SAP ID: 60004220127

CLASS: S.Y. BTech (COMPUTER ENGG.)
COURSE: COMPUTER NETWORKS (DJ19CEL405)

DATE OF PERFORMANCE:
DATE OF SUBMISSION:

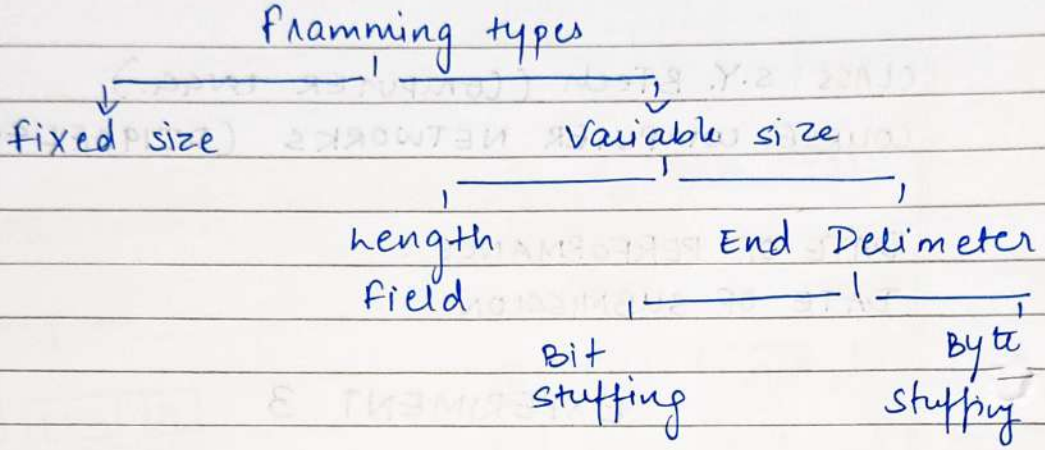
EXPERIMENT 3

AIM: To study the Framing Mechanism in Data Link layer

THEORY:

- Data link layer takes the packets from the Network layer and encapsulates them into frames
- If the frame size becomes too large, then the Packet may be divided into small sized frames. Smaller sized frames makes the flow control and error control more efficient
- Then, it sends each frame bit-by-bit At receiver's end, data link layer picks up signals from hardware & assembles them into frames.

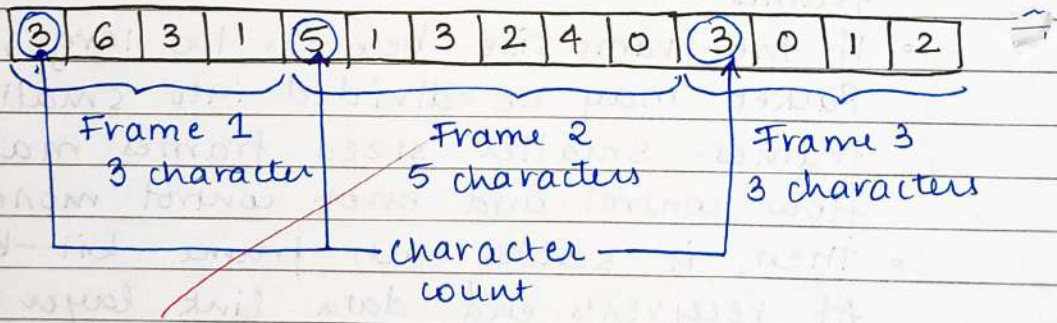
TYPES OF FRAMING:



METHODS OF FRAMING:

1. CHARACTER COUNT

- This method ensures data link layer at the receivers end about the total no. of characters that follow, and about cohere the frame ends.



- Disadvantage: If anyhow the character count is disturbed or distorted by an error occuring during transmission, they receiver might not be able to locate or identify the beginning of next frame.

2. BIT STUFFING

- This method is also known as bit-oriented framing in which extra bits are being added by network protocol designers to the data stream.
- Most protocols use a special 8-bit pattern flag 01111110 as a result of the delimiter to stipulate the beginning and so the end of the frame.

Data → 000111111100111110100

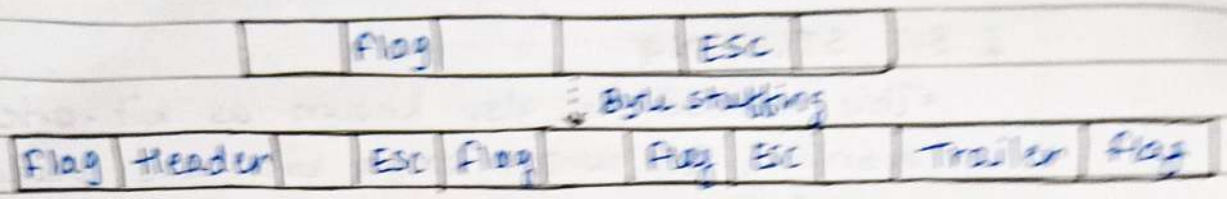
Stuffed ↓

Flag	Header	0001111111000111110001000	Trailer	Flag

- Disadvantage: This technique doesnt ensure that the sent data is intact at the receiver side. It is merely a way to ensure that transmission starts and ends at the correct places.

3. BYTE STUFFING

- In this method we add an additional byte once there is a flag or escape character within the text
- The sender sends the frame by adding three additional ESC bits and therefore the destination machine receives the frame and it removes the extra bits to convert the frame into an identical message



following are the examples of byte stuffing:

Original	After
A Flag B	A ESC FLAG B
A ESC B	A ESC ESC B
A ESC Flag B	A ESC ESC ESC Flag B
A ESC Flag B	A ESC ESC ESC ESC B

- Disadvantage: Size of frame varies unpredictably due to byte insertion

CONCLUSION:

- framing facilitates two of the primary functionalities
  i. It provides a mechanism for flow control that manages the frame flow such that the congestion of data is not there on slow receivers because of the fast sender
  ii. It provides reliable transfer services of bits between two layers of the peer entities.

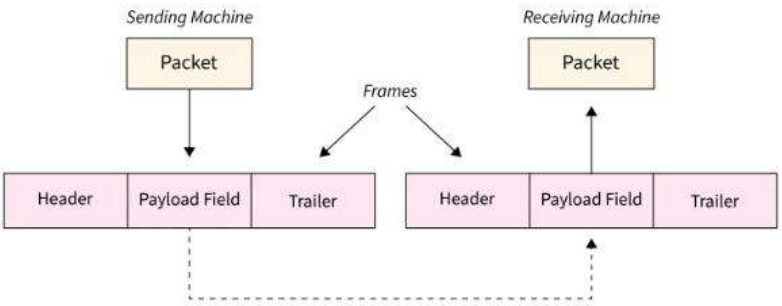
Asly

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	03
Aim:	Data Link Layer (Framing Mechanism)

AIM: DATA LINK LAYER – FRAMING MECHANISM

THEORY:

- In the physical layer, data transmission involves synchronised transmission of bits from the source to the destination.
- Data-link layer takes the packets from the Network Layer and encapsulates them into frames. If the frame size becomes too large, then the packet may be divided into small sized frames.

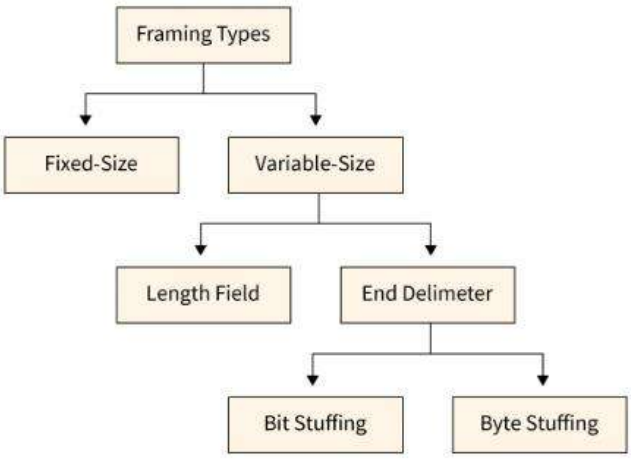


Smaller sized frames makes flow control and error control more efficient.

- Then, it sends each frame bit-by-bit on the hardware. At receiver's end, data link layer picks up signals from hardware and assembles them into frames.

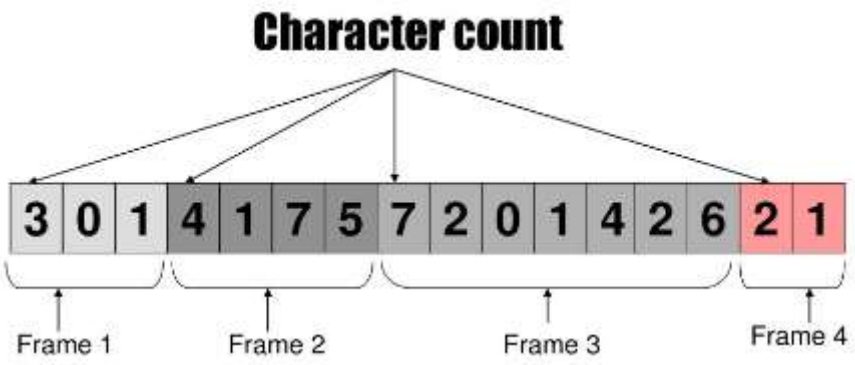
Types of Framing in Data Link Layer

- The frame can be of fastened or variable size. founded on the size, the following are the types of framing in data link layers in computer networks
- Fixed Size Framing is used in ATMs, Wide area networks(WAN)
- In variable-size framing, we need a way to outline the tip of the frame and also the starting of the succeeding frame. This can be utilized in local area networks(LAN).



Methods of Framing:

1. Character Count
   - Character count methodology makes sure that the Framing in the data link layer is at the receiver end regarding the total range of characters maintain, and wherever the frame ends.



   - It also has its disadvantage conjointly of utilizing this methodology which is, if in any case, the character count is issued or bended by a miscalculation occurring throughout the transmission process, then at the receiver end it may drop synchronization.
   - The receiver strength is ineffective to find or establish the start of the next frame.
   - Program:

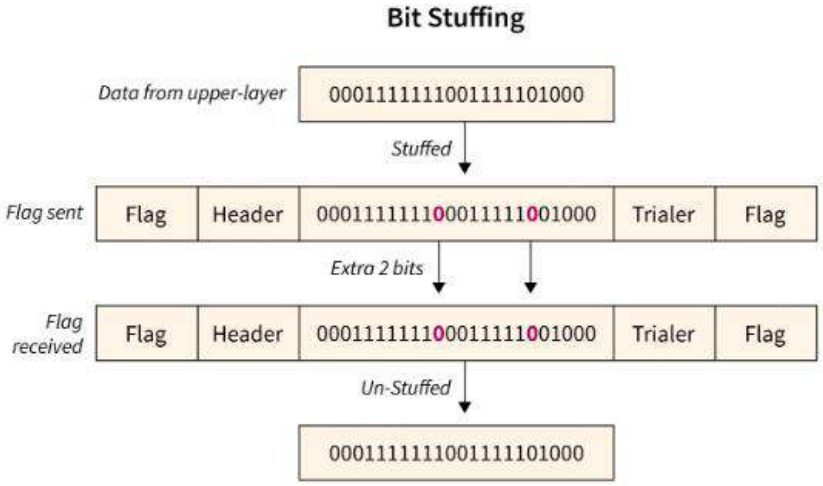
```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

char *sender(int n)
{
    static char ch[100];
    char data_unit[30];
    char length[1];

    for (int i = 0; i < n; i++)
    {
        printf("Enter the Data Unit %d: ", (i + 1));
        scanf("%s", data_unit);

        length[0] = (strlen(data_unit) + 1) + '0';
        strcat(ch, length);
    }
    return ch;
}
```

```
void receiver(char *data)
{
    size_t i = 0;
    int count = 0;
    printf("\nThe Data Units transferred were: \n");
    while (data[i] != '\0')
    {
        int isADigit = isdigit(data[i]);
        if (isADigit == 1)
        {
            count++;
            printf("Data Unit %d: ", count);
            int dataUnitLen = 0;
            dataUnitLen = data[i] - '0';
            for (int j = 0; j < dataUnitLen - 1; j++)
            {
                i++;
                printf("%c", data[i]);
            }
            i++;
            printf("\n");
        }
    }
}

int main()
{
    int n = 0;
    char *data_To_Be_Transmitted;

    printf("Enter the number of data units: ");
    scanf("%d", &n);

    if (n <= 0)
    {
        printf("No data units received");
        return 1;
    }

    data_To_Be_Transmitted = sender(n);
    printf("Data String to be Transmitted: %s", data_To_Be_Transmitted);
    receiver(data_To_Be_Transmitted);
    return 0;
}
```

- Output:

```
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>  & 'c:\Users\Jadhav\.vscod
s\ms-vscode.cpptools-1.14.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher
din=Microsoft-MIEngine-In-dtg4vblq.pjs' '--stdout=Microsoft-MIEngine-Out-dsm
'--stderr=Microsoft-MIEngine-Error-dwkzp4ny.lcx' '--pid=Microsoft-MIEngine-P
.nf0' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
Enter the number of data units: 4
Enter the Data Unit 1: This
Enter the Data Unit 2: is
Enter the Data Unit 3: Prerna
Enter the Data Unit 4: Jadhav
Data String to be Transmitted: 5This3is7Prerna7Jadhav
The Data Units transferred were:
Data Unit 1: This
Data Unit 2: is
Data Unit 3: Prerna
Data Unit 4: Jadhav
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

2. Bit-Oriented Framing
   - Most protocols use a special 8-bit pattern flag 01111110 as a result of the delimiter to stipulate the beginning and so the end of the frame.



   - Bit stuffing is completed at the sender end and bit removal at the receiver end.
   - Program:

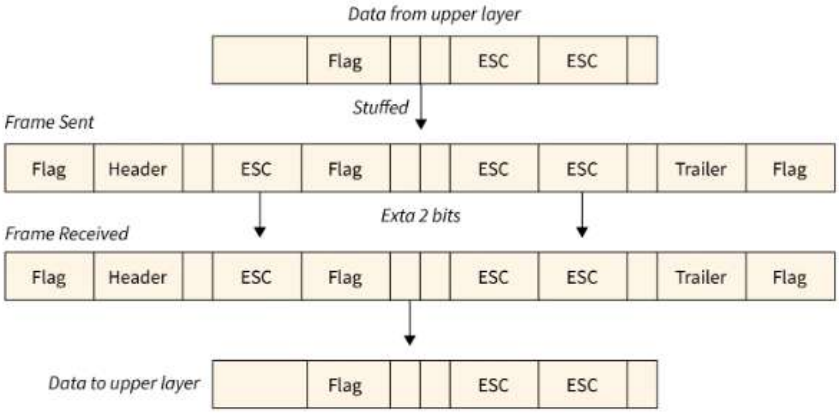
```c
#include <stdio.h>
#include <string.h>
int main()
{
    int a[20], b[30], i, j, k, count, n;
    printf("Enter frame size:");
    scanf("%d", &n);
    printf("Enter the frame in the form of 0 and 1 :");
```

```
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);

    i = 0;
    count = 1;
    j = 0;
    while (i < n)    {
        if (a[i] == 1)
        {
            b[j] = a[i];
            for (k = i + 1; a[k] == 1 && k < n && count < 5; k++)  {
                j++;
                b[j] = a[k];
                count++;
                if (count == 5) {
                    j++;
                    b[j] = 0;
                }
                i = k;
            }
        }
        else
        {
            b[j] = a[i];
        }
        i++;
        j++;
    }
    printf("After Bit Stuffing :");
    for (i = 0; i < j; i++)
        printf("%d", b[i]);

    return 0;
}
```

- Output:

```
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>  & 'c:\Users\Jadhav\.vscode
s\ms-vscode.cpptools-1.14.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.
din=Microsoft-MIEngine-In-ebxxnml1.db2' '--stdout=Microsoft-MIEngine-Out-w5se
'--stderr=Microsoft-MIEngine-Error-et5cuchp.cjc' '--pid=Microsoft-MIEngine-Pi
.kwf' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '--interpreter=mi'
Enter frame size:10
Enter the frame in the form of 0 and 1 : 0 1 1 1 1 1 1 0 1 0
After Bit Stuffing :01111101010
```

3. Byte-Oriented Framing
   - Byte stuffing is one of the methods of adding an additional byte once there is a flag or escape character within the text.



   - Take an illustration of byte stuffing as appeared in the given diagram.
   - The sender sends the frame by adding three additional ESC bits and therefore the destination machine receives the frame, and it removes the extra bits to convert the frame into an identical message.
   - Program:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char frame[50][50], str[50][50];
    char flag[10];
    strcpy(flag, "flag");
    char esc[10];
    strcpy(esc, "esc");
    int i, j, k = 0, n;
    strcpy(frame[k++], "flag");
    printf("Enter length of String : ");
    scanf("%d", &n);
    printf("Enter the String: \n");
    for (i = 0; i <= n; i++)
    {
        gets(str[i]);
    }
    printf("You entered :\n");
    for (i = 0; i <= n; i++)
    {
        puts(str[i]);
    }
```

```
    printf("\n");
    for (i = 1; i <= n; i++)
    {
        if (strcmp(str[i], flag) != 0 && strcmp(str[i], esc) != 0)
        {
            strcpy(frame[k++], str[i]);
        }
        else
        {
            strcpy(frame[k++], "esc");
            strcpy(frame[k++], str[i]);
        }
    }
    strcpy(frame[k++], "flag");
    printf("------------------------------\n");
    printf("Byte stuffing at sender side:\n");
    printf("------------------------------\n");
    for (i = 0; i < k; i++)
    {
        printf("%s\t", frame[i]);
    }
    return 0;
}
```

- Output:

```
Adapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-ilmaoa4f.qo4' '--stdout=Microsc
 '--stderr=Microsoft-MIEngine-Error-irnjnes4.4m1' '--pid=Microsoft-MIEngine-Pid-amwqinbj.myp' '--dbgE
exe' '--interpreter=mi'
Enter length of String : 6
Enter the String:
1
1
0
1
0
0
You entered :

1
1
0
1
0
0

------------------------------
Byte stuffing at sender side:
------------------------------
flag    1       1       0       1       0       0       flag
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

Conclusion:

- Framing in the Data link layer additionally contains headers that embody information like error-checking codes.
- Framing in data link layer relay, token ring, ethernet, and alternative sorts of data link layer ways have their frame structures.
- Framing in the Data link layer enables the information to be divided into multiple recoverable elements that may be inspected for corruption.
- Framing in the Data link layer provides a flow management mechanism that manages the frame flow such that the information congestion does not occur on slow receivers thanks to quick senders.
- Framing in the Data link layer provides valid information transfer services within the layers of the peer network.

NAME: PRERNA SUNIL JADHAV
SAP ID: 60004220127

CLASS: S.Y. B Tech (COMPUTER ENGG.)
COURSE: COMPUTER NETWORK (DJ19CEL405)

DATE OF PERFORMANCE:
DATE OF SUBMISSION:

EXPERIMENT 4

AIM: Write a program to implement Error Detection and correction mechanism
i) Hamming code ii) CRC

THEORY:

ERROR DETECTION

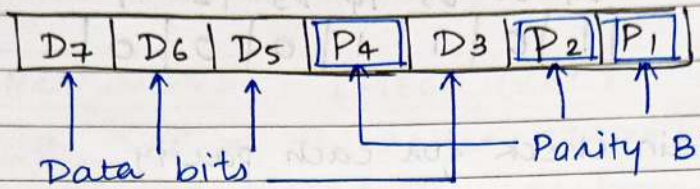
- Error is a condition when the receiver's information doesnt match with senders info.
- whenever a message is transmitted, it may get scrambled by noise or data may get corrupted.
- To avoid this, error detecting codes are added to help us detect if any error occured
- Basic approach used is the use of the redundancy bits, where additional bits are added to detect errors. Some popular ones are 1. CRC, 2. Checksum, 3. simple parity check etc.

ERROR CORRECTION

- Error correction codes are used to detect and correct the errors when data is transmitted
- For correcting the error, one has to know the exact position of the error. To achieve this, we have to add some additional bits
- Eg: Hamming code

HAMMING CODE

- This technique is capable of detecting and correcting errors that occur in the data
- In this technique, parity bits are used. The number of redundant bits can be calculated using: $2^r \geq m + r + 1$ where, r is the redundant bit, m = data bit
- Suppose the no. of data bits is 7 ($m = 7$), then $r = 4$ i.e., Parity bits = 4, since
  $2^4 \geq 7 + 4 + 1$
- There are 2 types of parity bits
  1. Even parity bits: In this case, if the count of 1's is odd the parity value is set to 1, making the total count of 1's even
  2. Odd parity bit: In this case, if the count of 1's is even, the parity value is set to 1, making odd no. of 1's.
- For Eg; if we have length of data = 7
  1) No. of databits = 7
  2) Redundant bits = 4
  3) Position of Redundant bits = 2^0, 2^1, 2^2, 2^3 = 1, 2, 4, 8



For Eg: Data given: 1110, even parity

1	1	1	P4	0	P2	P1

To find P_1:

P_1	D_3	D_5	D_7
0	0	1	1

← Because D_5, D_7 has 1 so we can put P_1 0

To find P_2:

P_2	D_3	D_6	D_7
0	0	1	1

← same as above reasoning

To find P_4:

P_4	D_5	D_6	D_7
1	1	1	1

← To make even no. of 1's
$P_4 \Rightarrow 1$

Now, our data to be transmitted is -

1	1	1	1	0	0	0

Suppose that D_6 is flipped, now the data to be transmitted will be -

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	1	1	0	0	0

Now again check for each parity,

For P_1,

P_1	D_3	D_5	D_7
0	0	1	1

→ All good, even no. of 1's

For P_2,

P_2	D_3	D_6	D_7
0	0	0	1

$P_2 = 1$ ← If any error put parity 1

→ Here there should be even 1's but there are odd 1's present

For P_4,

P_4	D_5	D_6	D_7
1	1	0	1

$P_4 = 1$ → same error as above

No change for P_1, but $P_2 \to 1$ and $P_4 \to 0$

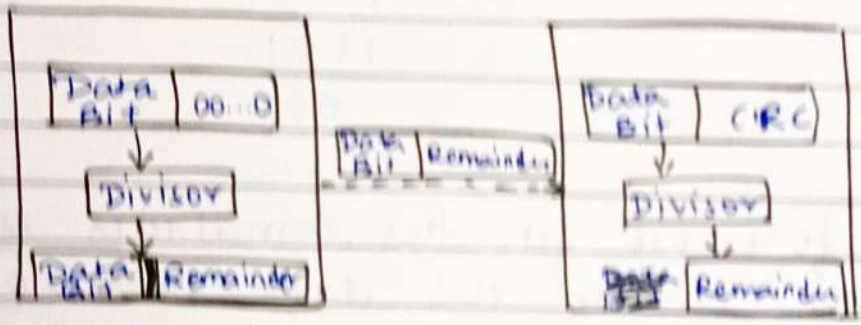
$\therefore (P_4, P_2, P_1) = (1, 1, 0)$ i.e., 6th bits has error and we need to correct it

$\therefore$ The corrected data is

D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	1	1	1	0	0	0

CYCLIC REDUNDANCY CHECK (CRC)

- It is a method to detect errors in the data.
- This is performed by performing a binary solution on the transmitted data at the sender's side and verifying the same at the receiver side.
- For eg:
  Given: $x^2 + x + 1$ (polynomial eqn)
  Convert it into binary format
  [1 1 1]
- Working



Sender side Receiver side

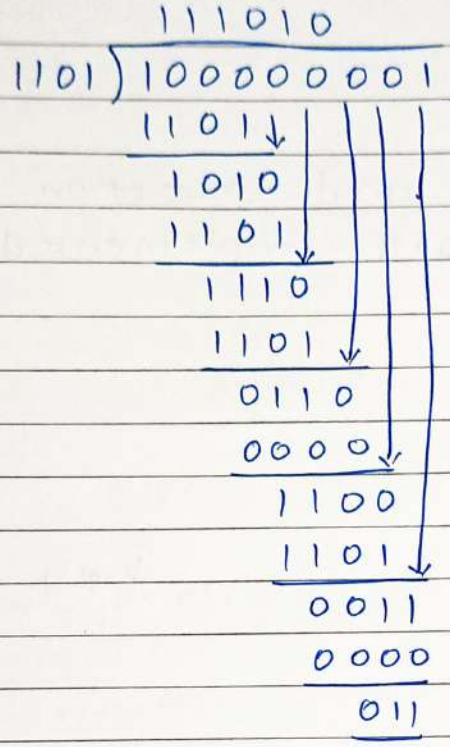
- If data to be sent = 100100 / key:- 1101 then we would append (n-1) zeros to the data where n is the length of key
  ⇒ 100100 000

- Now we would perform binary division operation of 100100000 using 1101 as divisor.

we get,

```
      111101
1101 ) 100100000
       1101
       ----
        1000
        1101
        ----
         1010
         1101
         ----
          1110
          1101
          ----
           0110
           0000
           ----
            1100
            1101
            ----
             001
```

- Therefore we get the remainder 001 and hence encode the data 100100001 and send to the receiver
- Assuming there was some error in the data transmission and the received data was 100000001.
- Now at receivers end we would again perform binary division. to check for error ~~and then correct it if any.~~



- Since the remainder is not all zero, the error is detected.

WHY CRC IS ADDED AT TRAILER OF THE FRAME?

- Having CRC in the trailer helps to compute the CRC while transmitting the packet and then splat the final CRC at the end
- If we put CRC in the header, it will handle each byte twice, once for checksum and once for transmitting
- Placing the CRC at the end of a frame reduces packet latency and reduces h/w buffering requirement.

CONCLUSION

Error detection and correction mechanism are studied and implemented.

Shilpa

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	04
Aim:	Error Detection and Correction Mechanism

AIM: Error Detection and Correction Mechanism

A. HAMMING CODE

CODE:

```
#include <stdio.h>
void main()
{
    int data[10];
    int dataatrec[10], c, c1, c2, c3, i;
    printf("Enter 4 bits of data one by one\n");
    scanf("%d", &data[0]);
    scanf("%d", &data[1]);
    scanf("%d", &data[2]);
    scanf("%d", &data[4]);
    // Calculation of even parity
    data[6] = data[0] ^ data[2] ^ data[4];
    data[5] = data[0] ^ data[1] ^ data[4];
    data[3] = data[0] ^ data[1] ^ data[2];
    printf("\nEncoded data is\n");
    for (i = 0; i < 7; i++)
        printf("%d", data[i]);
    printf("\n\nEnter received data bits one by one\n");
    for (i = 0; i < 7; i++)
        scanf("%d", &dataatrec[i]);
    c1 = dataatrec[6] ^ dataatrec[4] ^ dataatrec[2] ^ dataatrec[0];
    c2 = dataatrec[5] ^ dataatrec[4] ^ dataatrec[1] ^ dataatrec[0];
    c3 = dataatrec[3] ^ dataatrec[2] ^ dataatrec[1] ^ dataatrec[0];
    c = c3 * 4 + c2 * 2 + c1;

    if (c == 0)
    {
```

```
        printf("\nNo error while transmission of data\n");
    }
    else
    {
        printf("\nError on position %d", c);
        printf("\nData sent : ");
        for (i = 0; i < 7; i++)
            printf("%d", data[i]);
        printf("\nData received : ");
        for (i = 0; i < 7; i++)
            printf("%d", dataatrec[i]);
        printf("\nCorrect message is\n");
        if (dataatrec[7 - c] == 0)
            dataatrec[7 - c] = 1;
        else
            dataatrec[7 - c] = 0;
        for (i = 0; i < 7; i++)
        {
            printf("%d", dataatrec[i]);
        }
    }
}
```

OUTPUT:

```
cpptools-1.15.4-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-I
n-4ttzbfvz.1xx' '--stdout=Microsoft-MIEngine-Out-fozdixs4.njn' '--stderr=Microsoft-MIEngine-Error-0
b5sp0oy.3tf' '--pid=Microsoft-MIEngine-Pid-11fgscgb.wvo' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '
--interpreter=mi'
Enter 4 bits of data one by one
1
1
0
1

Encoded data is
1100110

Enter received data bits one by one
1
0
0
1
1
0
1

Error on position 1
Data sent : 1100110
Data received : 1001101
Correct message is
1001100
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

B. CRC

CODE:

```
#include <stdio.h>

#include <string.h>
#define N strlen(gen_poly)
char data[28];
char check_value[28];
char gen_poly[10];
int data_length, i, j;
void XOR()
{
    for (j = 1; j < N; j++)
        check_value[j] = ((check_value[j] == gen_poly[j]) ? '0' : '1');
}
void receiver()
{
    printf("Enter the received data: ");
    scanf("%s", data);
    printf("\n-----------------------------\n");
    printf("Data received: %s", data);
    crc();
    for (i = 0; (i < N - 1) && (check_value[i] != '1'); i++);
    if (i < N - 1)
        printf("\nError detected\n\n");
    else
        printf("\nNo error detected\n\n");
}

void crc()
{
    for (i = 0; i < N; i++)
        check_value[i] = data[i];
    do
    {
        if (check_value[0] == '1')
            XOR();
        for (j = 0; j < N - 1; j++)
            check_value[j] = check_value[j + 1];
        check_value[j] = data[i++];
    } while (i <= data_length + N - 1);
}

int main()
```

```
{
    printf("\nEnter data to be transmitted: ");
    scanf("%s", data);
    printf("\n Enter the Generating polynomial: ");
    scanf("%s", gen_poly);
    data_length = strlen(data);
    for (i = data_length; i < data_length + N - 1; i++)
        data[i] = '0';
    printf("\n----------------------------------------");
    printf("\n Data padded with n-1 zeros : %s", data);
    printf("\n----------------------------------------");
    crc();
    printf("\nCRC or Check value is : %s", check_value);
    for (i = data_length; i < data_length + N - 1; i++)
        data[i] = check_value[i - data_length];
    printf("\n----------------------------------------");
    printf("\n Final data to be sent : %s", data);
    printf("\n----------------------------------------\n");
    receiver();
    return 0;
}
```

OUTPUT:

```
Enter data to be transmitted: 1001101

 Enter the Generating polynomial: 1011

---------------------------------------
 Data padded with n-1 zeros : 1001101000
---------------------------------------
CRC or Check value is : 101
---------------------------------------
 Final data to be sent : 1001101101
---------------------------------------
Enter the received data: 1001100

-----------------------------
Data received: 1001100
Error detected

PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

NAME: PRERNA SUNIL JADHAV
SAP ID: 60004220127

CLASS: S.Y. B.Tech (COMPUTER ENGG.)
COURSE: COMPUTER NETWORKS (DJ19CEL405)

DATE OF PERFORMANCE: 11 APRIL 2023
DATE OF SUBMISSION: 18 APRIL 2023

EXPERIMENT No. 5

AIM: Implementing Routing Protocol => Distance vector routing (DVR)

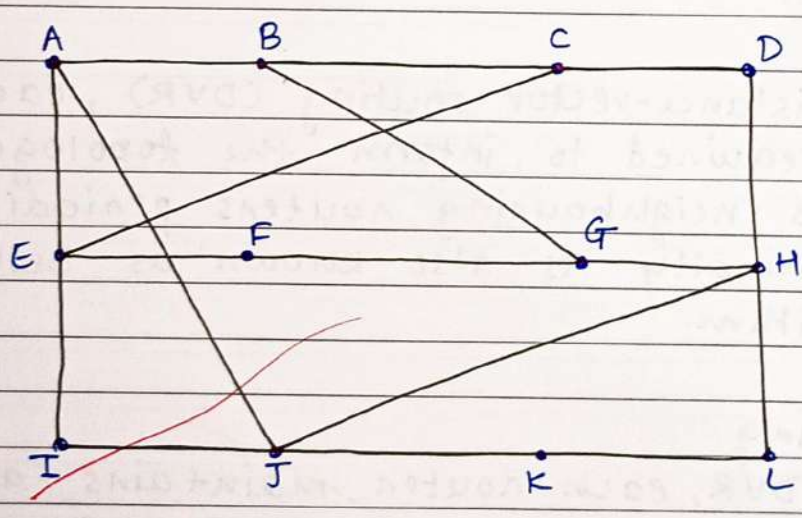
THEORY:

- In distance-vector routing (DVR), each router is required to inform the topology changes to its neighbouring routers periodically.
- Historically its also known as Bellman-Ford algorithm.

- Working
  - In DVR, each router maintains a routing table. It contains only one entry for each router. It contains two parts - a preferred outgoing line to use for that destination and an estimate of time (delay). Tables are updated by exchanging the information with the neighbour's nodes.

- Each router knows the delay in reaching its neighbours (Ex- send echo request)
- Routers periodically exchange routing tables with each of their neighbours.
- It compares the delay in the ~~neigh~~ local table with the delay in the neighbour's table and the cost of reaching that neighbour
- If the path via the neighbour has a lower cost, then the router updates its local table to forward packets to the neighbour.

→ For Eg:



A subnet

To	A	I	H	K	New estimated delay from J ↓	Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	—
K	24	22	22	0	6	K
L	29	33	9	9	15	K

JA delay is	JI delay is	JH delay is	JK delay is
8	10	12	6

Vectors received from J's four neighbours

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	05
Aim:	Distance Vector Routing using Bellman Ford

AIM: DISTANCE VECTOR ROUTING USING BELLMAN FORD

CODE:

```
#include <stdio.h>
struct node{
    unsigned dist[20];
    unsigned from[20];
} rt[10];
int main(){
    int dmat[20][20];
    int n, i, j, k, count = 0;
    printf("\nEnter the number of nodes : ");
    scanf("%d", &n);
    printf("\nEnter the cost matrix :\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)        {
            scanf("%d", &dmat[i][j]);
            dmat[i][i] = 0;
            rt[i].dist[j] = dmat[i][j];
            rt[i].from[j] = j;
        }
    do
    {
        count = 0;
        for (i = 0; i < n; i++)
            for (j = 0; j < n; j++)
                for (k = 0; k < n; k++)
                    if (rt[i].dist[j] > dmat[i][k] +
rt[k].dist[j])                    {
                        rt[i].dist[j] = rt[i].dist[k] + rt[k].dist[j];
                        rt[i].from[j] = k;
                        count++;
```

```
                    }
    } while (count != 0);
    for (i = 0; i < n; i++)    {
        printf("\n\nState value for router %d is \n", i + 1);
        for (j = 0; j < n; j++)        {
            printf("\t\nnode %d via %d Distance%d", j + 1, rt[i].from[j] + 1,
rt[i].dist[j]);
        }
    }
    printf("\n\n");
}
```

OUTPUT:

```
n-i4cwrjjx.cyd' '--stdout=Microsoft-MIEngine-Out-rjtptuzu.44c' '--stderr=Microsoft-MIEngine-Error-3
yjpnon1.bcw' '--pid=Microsoft-MIEngine-Pid-mmt3fsj2.g1f' '--dbgExe=C:\msys64\mingw64\bin\gdb.exe' '
--interpreter=mi'

Enter the number of nodes : 4

Enter the cost matrix :
0 3 5 99
3 0 99 1
5 4 0 2
99 1 2 0


State value for router 1 is

node 1 via 1 Distance0
node 2 via 2 Distance3
node 3 via 3 Distance5
node 4 via 2 Distance4

State value for router 2 is

node 1 via 1 Distance3
node 2 via 2 Distance0
node 3 via 4 Distance3
node 4 via 4 Distance1

State value for router 3 is

node 1 via 1 Distance5
node 2 via 4 Distance3
node 3 via 3 Distance0
node 4 via 4 Distance2

State value for router 4 is

node 1 via 2 Distance4
node 2 via 2 Distance1
node 3 via 3 Distance2
node 4 via 4 Distance0

PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

Academic Year: 2022-2023

Name:	Prerna Sunil Jadhav
Sap Id:	60004200137
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	06
Aim:	Socket TCP and UDP

AIM: TO IMPLEMENT TCP AND UDP SOCKET COMMUNICATION IN JAVA.

THEORY:

- The transport layer is the fourth layer of the OSI model which is responsible for the process-to-process delivery of data.
- The main aim of the transport layer is to maintain the order so that the data must be received in the same sequence as it was sent by the sender.
- The transport layer provides two types of services namely - connection-oriented and connection-less.
- The various protocols used in this layer are:
  - TCP (Transmission Control Protocol),
  - UDP (User Datagram Protocol), etc.

TCP - Transmission Control Protocol

- It is a **connection-oriented** protocol that defined the standard of establishing and maintaining the conversation (or connection) that will be used by the applications to exchange the data.
- In the connection-oriented protocol, we **first need to connect to the receiver before sending** our data.
- One of the prime reasons for using the transmission control protocol over the other protocol(s) like UDP is that the TCP ensures the **reliable transmission** and delivery of our data packets.
- The transmission control protocol **can deal with the various issues** that can occur in the data transmission such as packet duplication, packet corruption, packet disordering, packet loss, etc.
- **Working of TCP**
  - The entire process of establishing a connection, sending data packets, and then removing the connection comes under the working of the transmission control protocol.
    - Step 1: Establish a three-way handshake connection. The three-way handshake connection is used to create a connection between the host or client and the server. As the name suggests, it is a three-step process in which –
      - first the client (wants to establish a connection) sends an SYN segment (Synchronize Sequence Number segment) which tells the server that the client wants to start the communication.

- Then server responds with an SYN-ACK signal (SYN Acknowledgement). The SYN-ACK signifies the server has received the client's request of establishing the connection.
- Lastly the client again sends the ACK signal to the server and they both establish a reliable connection that will be used to transfer the data packets.
- The three-way handshake is also known as **SYN-SYN-ACK**. Refer to the diagram below for more clarity.

- Step 2: Sending of data packets along with the sequence number from the client. The second computer (server) responds to these sent packets by sending an acknowledgment or ACK. This acknowledgment bit keeps on increasing with the number of packets sent. This ACK bit helps to keep track of three things:
  - The successfully received packets,
  - The lost packets, and
  - The packets which were accidentally sent twice.
- Step 3: ither server or the client can close the connection. The first computer system (either server or the client) initiates the closing of the connection by sending a packet with a FIN bit or finish bit attached to it. The other computer sends back or responds with an ACK bit. Finally, the first computer sends an ACK bit back to the second computer and the connection gets closed.

✓ **Features of TCP**

- **Connection-oriented**
- **Reliable data transfer**.
- **Flow Control** limits the rate of data transfer from the sender's end. The receiving end (receiver) continuously hints the rate at which it can receive the data and the TCP uses the hints to adjust the data transfer rate of the sender.
- **Error Control** - error-checking and error recovery.
- **Sequencing of packets**
- **Congestion Control** - Congestion means the amount of data transferred in the network. So, the transmission control protocol also accounts for the level of congestion and sends the packets paccordingly.
- **Full duplex**

UDP – User Datagram Protocol

- ✓ UDP protocol is the **connectionless and unreliable** protocol.
- ✓ Since UDP is a connectionless protocol so there is **no need to establish a connection** before transmitting data.
- ✓ UDP packets are called **User Datagram**.
- ✓ User Datagram has **8 bytes fixed-size header.**
- ✓ UDP protocol will work just like an alternative of TCP (Transmission Control Protocol).
- ✓ **Working of UDP protocol**
  - UDP works by collecting data in a UDP packet, and in the packet, it adds its own header information.
  - UDP packet, called user datagram, consists of:
    - **Source port number** is 2 bytes field that defines the port number of the sender.
    - **Destination port number** is 2 bytes field that defines the port number of the destination.
    - **Packet length** is also 2 bytes field for defining the total length of the user datagram (header length + data length)
    - **Checksum** is a 2 bytes optional field for carrying checksum
  - UDP packets are sent to their destination after encapsulating it in an IP packet
  - In UDP, there is no acknowledgment generated for the packet received, so the sender does not wait for acknowledgment of the sent packet.
- ✓ **Features of UDP protocol**
  - Connectionless
  - Since it is connectionless, so packets are sent from different paths between sender and receiver.
  - Ordered delivery of data is not guaranteed.
  - The UDP protocol utilizes different port numbers for transmitting data to the correct destination.
  - The port numbers are defined between 0 - 1023.
  - Faster transmission
  - UDP provides us a faster service of data transmission as there is no prior connection establishment before transmitting the data.
  - UDP does not require any virtual path for data transmission.
  - No acknowledgment mechanism so there is no handshaking.
  - Segments are handled independently.
  - Every segment in UDP takes a different path to reach the destination. So, every UDP packet is handled independent of other UDP packets.
  - Stateless - sender does not wait for an acknowledgment after sending the packet.
- ✓ Applications of User Datagram Protocol (UDP)
  - UDP protocol can be utilized for simple request-response communication when there is a smaller size data since there is very less concern about the error and flow control.
  - UDP carries packet switching, so UDP is considered suitable protocol for multicasting.
  - UDP is also used in s

DIFFERENCE BETWEEN TCP AND UDP

CRITERIA	TCP	UDP
Connection Type	connection-oriented.	connectionless.
Reliability	TCP is reliable as there is an absolute certainty that the data sent will arrive intact and in the same order as it was sent.	In UDP, data delivery to the destination cannot be assured.
Speed	Speed of TCP is slower than UDP	Error recovery isn't attempted with UDP, therefore it's quicker.
Header Size	Header length is variable: 20-60 bytes	Header length is fixed: 8 bytes
Acknowledgement	Acknowledgement segments	No Acknowledgement
Data Transmission	TCP transmits data in a certain sequence, which assures that packets arrive at the recipient in the proper order.	On the other side, there is no data sequencing in UDP to provide a sequence.
Retransmission	In TCP, data packets can be retransmitted if they are lost or need to be resent.	Retransmission of packets, on the contrary, is not feasible with UDP.
Error Checking	TCP performs error checking and recovery. Packets that are incorrectly sent from the source are resent to the destination.	Error checking is performed by UDP; however, erroneous packets are simply discarded. There is no attempt at error recovery.
Applications	TCP is best suited for applications requiring great reliability, and whose transmission time is less crucial.	UDP is well suited to applications that require quick and efficient transmission, such as video games. The unidirectional characteristic of UDP is especially advantageous for servers that respond to short queries from a large number of clients.
Handshake	SYN, SYN-ACK, ACK	No handshake: connectionless protocol

CONCLUSION:

- ✓ UDP is a much quicker, simpler, and more efficient protocol, nonetheless, only TCP allows for the retransmission of lost data packets.
- ✓ When comparing the TCP and UDP protocols, TCP is heavier while UDP is lighter.

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	06
Aim:	Socket TCP and UDP

AIM: **TO IMPLEMENT TCP AND UDP SOCKET COMMUNICATION IN JAVA.**

TCP:

CODE:

Server.java

```
// A Java program for a Server
import java.net.*;
import java.io.*;

public class Server {
    // initialize socket and input stream
    private Socket socket = null;
    private ServerSocket server = null;
    private DataInputStream in = null;

    private DataInputStream input = null;
    private DataOutputStream out = null;

    // constructor with port
    public Server(int port) {
        // starts server and waits for a connection
        try {
            server = new ServerSocket(port);
            System.out.println("Server started");

            System.out.println("Waiting for a client ...");

            socket = server.accept();
            System.out.println("Client accepted");
```

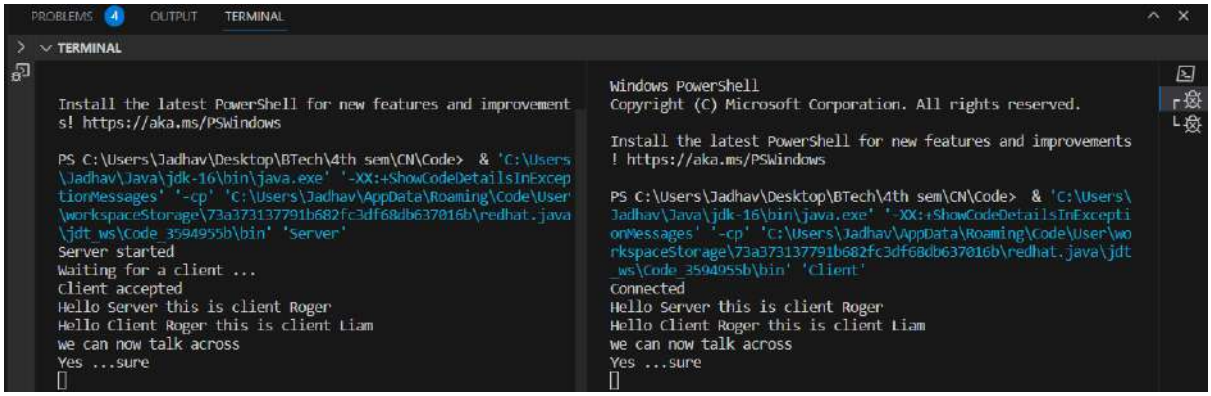
```
            // takes input from the client socket
            in = new DataInputStream(
                    new BufferedInputStream(socket.getInputStream()));

            out = new DataOutputStream(
                    socket.getOutputStream());

            input = new DataInputStream(System.in);

            String line = "";

            // reads message from client until "Over" is sent
            while (!line.equals("Over")) {
                try {
                    line = in.readUTF();
                    System.out.println(line);

                    line = input.readLine();
                    out.writeUTF(line);

                } catch (IOException i) {
                    System.out.println(i);
                }
            }
            System.out.println("Closing connection");

            // close connection
            socket.close();
            in.close();
        } catch (IOException i) {
            System.out.println(i);
        }
    }

    public static void main(String args[]) {
        Server server = new Server(5000);
    }
}
```

Client.java

```
// A Java program for a Client
import java.io.*;
import java.net.*;

public class Client {
    // initialize socket and input output streams
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;

    private DataInputStream in = null;

    // constructor to put ip address and port
    public Client(String address, int port) {
        // establish a connection
        try {
            socket = new Socket(address, port);
            System.out.println("Connected");

            // takes input from terminal
            input = new DataInputStream(System.in);

            // sends output to the socket
            out = new DataOutputStream(
                    socket.getOutputStream());

            in = new DataInputStream(
                    new BufferedInputStream(socket.getInputStream()));
        } catch (UnknownHostException u) {
            System.out.println(u);
            return;
        } catch (IOException i) {
            System.out.println(i);
            return;
        }

        // string to read message from input
        String line = "";

        // keep reading until "Over" is input
        while (!line.equals("Over")) {
            try {
                line = input.readLine();
```

```
                out.writeUTF(line);

                line = in.readUTF();
                System.out.println(line);
            } catch (IOException i) {
                System.out.println(i);
            }
        }

        // close the connection
        try {
            input.close();
            out.close();
            socket.close();
        } catch (IOException i) {
            System.out.println(i);
        }
    }

    public static void main(String args[]) {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

OUTPUT:



UDP:

CODE:

Server.java:

```
import java.io.*;
import java.net.*;

class Server {
    public static void main(String args[]) throws Exception {
        DatagramSocket serverSocket = new DatagramSocket(9876);
        BufferedReader inFromServer = new BufferedReader(new InputStreamReader(System.in));
        byte[] receiveData = new byte[1024];
        byte[] sendData = new byte[1024];
        while (true) {
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            serverSocket.receive(receivePacket);
            String sentence = new String(receivePacket.getData());
            System.out.println("RECEIVED: " + sentence);
            InetAddress IPAddress = receivePacket.getAddress();
            int port = receivePacket.getPort();
            String sentSentence = inFromServer.readLine();
            // String capitalizedSentence = sentence.toUpperCase();
            sendData = sentSentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
port);
            serverSocket.send(sendPacket);
            receiveData = new byte[1024];
            if (sentSentence.equalsIgnoreCase("bye")) {
                serverSocket.close();
                break;
            }
        }
    }
}
```

Client.java:

```
import java.io.*;
import java.net.*;

class Client {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        while (true) {
            String sentence = inFromUser.readLine();
            sendData = sentence.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress,
9876);
            clientSocket.send(sendPacket);
```

```
            DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
            clientSocket.receive(receivePacket);
            String modifiedSentence = new String(receivePacket.getData());
            System.out.println("FROM SERVER:" + modifiedSentence);
            receiveData = new byte[1024];
            if (sentence.equalsIgnoreCase("bye")) {
                clientSocket.close();
                break;
            }
        }
    }
}
```

OUTPUT:

```
Java\jdk-16\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Jadhav\AppData\Roaming\Code\User\workspaceStorage\73a373137791b682fc3df68db637016b\redhat.java\jdt_ws\Code_3594955b\bin' 'Server'
RECEIVED: Hello This is Client George
This is Server Roger
RECEIVED: How are we connected
We used UDP to connect
RECEIVED: bye
bye
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

```
s\Code_3594955b\bin' 'Client'
Hello This is Client George
FROM SERVER:This is Server Roger
How are we connected
FROM SERVER:We used UDP to connect
bye
FROM SERVER:bye
PS C:\Users\Jadhav\Desktop\BTech\4th sem\CN\Code>
```

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	07
Aim:	Determine Class of the IP address

AIM: TO DETERMINE THE CLASS OF THE IP ADDRESS

THEORY:

- ✓ An Internet Protocol (IP) address is a unique numerical identifier for every device or network that connects to the internet.
- ✓ Typically assigned by an internet service provider (ISP), an IP address is an online device address used for communicating across the internet.
- ✓ There are two versions of IP addresses that are commonly used on the internet: **IPv4 and IPv6**.
- ✓ An IPv4 address is expressed as a set of four dotted decimal numbers, where each octet is separated by a period, such as 192.168.35.4.
- ✓ The three digits in **the first octet represent a particular network** on the internet while the rest of the digits represent the actual host address within the local network, such as a workstation or a server.
- ✓ An IPv6 address represents eight groups of four hexadecimal digits separated by colons, such as 2620:cc:8000:1c82:544c:cc2e:f2fa:5a9b.
- ✓ Each internet protocol address can send information to other IP addresses through discrete chunks known as packets
- ✓ Each network packet contains the data being transferred along with a header containing the metadata of the packet.
- ✓ The following is a list of differences between IPv4 and IPv6:
  - IPv4 is 32-bit, whereas IPv6 is 128-bit.
  - In IPv4, binary bits are separated by a dot (.) IPv6 separates binary bits by a colon (:).
  - IPv4 follows the numeric addressing method and IPv6 is alphanumeric.
- ✓ Classful addressing is a network addressing the Internet's architecture from 1981 till Classless Inter-Domain Routing was introduced in 1993.
- ✓ This addressing method divides the IP address into five separate classes based on four address bits.
- ✓ Here, classes A, B, C offers addresses for networks of three distinct network sizes. Class D is only used for multicast, and class E reserved exclusively for experimental purposes.

- **Class A Network**

This IP address class is used when there **are a large number of hosts.** In a Class A type of network, the first 8 bits (also called the first octet) identify the network, and the remaining have 24 bits for the host into that network.
An example of a Class A address is 102.168.212.226. Here, "102" helps you identify the network and 168.212.226 identify the host.
Class A addresses 127.0.0.0 to 127.255.255.255 cannot be used and is reserved for loopback and diagnostic functions.

- **Class B Network**

In a B class IP address, the binary addresses start with 10. In this IP address, the class decimal number that can be between 128 to 191. The number 127 is reserved for loopback, **which is used for internal testing on the local machine**. The first 16 bits (known as two octets) help you identify the network. The other remaining 16 bits indicate the host within the network.
An example of Class B IP address is 168.212.226.204, where *168 212* identifies the network and *226.204* helps you identify the Hut network host.

- **Class C Network**

Class C is a type of IP address that is used for the small network. In this class, three octets are used to indent the network. This IP ranges between 192 to 223.
In this type of network addressing method, the first two bits are set to be 1, and the third bit is set to 0, which makes the first 24 bits of the address them and the remaining bit as the host address.
Mostly **local area network used Class C** IP address to connect with the network.
Example for a Class C IP address
192.168.178.1

- **Class D Network**

Class D addresses are only used **for multicasting applications**. Class D is never used for regular networking operations. This class addresses the first three bits set to "1" and their fourth bit set to use for "0". Class D addresses are 32-bit network addresses. All the values within the range are used to identify multicast groups uniquely.
Therefore, there is no requirement to extract the host address from the IP address, so Class D does not have any subnet mask.
Example for a Class D IP address:
227.21.6.173

- **Class E Network**

Class E IP address is defined by including the starting four network address bits as 1, which allows you two to incorporate addresses from 240.0.0.0 to 255.255.255.255. However, E class is reserved, and its usage is never defined. Therefore, many network implementations discard these addresses **as undefined** or illegal.
Example for a Class E IP address:
243.164.89.28

- **Limitations of classful IP addressing**
  - ✓ Risk of running out of address space soon
  - ✓ Class boundaries did not encourage efficient allocation of address space

R 415

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	07
Aim:	Determine class of IP Address

AIM: DETERMINE CLASS OF IP ADDRESS.

CODE:

```
#include<stdio.h>
#include<string.h>

char findClass(char str[])
{
    char arr[5];
    int i = 0;
    while (str[i] != '.')
    {
        arr[i] = str[i];
        i++;
    }
    i--;
    int ip = 0, j = 1;
    while (i >= 0)
    {
        ip = ip + (str[i] - '0') * j;
        j = j * 10;
        i--;
    }

    if (ip >=1 && ip <= 127)
        return 'A';

    else if (ip >= 128 && ip <= 191)
        return 'B';

    else if (ip >= 192 && ip <= 223)
        return 'C';

    else if (ip >= 224 && ip <= 239)
        return 'D';

    else if (ip >= 240 && ip <= 254)
        return 'E';
```

```
    else
        printf("Invalid IP");

    return 'Z';
}

int main()
{
    char str[] = {};
    printf("Enter an IP Address: ");
    scanf("%[^\n]s",str);
    char ipClass = findClass(str);
    printf("IP Address: %s\n", str);
    if (ipClass != 'Z'){
        printf("Given IP address belongs to Class %c\n", ipClass);
    }

    return 0;
}
```

OUTPUT:

```
p20jnl5.32x' '--pid=Microsoft-MIEngine-Pid-4gdunfeq.ecn' '--dbgExe=C:\msys64\mingw64\bin\gdb.ex
--interpreter=mi'
Enter an IP Address: 120.90.11.0
IP Address: A20.90.11.0
Given IP address belongs to Class A
```

Academic Year: 2022-2023

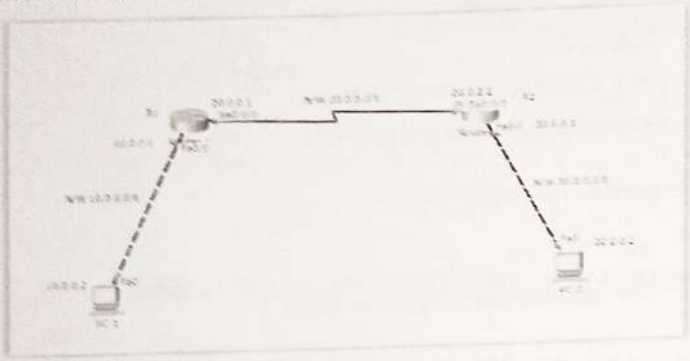
Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	08
Aim:	RIP configuration using packet tracer

AIM: RIP CONFIGURATION USING PACKET TRACER

THEORY:

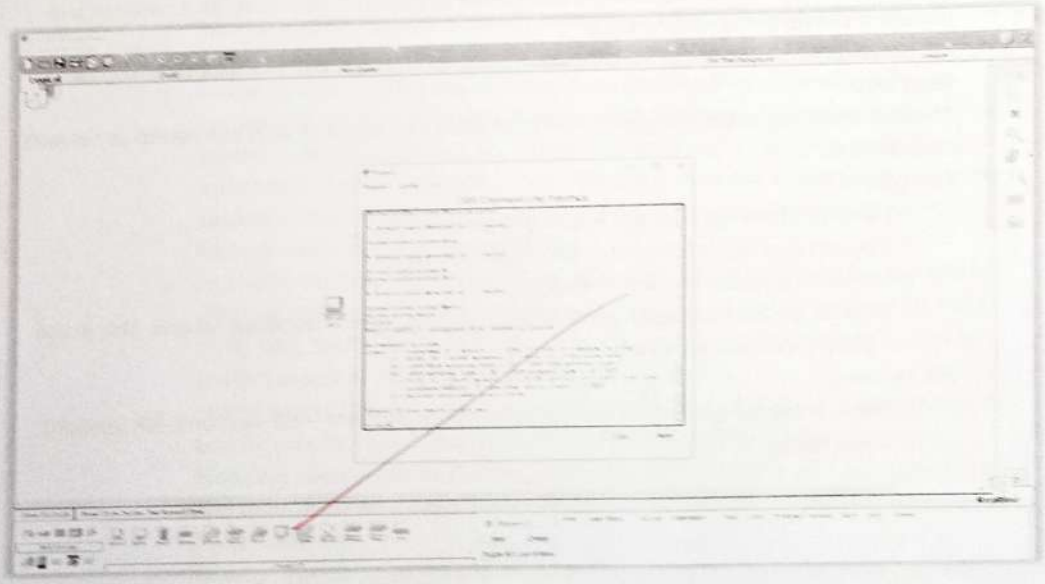
- ✓ Routing Information Protocol (RIP) is a dynamic routing protocol that uses hop count as a routing metric to find the best path between the source and the destination network.
- ✓ It is a distance-vector routing protocol that has an AD value of 120 and works on the Network
- ✓ layer of the OSI model.
- ✓ RIP uses port number 520.
- ✓ Hop count 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
  - o Updates of the network are exchanged periodically.
  - o Updates (routing information) are always broadcast.
  - o Full routing tables are sent in updates.
  - o Routers always trust routing information received from neighbour routers. This is also known as Routing on rumors.
- ✓ RIP versions:
  - o There are three versions of routing information protocol – RIP Version1, RIP Version2, and RIPng.

The Network:



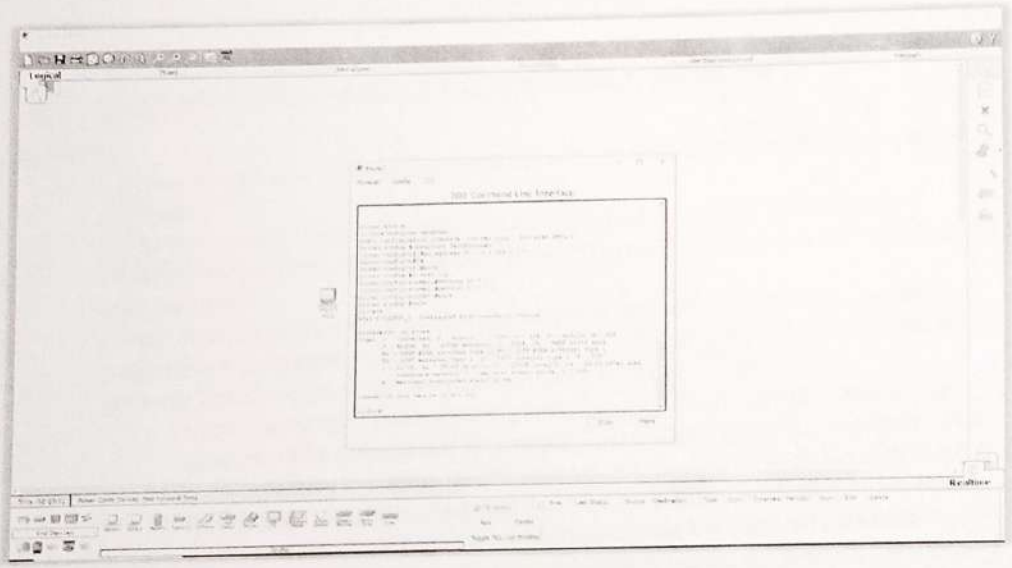
Configuring Router 1:

```
R1(config)#int fa0/0
R1(config-if)#ip address 10.0.0.1 255.0.0.0
R1(config-if)#int serial 0/0/0
R1(config-if)#ip add 20.0.0.1 255.0.0.0
R1(config-if)#no shut
```



Configuring Router 2:

```
R2(config)#
R2(config)#int fa0/0
R2(config-if)#ip add 30.0.0.1 255.0.0.0
R2(config-if)#no shut

R2(config-if)#
R2(config-if)#int serial 0/0/0
R2(config-if)#ip add 20.0.0.2 255.0.0.0
R2(config-if)#no shut
```



Academic Year: 2022-2023

Name:	Prerna Sunil Jadhav
Sap Id:	60004220127
Class:	S. Y. B.Tech (Computer Engineering)
Course:	Computer Networks (DJ12CEL405)
Date of Performance:	
Date of Submission:	
Experiment No.:	09
Aim:	Packet Capturing In Wireshark

AIM: TO IMPLEMENT AND SIMULATE PACKET CAPTURING IN WIRESHARK.

THEORY:

- ✓ A packet is a unit of data which is transmitted over a network between the origin and the destination.
- ✓ Network packets are small, i.e., maximum 1.5 Kilobytes for Ethernet packets and 64 Kilobytes for IP packets. The data packets in the Wireshark can be viewed online and can be analyzed offline.
- ✓ Packet capturing and analyzing are essential tasks in computer networking, especially in troubleshooting network problems and securing the network from malicious attacks. Here are some notes on packet capturing and analyzing in computer networking:
    - Packet capturing: Packet capturing is the process of intercepting and collecting network traffic data packets for analysis. Packet capturing tools are used to capture packets passing through a specific network interface. Some popular packet capturing tools include Wireshark, tcpdump, and Microsoft Network Monitor.
    - Packet analysis: Packet analysis involves examining the captured packets to understand the network traffic patterns, identify issues, and pinpoint the source of problems. Packet analysis tools are used to analyze the captured packets to gain insight into the network traffic, identify performance issues, and detect potential security threats. Some popular packet analysis tools include Wireshark, Microsoft Message Analyzer, and Network Miner.
    - Protocols: Network traffic consists of packets of data that are exchanged between different devices using different protocols. Popular protocols include TCP, UDP, HTTP, DNS, SMTP, FTP, and ICMP. Understanding the protocols used in network traffic is essential to perform effective packet analysis.
    - Filters: Packet filtering is the process of selecting and analyzing packets based on specific criteria. Packet filtering tools allow users to filter packets based on protocols, source and destination addresses, port numbers, and other parameters. This helps in isolating specific packets for analysis and improving the efficiency of packet analysis.
    - Security: Packet capturing and analysing can also be used for security purposes, such as detecting and preventing cyber-attacks. Network administrators can use packet

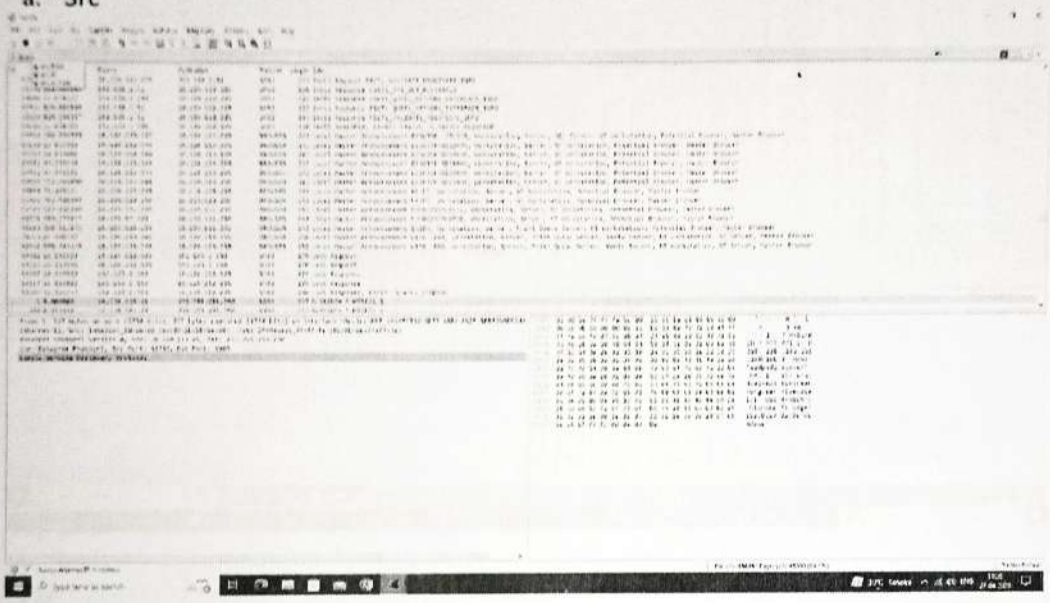
analysis to identify and block malicious traffic, analyse security threats, and detect intrusions.

- ✓ Wireshark is an open-source packet analyzer, which is used for education, analysis, software development, communication protocol development, and network troubleshooting.
- ✓ It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a sniffer, network protocol analyzer, and network analyzer. It is also used by network security engineers to examine security problems.
- ✓ Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.
- ✓ Uses of Wireshark:
  1. It is used by network security engineers to examine security problems.
  2. It allows the users to watch all the traffic being passed over the network.
  3. It is used by network engineers to troubleshoot network issues.
  4. It also helps to troubleshoot latency issues and malicious activities on your network.
  5. It can also analyze dropped packets.
  6. It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

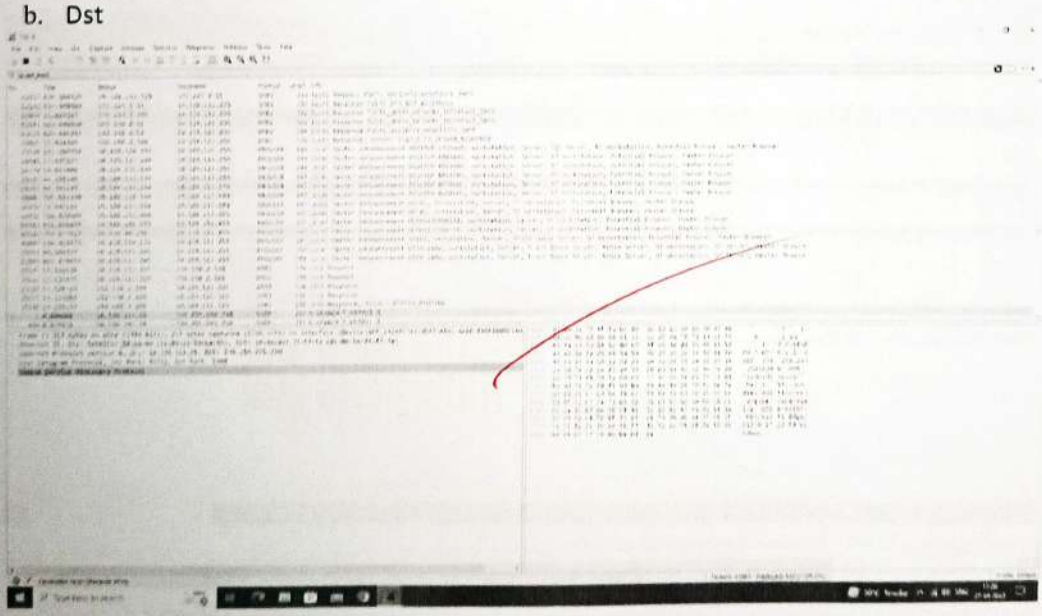
R 415

Filters in Wireshark

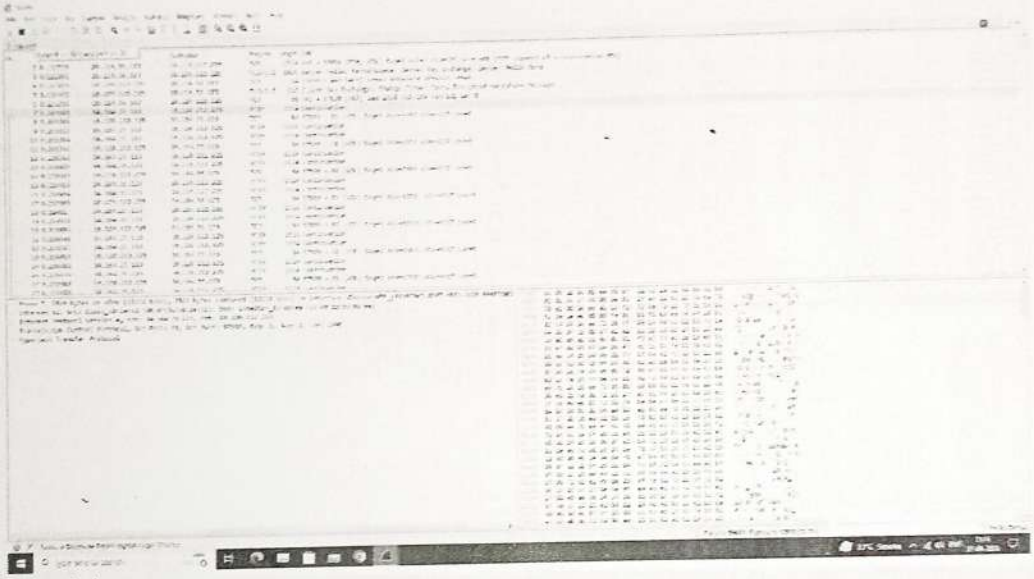
1. Ip:
   a. Src



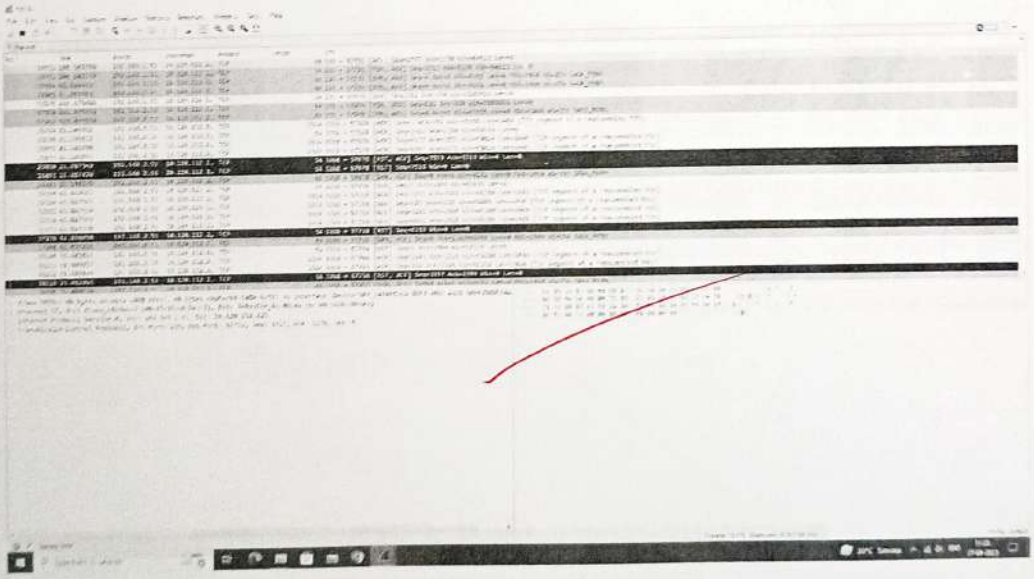
   b. Dst



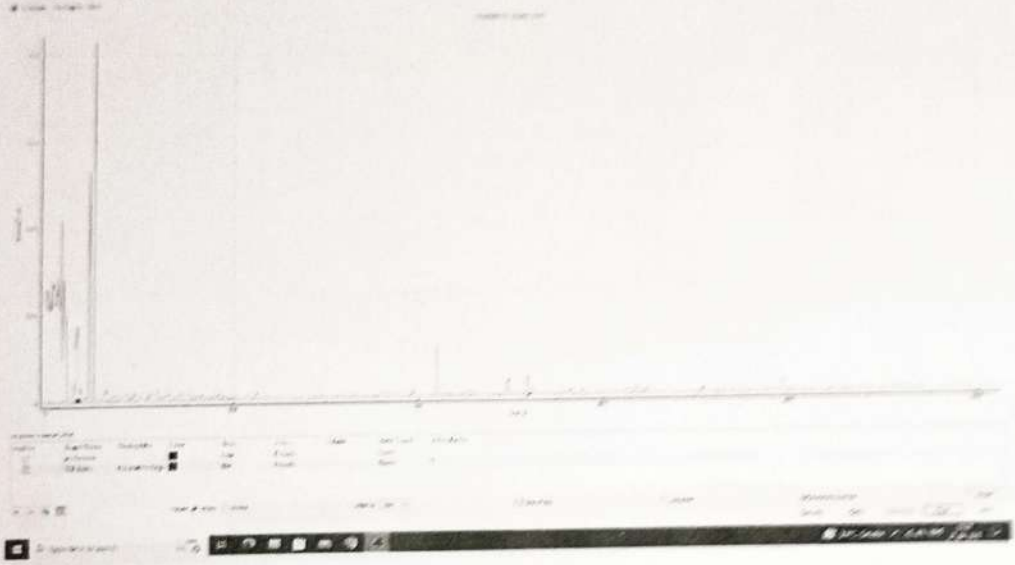
b. Port



c. Ack



6. I/O Graph:



Conclusion: Thus, we have simulated Packet Capturing in Wireshark.

9/4/25