COMPUTER NETWORK

S.NO.	DATE	EXPERIMENT TITLE	MARKS/ 10	SIGN.
1		Introduction to network		
2		Implement the data link layer framing methods such as character, character-stuffing and bit stuffing		
3		Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism		
4		Take an example subnet of hosts and obtain a broadcast tree for the subnet.		
5		Implement distance vector routing algorithm for obtaining routing tables at each node		
6		Implement data encryption and data decryption.		
9		Wireshark • Packet Capture Using Wire shark • Starting Wire shark • Viewing Captured Traffic • Analysis and Statistics & Filters.		

10		Do the following using NS2 Simulator • NS2 Simulator- Introduction • Simulate to Find the Number of Packets Dropped • Simulate to Find the Number of Packets Dropped by TCP/UDP • Simulate to Find the Number of Packets Dropped due to Congestion • Simulate to Compare Data Rate& Throughput. • Simulate to Plot Congestion for Different Source/Destination • Simulate to Determine the Performance with respect to Transmission of Packets		

Introduction to network

A computer network is a system that connects many independent computers to share information (data) and resources. The integration of computers and other different devices allows users to communicate more easily. A computer network is a collection of two or more computer systems that are linked together.

Types of Network

- **Local Area Network (LAN):** A LAN is a network that covers an area of around 10mtrs 200 mtr. For example, a college network or an office network. Depending upon the needs of the organization, a LAN can be a single office, building, or Campus. We can have two PCs and one printer in-home office or it can extend throughout the company and include audio and video devices. Each host in LAN has an identifier, an address that defines hosts in LAN. A packet sent by the host to another host carries both the source host's and the destination host's address.
- **Metropolitan Area Network (MAN):** MAN refers to a network that covers an entire city. For example: consider the cable television network.
- **Wide Area Network (WAN):** WAN refers to a network that connects countries or continents. For example, the Internet allows users to access a distributed system called www from anywhere around the globe.WAN interconnects connecting devices such as switches, routers, or modems.

1. Network Components

Basic hardware interconnecting network nodes, such as Network Interface Cards (NICs), Bridges, Hubs, Switches, Routers & cables are used in all networks.:

- **NIC (Network Interface Card):** A network card, often known as a network adapter or NIC (network interface card), is computer hardware that enables computers to communicate via a network. It offers physical access to networking media and, in many cases, MAC addresses serve as a low-level addressing scheme. Each network interface card has a distinct identifier. This is stored on a chip that is attached to the card.
- **Repeater:** A repeater is an electrical device that receives a signal, cleans it of unwanted noise, regenerates it, and retransmits it at a higher power level or to the opposite side of an obstruction, allowing the signal to travel greater distances without degradation. In the majority of twisted pair Ethernet networks, Repeaters are necessary for cable lengths longer than 100 meters in some systems. Repeaters are based on physics.
- **Hub:** A hub is a device that joins together many twisted pairs or fiber optic Ethernet devices to give the illusion of a formation of a single network segment. The device can be visualized as a multiport repeater. A network hub is a relatively simple broadcast device. Any packet entering any port is regenerated and broadcast out on all other ports, and hubs do not control any of the traffic that passes through them. Packet collisions occur as a result of every packet being sent out through all other ports, substantially impeding the smooth flow of communication.
- **Bridges:** Bridges broadcast data to all the ports but not to the one that received the transmission. Bridges, on the other hand, learn which MAC addresses are reachable through specific ports rather than copying messages to all ports as hubs do. Once a port and an address are associated, the bridge will only transport traffic from that address to that port.
- **Switches:** A switch differs from a hub in that it only forwards frames to the ports that are participating in the communication, rather than all of the ports that are connected. The collision domain is broken by a switch, yet the switch depicts itself as a broadcast domain. Frame-forwarding decisions are made by switches based on MAC addresses.
- **Routers:** Routers are networking devices that use headers and forwarding tables to find the optimal way to forward data packets between networks. A router is a computer networking device that links two or more computer networks and selectively exchanges data packets between them. A router can use address information in each data packet to determine if the source and destination are on the same network or if the data packet has to be transported between networks. When numerous routers are deployed in a wide collection of interconnected networks, the routers share target system addresses so that each router can develop a table displaying the preferred pathways between any two systems on the associated networks.
- **Gateways:** To provide system compatibility, a gateway may contain devices such as protocol translators, impedance-matching devices, rate converters, fault isolators, or signal translators. It also necessitates the development of administrative procedures that are acceptable to both networks. By completing the necessary protocol conversions, a protocol translation/mapping gateway joins networks that use distinct network protocol technologies.
- **Networking Operating System-**

  A **Network Operating System (NOS)** is a specialized software that controls and manages network resources, allowing different devices (computers, servers, printers, etc.) to communicate, share resources, and interact over a network.

2. Links

Links are the ways information travels between devices, and they can be of two types:

- **Wired:** Communication done in a wired medium. Copper wire, twisted pair, or fiber optic cables are all options. A wired network employs wires to link devices to the Internet or another network, such as laptops or desktop PCs.

  CROSSOVER PATCH CORD A **Crossover Patch Cord** (also known as a **Crossover Cable**) is a type of Ethernet cable designed to connect similar devices directly, such as connecting two **computers**, **network switches**, or **routers** without a hub or switch in between.

  STRAIGHT-THROUGH **PATCH CORD** A **straight-through cable** is a type of Ethernet cable commonly used in local area networks (LANs). It connects two **different types of network devices**, such as a **computer to a switch** or a **router to a switch**. It is one of the most commonly used cables in networking.

- **Wireless:** Wireless means without wire, media that is made up of electromagnetic waves (EM Waves) or infrared waves. Antennas or sensors will be present on all wireless devices. For data or voice communication, a wireless network uses radio frequency waves rather than wires.

3. Communication Protocols

A communication protocol is a set of rules that all devices follow when they share information. Some common protocols are TCP/IP, IEEE 802, Ethernet, wireless LAN, and cellular standards. TCP/IP is a model that organizes how communication works in modern networks. It has four functional layers for these communication links:

The OSI (Open Systems Interconnection) model and the TCP/IP model are both reference models used to describe how data travels across networks. They consist of different layers, each of which plays a specific role in facilitating communication between devices.

OSI Model (7 Layers)

The OSI model has **7 layers**, from the physical transmission of data to the application layer.

1. **Physical Layer (Layer 1)**:
   - Responsible for transmitting raw bits over a physical medium.
   - Deals with hardware like cables, switches, and electrical signals.
   - Examples: Ethernet cables, optical fiber, network adapters.
2. **Data Link Layer (Layer 2)**:
   - Responsible for node-to-node data transfer and error detection/correction.
   - It defines how data frames are transmitted over the physical medium.
   - Examples: MAC addresses, Ethernet, PPP (Point-to-Point Protocol).
3. **Network Layer (Layer 3)**:
   - Responsible for routing and forwarding data packets between devices across different networks.
   - Ensures data reaches its destination by using logical addresses (e.g., IP addresses).
   - Examples: IP (Internet Protocol), routers.
4. **Transport Layer (Layer 4)**:
   - Ensures reliable data transfer between end systems and handles error detection and correction.
   - Responsible for flow control, segmentation, and reassembly.
   - Examples: TCP, UDP (User Datagram Protocol).
5. **Session Layer (Layer 5)**:
   - Manages sessions between applications, i.e., establishing, maintaining, and terminating connections.
   - Responsible for synchronization and dialogue control.
   - Examples: NetBIOS, RPC (Remote Procedure Call).
6. **Presentation Layer (Layer 6)**:
   - Responsible for data translation, encryption, and compression.
   - Ensures that data is in a readable format for the application layer.
   - Examples: SSL/TLS, JPEG, ASCII.
7. **Application Layer (Layer 7)**:
   - The top layer that interacts directly with the end-user.

- Defines protocols and interfaces for applications to communicate.
- Examples: HTTP, FTP, SMTP, DNS.

TCP/IP Model (4 Layers)

The TCP/IP model, which is often used in modern networking, has **4 layers** and is more practical for real-world implementations, especially for the internet.

1. **Link Layer** (or Network Interface Layer):
   - Corresponds to the OSI's Physical and Data Link layers.
   - Responsible for handling physical addressing, access to physical media, and error detection/correction.
   - Examples: Ethernet, Wi-Fi, ARP.
2. **Internet Layer**:
   - Corresponds to the OSI's Network layer.
   - Responsible for logical addressing and routing of data packets across different networks.
   - The key protocol here is **IP** (Internet Protocol).
   - Examples: IP, ICMP (Internet Control Message Protocol), routers.
3. **Transport Layer**:
   - Corresponds to the OSI's Transport layer.
   - Handles end-to-end communication, error handling, and flow control.
   - Examples: TCP, UDP.
4. **Application Layer**:
   - Corresponds to the OSI's Session, Presentation, and Application layers.
   - Deals with end-user applications and provides services for communication.
   - Examples: HTTP, FTP, DNS, SMTP, POP3.

Common Networking Commands for Windows:

1. **ipconfig**:
   - Displays the current IP configuration of the computer, including IP address, subnet mask, and default gateway.
   - Common Usage: ipconfig or ipconfig /all (to see detailed information).
   - Example:

     bash
     Copy
     ipconfig

2. **ping**:
   - Sends ICMP Echo Request packets to a target IP address or domain to test connectivity.
   - Common Usage: ping [hostname or IP address]
   - Example:

bash
Copy
ping google.com

3. **tracert**:
   - Traces the route that data packets take to a destination, showing each hop along the way.
   - Common Usage: tracert [hostname or IP address]
   - Example:

     bash
     Copy
     tracert google.com

4. **nslookup**:
   - Queries DNS servers to obtain domain name or IP address information.
   - Common Usage: nslookup [domain]
   - Example:

     bash
     Copy
     nslookup google.com

5. **netstat**:
   - Displays active connections and listening ports on the system, along with network statistics.
   - Common Usage: netstat -an
   - Example:

     bash
     Copy
     netstat -an

6. **netsh**:
   - A command-line tool for network configuration and management in Windows.
   - Common Usage: netsh interface ip set address [interface name] static [IP address] [subnet mask] [default gateway]
   - Example:

     bash
     Copy
     netsh interface ip set address "Ethernet" static 192.168.1.10 255.255.255.0 192.168.1.1

7. **route**:
   - Displays or modifies the routing table in Windows.
   - Common Usage: route print to display the routing table.
   - Example:

bash
Copy
route print

8. **arp**:
   - Displays or modifies the ARP (Address Resolution Protocol) cache, mapping IP addresses to physical MAC addresses.
   - Common Usage: arp -a to display the ARP cache.
   - Example:

     bash
     Copy
     arp -a

9. **ipconfig /flushdns**:
   - Clears the DNS resolver cache, useful for troubleshooting DNS issues.
   - Example:

     bash
     Copy
     ipconfig /flushdns

10. **net use**:
    - Connects or disconnects network drives or resources.
    - Example:

      bash
      Copy
      net use Z: \\ServerName\ShareName

COMMON NETWORKING COMMANDS FOR LINUX/MACOS:

1. **ifconfig** (Linux) / **ip a** (Linux) / **ifconfig** (macOS):
   - Displays the network interface configuration (IP addresses, netmask, etc.).
   - Common Usage: ifconfig or ip a (Linux) or ifconfig (macOS).
   - Example (Linux):

     bash
     Copy
     ifconfig

2. **ping**:
   - Same as in Windows, sends ICMP Echo Request packets to a destination to check connectivity.
   - Common Usage: ping [hostname or IP address]
   - Example:

     bash
     Copy

ping google.com

3. **traceroute** (Linux/macOS):
   - Traces the route to a destination.
   - Common Usage: traceroute [hostname or IP address]
   - Example:

     bash
     Copy
     traceroute google.com

4. **nslookup**:
   - Used to query DNS to get the domain name or IP address of a resource.
   - Common Usage: nslookup [hostname]
   - Example:

     bash
     Copy
     nslookup google.com

5. **netstat**:
   - Displays network connections, routing tables, and interface statistics.
   - Common Usage: netstat -tuln
   - Example:

     bash
     Copy
     netstat -tuln

6. **route**:
   - Shows or modifies the system's routing table.
   - Common Usage: route -n (Linux) or netstat -r (macOS)
   - Example (Linux):

     bash
     Copy
     route -n

7. **dig** (Linux/macOS):
   - A more advanced tool than nslookup for querying DNS servers and gathering DNS information.
   - Common Usage: dig [domain]
   - Example:

     bash
     Copy
     dig google.com

8. **arp**:

- Displays or modifies the ARP cache (IP to MAC address mappings).
- Common Usage: arp -a to view the ARP table.
- Example:

```
bash
Copy
arp -a
```

9. **hostname**:
   - Displays or sets the system's hostname.
   - Common Usage: hostname
   - Example:

```
bash
Copy
hostname
```

10. **ip** (Linux):
    - A powerful utility for managing network interfaces, routing, and IP addresses.
    - Common Usage: ip addr show or ip route show.
    - Example (show IP address):

```
bash
Copy
ip addr show
```

11. **curl** (Linux/macOS):
    - A command-line tool for transferring data using various protocols like HTTP, FTP, etc. It can be used to check web server responses.
    - Common Usage: curl [URL]
    - Example:

```
bash
Copy
curl http://google.com
```

12. **systemctl** (Linux):
    - Used to control and manage services (including network services) on Linux.
    - Common Usage: systemctl restart network to restart networking services.
    - Example:

```
bash
Copy
sudo systemctl restart networking
```

General Troubleshooting and Network Diagnostic Commands:

1. **ping**:
    - Used in both Windows and Linux/macOS to check if a network device or host is reachable.
2. **traceroute** (Linux/macOS) / **tracert** (Windows):
    - Traces the route of packets to identify where delays or issues might be happening in the network.
3. **netstat**:
    - Displays active network connections, listening ports, and other network statistics.
4. **nslookup / dig**:
    - These are useful for checking DNS-related issues and getting information about domain names and IP addresses.
5. **ipconfig / ifconfig**:
    - Displays IP configuration details for troubleshooting network connectivity problems.
6. **route**:
    - Useful for checking and modifying the network routing table, which is important when diagnosing routing issues.

LINUX NETWORKING COMMAND

ip addr
ip a
To display the IP addresses on your system

Pin to IP address or domain
The ping command checks the availability of remote hosts by sending an ICMP echo request awaiting a reply.

hostname
The hostname command displays the hostname of a server

To implement data link layer framing methods like Character, Character-stuffing, and Bit -stuffing in C, we need to simulate how data is framed for transmission over a network. Let's break it down with examples for each of these methods.

1. Character Framing:

In character framing, special characters (like a start and stop character) are used to mark the beginning and end of a frame.

2. Character Stuffing:

Character stuffing is used to ensure that the special characters used for framing don't appear within the data. If the special character appears in the data, an extra "escape" character is inserted before it.

3. Bit Stuffing:

Bit stuffing ensures that a sequence of consecutive bits (like 5 consecutive 1s) does not appear in the data stream. If such a sequence appears, a `0` bit is inserted after 5 consecutive 1s.

---

1. Character Framing in C

Here's an implementation of character framing using a START and STOP character.

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

#define START_CHAR 'S'
#define STOP_CHAR 'E'

void apply_character_framing(char* data) {
    printf("%c", START_CHAR);
    for (int i = 0; i < strlen(data); i++) {
        printf("%c", data[i]);
    }
    printf("%c", STOP_CHAR);
}

int main() {
    char data[] = "HelloWorld";
    printf("Original data: %s\n", data);
    printf("Framed data: ");
    apply_character_framing(data);
    return 0;
}
```

2. Character Stuffing in C

Character stuffing involves inserting an escape character before any character that is the same as the framing characters (START_CHAR or STOP_CHAR).

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

#define START_CHAR 'S'
#define STOP_CHAR 'E'
#define ESCAPE_CHAR 'X'

void apply_character_stuffing(char* data) {
    printf("%c", START_CHAR);
```

```
    for (int i = 0; i < strlen(data); i++) {
        if (data[i] == START_CHAR || data[i] == STOP_CHAR) {
            printf("%c", ESCAPE_CHAR);  // Stuff the escape character
        }
        printf("%c", data[i]);
    }
    printf("%c", STOP_CHAR);
}

int main() {
    char data[] = "HelSloWorldE";
    printf("Original data: %s\n", data);
    printf("Stuffed data: ");
    apply_character_stuffing(data);
    return 0;
}
```

3. Bit Stuffing in C

In bit stuffing, we need to monitor the bit sequence and insert a `0` after five consecutive `1`s. Here's how we can do it:

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

void apply_bit_stuffing(char* data) {
    int bit_count = 0;
    printf("Framed data with bit stuffing: ");

    for (int i = 0; data[i] != '\0'; i++) {
        unsigned char byte = data[i];

        // Check each bit in the byte
        for (int j = 7; j >= 0; j--) {
            int bit = (byte >> j) & 1;
            printf("%d", bit);

            // If we encounter 5 consecutive 1s, insert a 0
            if (bit == 1) {
                bit_count++;
                if (bit_count == 5) {
                    printf("0"); // Stuff a zero after 5 consecutive ones
                    bit_count = 0; // Reset count
                }
            } else {
                bit_count = 0; // Reset count on 0
            }
        }
    }
}

int main() {
    char data[] = "HelloWorld";
```

```
    printf("Original data: %s\n", data);
    apply_bit_stuffing(data);
    return 0;
}
```

Explanation of Each Method:

1. **Character Framing**: We simply wrap the data with the START_CHAR and STOP_CHAR. It is straightforward and used when special characters (such as S and E) are reserved for this purpose.
2. **Character Stuffing**: We loop through the data and check for START_CHAR or STOP_CHAR. If we encounter these, we stuff them with an ESCAPE_CHAR to ensure they don't interfere with the framing characters.
3. **Bit Stuffing**: We read each byte of data and examine its bits. If five consecutive 1 bits are found, we insert a 0 to prevent this pattern from being misinterpreted as the end of a frame.

Sample Output for Each Program:

Character Framing:

kotlin
Copy
Original data: HelloWorld
Framed data: SHElloWorldE

Character Stuffing:

kotlin
Copy
Original data: HelSloWorldE
Stuffed data: XSHelloXWorldXE

Bit Stuffing:

kotlin
Copy
Original data: HelloWorld
Framed data with bit stuffing:
0100100001100101011011000110110001101111001000000101011101101111011101000110110001100100

This is how you can implement the three types of data link layer framing techniques in C.

Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP in network

To compute the CRC (Cyclic Redundancy Check) for different polynomials like CRC-12, CRC-16, and CRC-CCITT in a network communication context, we need to define the