Computer Networks Lab

Experiment 1:

Study of Network devices in detail and connect the computers in Local Area Network

Different type of network devices

Device	OSI Layer	Main Role	Intelligent Forwarding	Used In
Router	Layer 3	Routing between networks	Yes (IP)	LAN-WAN, Internet access
Switch	Layer 2/3	Internal LAN connectivity	Yes (MAC)	Office networks
Hub	Layer 1	Basic data broadcasting	No	Legacy LANs (now obsolete)
Modem	Layer 1/2	Internet signal conversion	No	ISP-provided connections
Access Point	Layer 2	Wireless access for devices	Yes	Wi-Fi environments
Repeater	Layer 1	Signal amplification	No	Signal extension
Bridge	Layer 2	LAN segment connection	Yes	Small or legacy networks
Gateway	All	Protocol translation	Yes	Enterprise/internet bridge
Firewall	Varies	Network security	Yes (Packet filtering)	Securing internal networks

Connecting to a Local Area Network

Manual (Static) IP Configuration

Windows

1. Go to Control Panel > Network and Sharing Centre > Change adapter settings
2. Right-click Ethernet > Properties
3. Select Internet Protocol Version 4 (TCP/IPv4) > Click **Properties**

Linux

1. Go to settings > wired Connected > wired settings > select the setting option in wired > click on IPv4.
2. Inspect the settings and ip address and subnet address as well (don't change any values).

Task	**Windows Command**	**Linux Command**
View IP info	ipconfig /all	ifconfig or ip addr
Test connection	ping [IP]	ping [IP]
View ARP cache	arp -a	arp -n
View routing table	route print	netstat -rn

Note: Execute the following commands and observe the data..

Experiment 2:

Write a Program to implement the data link layer framing methods such as

i) **Character stuffing**
ii) **ii) bit stuffing**

Character stuffing:

The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing.

Symbol	ASCII Value (Decimal)	Purpose
DLE	16 (0x10)	**Data Link Escape** – an escape character
STX	2 (0x02)	**Start of Text** – indicates frame start
ETX	3 (0x03)	**End of Text** – indicates frame end

Ex: Input – stuffed Data

[DLE][STX] ABC DLE DLE DEF [DLE][ETX]

Output:

ABCDEF

Code: Character stuffing

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

void main() {
   char input[50], stuffed[100], insertChar;
   int pos, i = 0, j = 0, len;

   // Input data
   printf("Enter input string (without spaces): ");
   scanf("%s", input);

   len = strlen(input);

   // Get position to insert new character
   do {
      printf("Enter position to insert character (1 to %d): ", len + 1);
      scanf("%d", &pos);
      if (pos < 1 || pos > len + 1)
         printf("Invalid position. Try again.\n");
   } while (pos < 1 || pos > len + 1);
   // Get character to insert
   printf("Enter the character to insert at position %d: ", pos);
   while ((getchar()) != '\n');  // Clear buffer before reading char
   insertChar = getchar();
   // Start frame with DLE STX
   strcpy(stuffed, "DLESTX");
   j = strlen(stuffed);
```

```
    for (i = 0; i < len; i++) {
        // Insert special character with DLE stuffing at specified position
        if (i == pos - 1) {
            strcat(stuffed + j, "DLE");
            j += 3;
            stuffed[j++] = insertChar;
            strcat(stuffed + j, "DLE");
            j += 3;
        }

        // Copy character from original input
        stuffed[j++] = input[i];
    }

    // If insertion position is at the end
    if (pos == len + 1) {
        strcat(stuffed + j, "DLE");
        j += 3;
        stuffed[j++] = insertChar;
        strcat(stuffed + j, "DLE");
        j += 3;
    }

    // End frame with DLE ETX
    strcat(stuffed + j, "DLEETX");
    printf("\nFrame after stuffing:\n%s\n", stuffed);
}
```

Bit Stuffing:

In protocols using special bit patterns to mark frame boundaries (like 01111110 for HDLC), we need to ensure that this pattern does not appear in the data itself.

- Bit stuffing rule:
  → After every five consecutive 1 bits, insert a 0.
  → The receiver removes any 0 that follows five consecutive 1s.

Code: Bit Stuffing and unstuffing

```
#include <stdio.h>

#include <string.h>


void bitStuffing(char input[], char stuffed[]) {
   int i = 0, j = 0, count = 0;


   while (input[i] != '\0') {
      stuffed[j++] = input[i];


      if (input[i] == '1') {
         count++;
         if (count == 5) {
            stuffed[j++] = '0'; // Stuff a 0 after five 1s
            count = 0;
         }
      } else {
         count = 0;
      }
      i++;
   }
   stuffed[j] = '\0';
```

```
}

void bitDeStuffing(char stuffed[], char destuffed[]) {
    int i = 0, j = 0, count = 0;

    while (stuffed[i] != '\0') {
        destuffed[j++] = stuffed[i];

        if (stuffed[i] == '1') {
            count++;
            if (count == 5) {
                i++; // Skip the stuffed 0
                count = 0;
            }
        } else {
            count = 0;
        }
        i++;
    }
    destuffed[j] = '\0';
}

int main() {
    char input[100], stuffed[150], destuffed[100];

    printf("Enter the binary data: ");
    scanf("%s", input);
```

```
    bitStuffing(input, stuffed);
    printf("After Bit Stuffing : %s\n", stuffed);

    bitDeStuffing(stuffed, destuffed);
    printf("After Bit De-Stuffing : %s\n", destuffed);

    return 0;
}
```