COMPUTER NETWORK
LAB EXPERIMENTS

HINDUSTAN
INSTITUTE OF TECHNOLOGY & SCIENCE
(DEEMED TO BE UNIVERSITY)

EXPERIMENT-01
USE COMMANDS LIKE TCPDUMP, NETSTAT, IFCONFIG, NS LOOKUP AND TRACEROUTE

Aim:-

To learn to use commands like tcpdump netstat, ifconfig num, nslookup and traceroute ping.

PRE LAB DISCUSSION:-

TCP dump:-

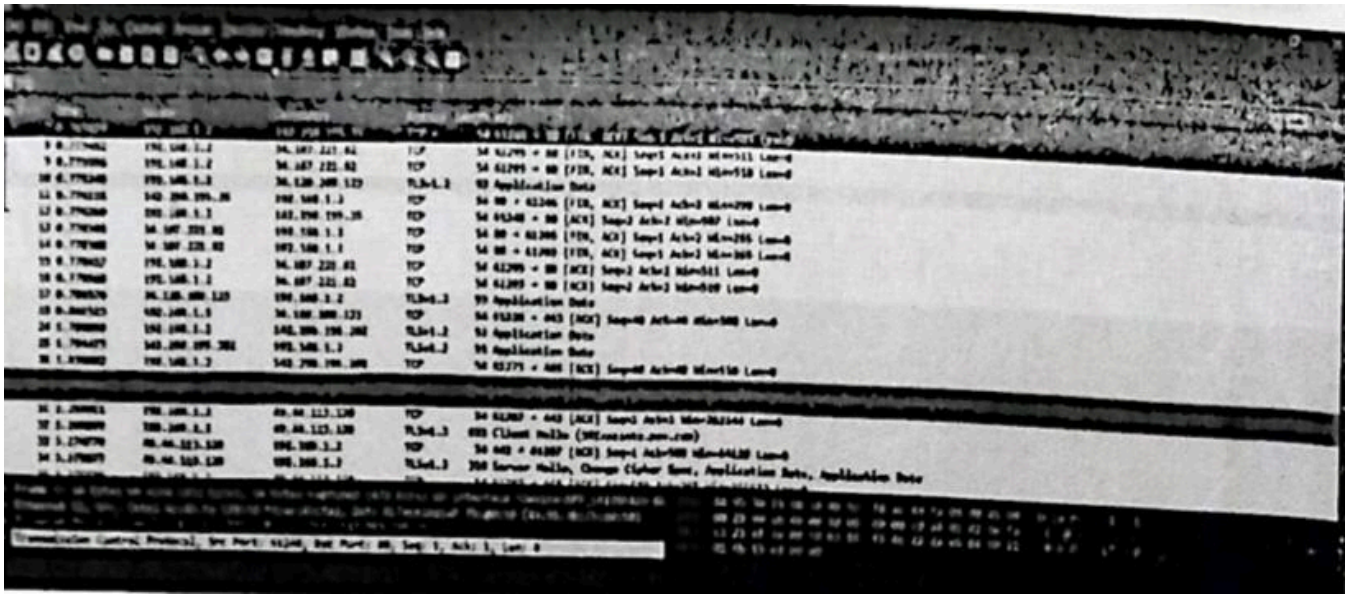
TCP dump is a command line utility that allows caplux and analyzes network traffic going through your system. It is often used to help troubleshoot network issues, as well as a security tool.

Netstat :-

Netstat is a common command line TCP/IP networking available is most versions of windows linux , unix & other operating systems. Netstat provides information & statistics about protocols in use & current TCP/IP network connections.

IPCONFIG-

IPconfig is a console application designed to run from the windows command prompt.



C:\Windows\system32\cmd.e

```
Microsoft Windows [Version 10.0.22631.3085]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Sarthak>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:1522         Sarthak:49786          ESTABLISHED
  TCP    127.0.0.1:49686        Sarthak:49687          ESTABLISHED
  TCP    127.0.0.1:49687        Sarthak:49686          ESTABLISHED
  TCP    127.0.0.1:49689        Sarthak:49690          ESTABLISHED
  TCP    127.0.0.1:49690        Sarthak:49689          ESTABLISHED
  TCP    127.0.0.1:49736        Sarthak:49737          ESTABLISHED
  TCP    127.0.0.1:49737        Sarthak:49736          ESTABLISHED
  TCP    127.0.0.1:49786        Sarthak:1522           ESTABLISHED
  TCP    192.168.1.2:49413      20.198.119.84:https    ESTABLISHED
  TCP    192.168.1.2:60372      ec2-3-6-211-252:https  ESTABLISHED
  TCP    192.168.1.2:61160      whatsapp-chatd-edge-shv-01-bom2:5222  ESTABLISHED
  TCP    192.168.1.2:61173      sb-in-f188:5228        ESTABLISHED
  TCP    192.168.1.2:61176      maa05s22-in-f10:https  TIME_WAIT
  TCP    192.168.1.2:61177      maa03s40-in-f3:https   TIME_WAIT
  TCP    192.168.1.2:61178      maa05s21-in-f14:https  ESTABLISHED
  TCP    192.168.1.2:61179      maa05s18-in-f3:https   ESTABLISHED
  TCP    192.168.1.2:61180      202.88.167.16:http     TIME_WAIT
  TCP    192.168.1.2:61181      maa03s39-in-f10:https  ESTABLISHED
  TCP    192.168.1.2:61182      maa03s29-in-f3:https   ESTABLISHED
```

This utility allows you to get the IP address information of windows computers.

NS Look Up:-

The nslookup (nome server lookup) command is a network utility program used to obtain information for domains by querying the Domain name system.

Trace route:-

Traceroute is a network diagnostic tool used to track the pathway taken by a packet on an IP network from source to destination. Trace route also records the time taken for each top of the packet maker during its route to the destination.

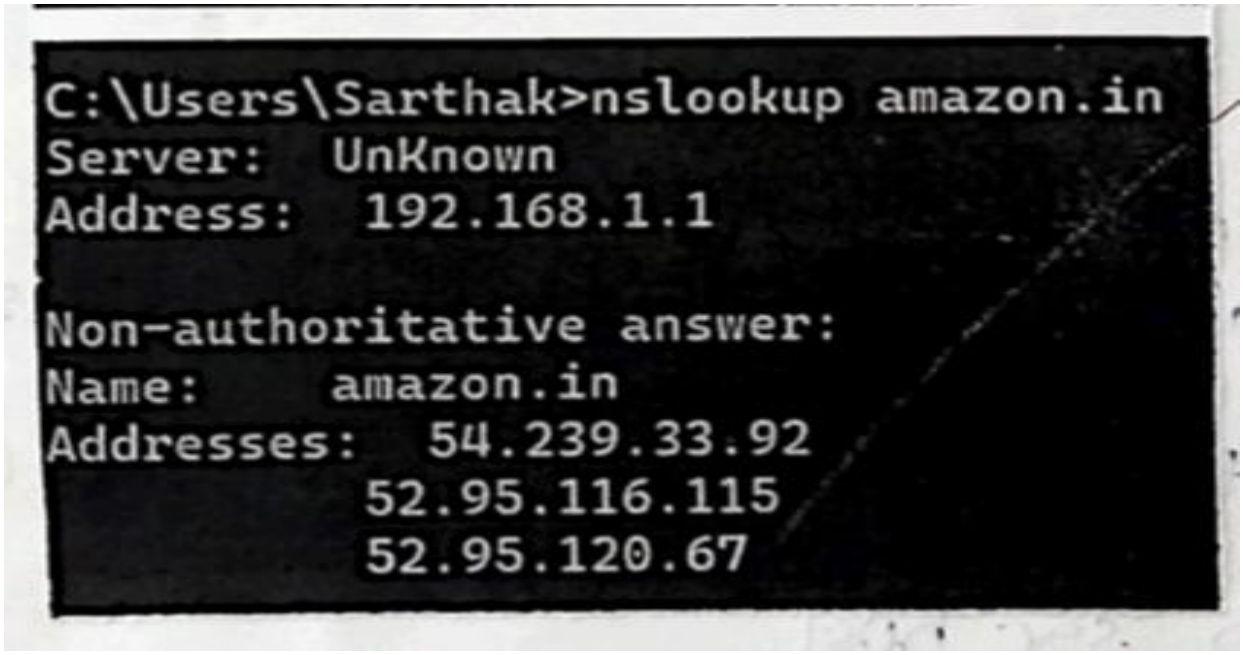
Commands

TCP dump [In linux]
*To display all traffic between two hosts :-
#top dump host 1 & host 2.
*To display traffic from only a source (src) or destination (dst) host:
#tcp dump src host.
#tcp dump dst host.

Net Stat:-

Netstat stands for network statistics.


Windows IP Configuration
Wireless LAN adapter Local Area Connection* 1:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Wireless LAN adapter Local Area Connection* 2:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :
Ethernet adapter VMware Network Adapter VMnet1:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::97e0:2739:8b06:d74a%2
IPv4 Address. . . . . : 192.168.234.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Ethernet adapter VMware Network Adapter VMnet8:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::d4a4:3c81:fb59:d008%19
IPv4 Address. . . . . : 192.168.30.1
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . :
Wireless LAN adapter Wi-Fi:
Connection-specific DNS Suffix . :
Link-local IPv6 Address . . . . . : fe80::3207:fa3e:a7d5:846b%10
IPv4 Address. . . . . : 192.168.1.2
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::1%10
192.168.1.1


C:\Users\Sarthak>nslookup amazon.in
Server: UnKnown
Address: 192.168.1.1
Non-authoritative answer:
Name: amazon.in
Addresses: 54.239.33.92
52.95.116.115
52.95.120.67

It allows users to display network related information & diagnose various network issues.

The windows help screen analogous to a linux or unix for netstat reads as follows: displays protocol Statistics & current TCP/IP network connections.

IPconfig:-

In windows, ipconfig is a console application designed to run from the windows prompt. This utility allows you to get the IP address information of a windows computer.

Using ipconfig:-

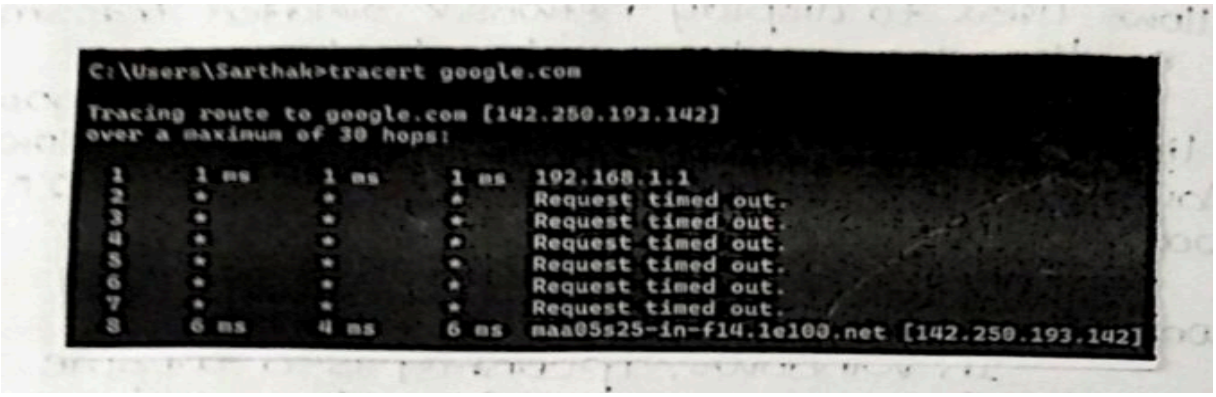
From the command prompt type ipconfig to run the utility with default options. The output of the

default command contains an IP address network mask, and gateway for all. physical & virtual network adapter.

Nslookup:-

The nslookup command is a network utility program used to obtain information for domains by querying the Domain name System.

The nslookup command is a powerful tool for diagnosing DNS problems.



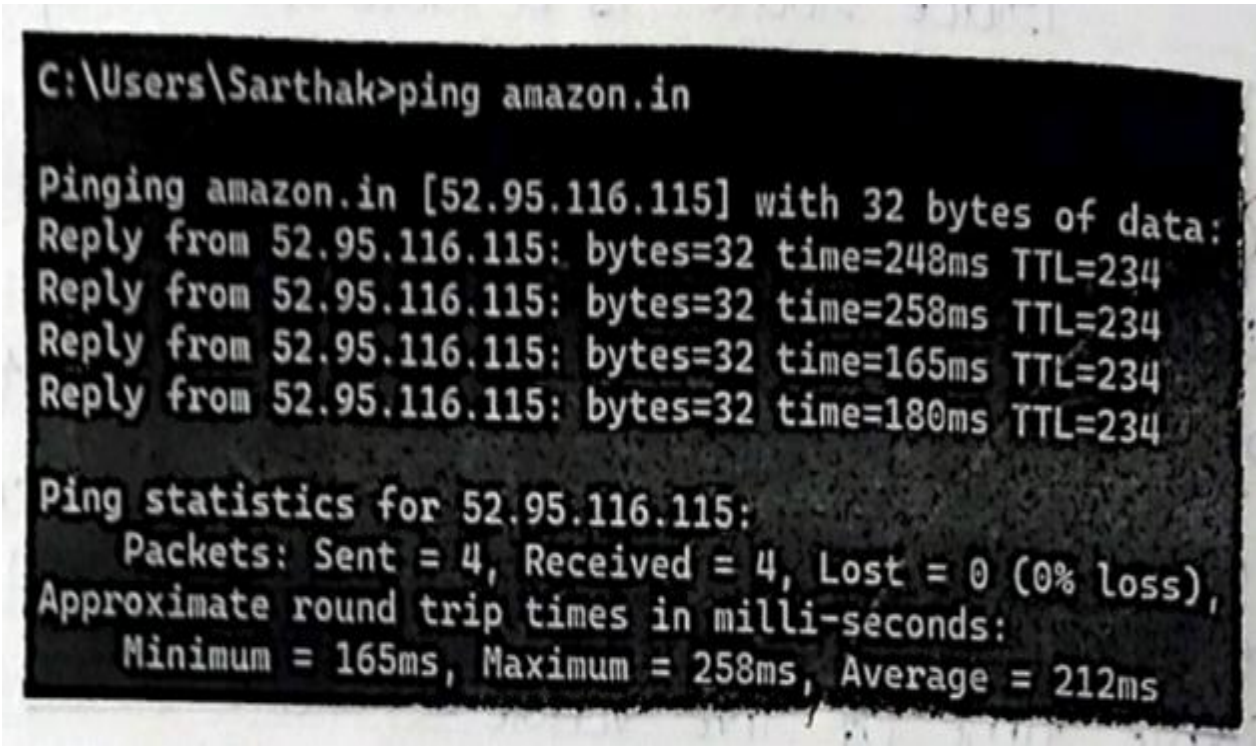
Trace route:-

Traceroute uses Internet control message protocol (ICMP) echo packets with variable time to live (TTL) values. The response time of each hop is calculated. To guarantee accuracy each hop is calculated. To guarantee queried multiple times to better measure the response of the particular hop.

Traceroute is a network diagnostics tool used to track the pathway taken by a packet on an IP network from source to destination. Trace route also records the time taken for each hop the packet makes during its route to the destination.
Traceroute uses Internet control message protocol (IMP) echo pockets until Variable time of live value.

The response time of each hop is called calculated to guarantee sends packets with TTL value that gradually increase from packet to packet starting with TTL value that gradually increases from packet to packet Starting with TTL value that gradually packets by one when routing & discarding. packet whose TTL Value has reached. zero, returning the ICMP error message ICMP time exceeded.

For the first set of packets the first router requires the packets, decrements the TTL value & drops the packet because it then has TTL value zero. The router sends an ICMP time exceeded message back to the source.



Ping:-

Determining the status of the network and various foreign hosts. The ping command is usually used as a simple way to verify that a computer can communicate over the network with another computer or network device. The ping command operates by sending an internet control message protocol (ICMP) echo request message to the destination computer & writing for a response.

RESULT:-

Thus, the various network commands. like tcpdump netstat, ifconfig, nslookup & trace route ping are executed successfully.

EXPERIMENT -02

DEVELOP A CHAT BETWEEN SERVER & CLIENT

Aim:

To write a C program to develop a chat between server & client.

Algorithm:-

→ **Server:-**

Step-1:- Start the program.

Step-2: Include necessary header files like stdio.h, netlin.h, stdlib.h, sys / Socket.h, sys/types.h.

Step-3:- Declare on int variable 'servsockd' to store the server socket.

Step-4: Declare a char array 'ser Msg' to store the message to be sent to the client.

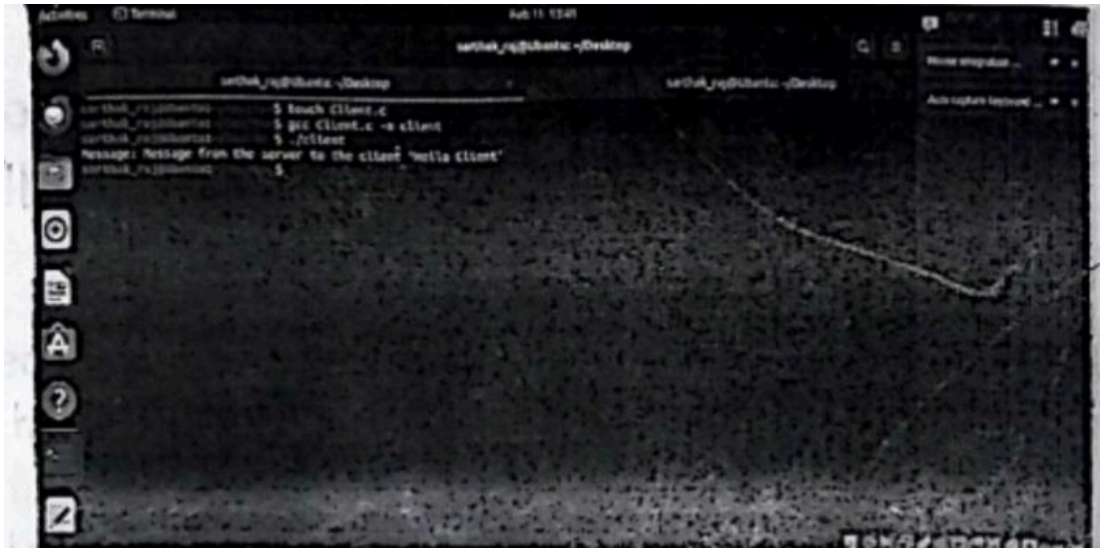
Step-5:- Set up the server address structure

Step-6: Create a socket using socket (AF-INET, sock adstream, 0)

Step-7:- Bind the socket to the specified IP & port using “bind".

Step-8: send the message to client using 'send'.

Step-9- Stop the program.



→Client

Step-1:- Start the program.

Step-2:- Include necessary header file.

Step-3: Declare an int variable 'Sock 0' to store the selected descriptor.

Step-3:- Declare a structure ‘mnr Add r’ to store the socket descriptor.

Step-5:-Set the port No using 'n tone'

Step-6:-Create a socket and store the returned descriptor in ‘sock 0’

Step-7:-Print the received message.

Step-8:- End the program.

Program:-

Client:-

```
#include <netinet/in.h> //structure for storing address information
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //for socket APIs
```

```
#include <sys/types.h>

int main(int argc, char const* argv[])
{
   int sockD = socket(AF_INET,
SOCK_STREAM, 0);

   struct sockaddr_in servAddr;

   servAddr.sin_family = AF_INET;
   servAddr.sin_port
      = htons(9001); // use some unused port
number
   servAddr.sin_addr.s_addr = INADDR_ANY;

   int connectStatus
      = connect(sockD, (struct
sockaddr*)&servAddr,
```

```
            sizeof(servAddr));

    if (connectStatus == -1) {
        printf("Error...\n");
    }

    else {
        char strData[255];

        recv(sockD, strData, sizeof(strData), 0);

        printf("Message: %s\n", strData);
    }

    return 0;
}
```

Server:-

```
#include <netinet/in.h> //structure for storing
address information
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h> //for socket APIs
#include <sys/types.h>

int main(int argc, char const* argv[])
{

   // create server socket similar to what was done
in
   // client program
   int servSockD = socket(AF_INET,
SOCK_STREAM, 0);

   // string store data to send to client
```

```
    char serMsg[255] = "Message from the server to
the "
                       "client \'Hello Client\' ";

    // define server address
    struct sockaddr_in servAddr;

    servAddr.sin_family = AF_INET;
    servAddr.sin_port = htons(9001);
    servAddr.sin_addr.s_addr = INADDR_ANY;

    // bind socket to the specified IP and port
    bind(servSockD, (struct sockaddr*)&servAddr,
        sizeof(servAddr));

    // listen for connections
    listen(servSockD, 1);
```

```
    // integer to hold client socket.
    int clientSocket = accept(servSockD, NULL,
NULL);

    // send's messages to client socket
    send(clientSocket, serMsg, sizeof(serMsg), 0);

    return 0;
}
```

Output:

Message from the server to the client
'Hello client!'

RESULT:

Thus the c program to develop chat between client & server has been executed successfully & output is verified.

EXPERIMENT-03
BIT STUFFING & CRC CONJUATION

Aim:

To write a C program that takes binary files as input & performs bit stuffing & CRC computation.

Algorithm:

Step-1:Start the program.
Step-2:Define a function "bit stuffing" that takes.
two parameters.
-~ The size of input array
-arr: an array of integers representing a binary tree.
Step-3 Declare an array bit to store the stuffed bits.

Step-4:Intiliare variable i, j,k to traverse array and j to 0.
Step-5:If the current bit is 1; Initiate variable name count to 1.
Step-7-After processing, print the stuffed bits stored in by br[].
Step-8:-Stop the program.

→For a bit destuffing

Step-1: Start the program.
Step-2:- Define a function bit destuffing that takes two parameters N-the size of the array
are:- an array of integers representing binary data.
Step-3:-Declare an array named 'br' to store destuffing bits.
Step-4:- Initialize a variable named count to 1 to store the count of consecutive Set bits.

Step-5:- If the current bits is 0 opened it to bar []

Step-6:- After processing print the stuffing bits stored in [].
Step-7-Stop the program.

C-program:-

c-program for bit stuffing

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

// Function for bit stuffing
void bitStuffing(int N, int arr[])
{
   // Stores the stuffed array
   int brr[30];
```

```
// Variables to traverse arrays
int i, j, k;
i = 0;
j = 0;

// Loop to traverse in the range [0, N)
while (i < N) {

    // If the current bit is a set bit
    if (arr[i] == 1) {

        // Stores the count of consecutive ones
        int count = 1;

        // Insert into array brr[]
        brr[j] = arr[i];

        // Loop to check for
```

```
        // next 5 bits
        for (k = i + 1;
             arr[k] == 1 && k < N && count < 5;
k++) {
            j++;
            brr[j] = arr[k];
            count++;

            // If 5 consecutive set bits
            // are found insert a 0 bit
            if (count == 5) {
               j++;
               brr[j] = 0;
            }
            i = k;
        }
    }
```

```
        // Otherwise insert arr[i] into
        // the array brr[]
        else {
            brr[j] = arr[i];
        }
        i++;
        j++;
    }

    // Print Answer
    for (i = 0; i < j; i++)
        printf("%d", brr[i]);
}

// Driver Code
int main()
{
    int N = 6;
```

```
    int arr[] = { 1, 1, 1, 1, 1, 1 };

    bitStuffing(N, arr);

    return 0;
}
```

OUTPUT:

1111101

C-program for bit deleting

```
// C program for the above approach
#include <stdio.h>
#include <string.h>

// Function for bit de-stuffing
void bitDestuffing(int N, int arr[])
{
    // Stores the de-stuffed array
```

```
int brr[30];

// Variables to traverse the arrays
int i, j, k;
i = 0;
j = 0;

// Stores the count of consecutive ones
int count = 1;

// Loop to traverse in the range [0, N)
while (i < N) {

    // If the current bit is a set bit
    if (arr[i] == 1) {

        // Insert into array brr[]
        brr[j] = arr[i];
```

```
// Loop to check for
// the next 5 bits
for (k = i + 1;
     arr[k] == 1
     && k < N
     && count < 5;
     k++) {
    j++;
    brr[j] = arr[k];
    count++;

    // If 5 consecutive set
    // bits are found skip the
    // next bit in arr[]
    if (count == 5) {
        k++;
    }
```

```
            i = k;
        }
    }

    // Otherwise insert arr[i] into
    // the array brr
    else {
        brr[j] = arr[i];
    }
    i++;
    j++;
  }

  // Print Answer
  for (i = 0; i < j; i++)
    printf("%d", brr[i]);
}
```

```
// Driver Code
int main()
{
    int N = 7;
    int arr[] = { 1, 1, 1, 1, 1, 0, 1 };

    bitDestuffing(N, arr);

    return 0;
}
```

OUTPUT:

```
1111111
```

RESULT:

Thus the C Program bit stuffing & CRC computation has been executed & output is verified.

EXPERIMENT-04
LAN CONNECTIVITY WITH PING

Aim:

To perform LAN connectivity with ping

Procedure:

Step-1:- First, open the cisco packet tracer desktop and select the devices given below.

S.no	**Device**	**Model name**	**Qty**
1	Pc	Pc	5
2	Switch	2960-24TT	1
3	Router	2911	1

IP Addressing Table for pcs of LAN 1 & LAN 2:

S.no	Device	IPV4 ADD	Subnet mask	Default gateway
1	PC0	13.12.11.1	255.255.255.0	13.12.11.5
2	PC1	13.12.11.1	255.255.255.0	13.12.11.5
3	PC2	13.12.11.3	255.255.255.0	13.12.11.5
4	PC3	13.12.11.4	255.255.255.0	13.12.11.5
5	PC4	13.12.11.4	255.255.255.0	13.12.11.5

->Then create two network topologies

->Use an Automatic connection cable to connect the device with others.

Step-2:-Configure the PCS (hosts) with IPV4 address, subnet mask, and Default gateway according to the IP addressing table above.

•To assign an IP address in PCO, click on pco. Then, go to desktop & IP configuration & there we will find IPV4 configuration.

• Fill IPV4, address, subnet mask, & default gateway to the particular input box.

Step-3:- Assigning IP address using the inconfig command.

•We can also assign an IP address with the help of a command.

Go to the command terminal of PCO.

•Then type inconfig & <IPV4 address> <subnet mask> <default gateway>

•Repeat the same procedure with other PCs of both LAN'S to configure them thoroughly.

Step4: Configure router with IP address & subnet mask.

• To assign an IP address in routers, click on router 0.

Then, go to config & then Interface.

Then, configure the IP address in fast Ethernet & serial ports according to the IP addressing table.

Step-5:-Verifying the network by pinging the IP address of any PC.

• First, click on PCO then go to the command prompt.

*Then type ping < IP address of target node>.

RESULT:

Thus, the LAN connectivity with ping has been successfully executed.

EXPERIMENT-05
SIMULATION Of SLIDING - WINDOW PROTOCOL

Aim:

To write a C program for simulation of sliding-window protocol.

Algorithm:-

Step-1:- Start the program.
Step-2:- Initialize the sender & receiver buffers.
Step-3:-Print “Simulation of sliding-window protocol”
Step-4-While next_frame to send <WINDOW_ SIZE
a) Send frames in the window.
b) Receive acknowledgement.

Step-5:-Print All frames have been successfully transmitted.
Step-6:-End the program.

Program :-

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>  // For sleep function

#define WINDOW_SIZE 4  // Sliding window size
#define TOTAL_FRAMES 10 // Total frames to send

typedef struct {
    int frame_id;
    bool sent;
```

```
    bool acked;
} Frame;

Frame buffer[WINDOW_SIZE]; // Buffer for the sliding window

void send_frame(Frame *frame);
bool receive_ack(Frame *frame);

int main() {
    int next_frame = 0;  // Frame ID to be sent next
    int base = 0;        // Base of the sliding window
    int acked = 0;       // Number of acknowledged frames

    printf("Sliding Window Protocol with Buffer Simulation\n");
    printf("Window Size: %d\n", WINDOW_SIZE);
```

```
   printf("Total Frames: %d\n\n",
TOTAL_FRAMES);

   while (acked < TOTAL_FRAMES) {
      // Load frames into the buffer
      for (int i = 0; i < WINDOW_SIZE; i++) {
         if (base + i < TOTAL_FRAMES &&
!buffer[i].sent) {
            buffer[i].frame_id = base + i;
            buffer[i].sent = false;
            buffer[i].acked = false;
         }
      }

      // Send frames within the window
      for (int i = 0; i < WINDOW_SIZE; i++) {
         if (!buffer[i].sent && base + i <
TOTAL_FRAMES) {
```

```
            printf("Sending frame %d\n",
buffer[i].frame_id);
            send_frame(&buffer[i]);
            buffer[i].sent = true;
        }
    }

    // Receive acknowledgments
    for (int i = 0; i < WINDOW_SIZE; i++) {
        if (buffer[i].sent && !buffer[i].acked) {
            if (receive_ack(&buffer[i])) {
                printf("Acknowledged frame %d\n",
buffer[i].frame_id);
                buffer[i].acked = true;
                acked++;
            } else {
                printf("Frame %d lost! Retransmitting
from frame %d\n", buffer[i].frame_id, base);
```

```
                // Retransmit all frames from the base
                for (int j = 0; j < WINDOW_SIZE; j++)
{
                    buffer[j].sent = false;
                    buffer[j].acked = false;
                }
                break;
            }
        }
    }

    // Slide the window
    while (buffer[0].acked) {
        for (int i = 1; i < WINDOW_SIZE; i++) {
            buffer[i - 1] = buffer[i];
        }
        buffer[WINDOW_SIZE - 1].sent = false; //
Clear the last slot
```

```
            base++;
        }

        printf("\n");
    }

    printf("All frames sent and acknowledged
successfully!\n");
    return 0;
}

void send_frame(Frame *frame) {
    // Simulate sending a frame
    sleep(1);
}

bool receive_ack(Frame *frame) {
```

```
    // Simulate random acknowledgment loss (10% chance of loss)
    return rand() % 10 >= 1;
}
```

Output:-

```
Sliding Window Protocol with Buffer Simulation
Window Size: 4
Total Frames: 10

Sending frame 0
Sending frame 1
Sending frame 2
Sending frame 3
Acknowledged frame 0
Acknowledged frame 1
Acknowledged frame 2
Acknowledged frame 3
```

RESULT:

Thus, the C program has been executed & output is verified successfully.

EXPERIMENT-06
SIMULATION OF THE BGP / OSPF ROUTING PROTOCOLS

Aim:

To write a c program for the simulation. of BGP/OSPE routings protocol.

Algorithm:-

Step-1:- Start the program.

Step-2:- Declare variables routers (MAX-NODES] to store router information & network [MAX_NODES] [MAX_NODES] to store network links & costs.

Step-3:-Initialize routers & network links.

Step-4:-Simulate OSPF routers to cydate routing tables.

Step-6:- Cyndate routing table function:

•Iterate through all nodes in the networks.

•If there is a direct link between the current router & another node.

(network [i][j] = INFINITY)

Step-7:-Update the routing table of the current router based on the received information using the ‘update routing Table' function.

Step-8:- Update routing table:

•Iterate through all nodes in the network.

•If the receiver costs of a node is less than the current cost in the routing table to the received cost.

•print the message indicating the update cost.

step-9:- Print the routing tables of all routers.

Step-10: End the program.

Program:-

```
#include <stdio.h>
#include <limits.h> // For INT_MAX
#include <stdbool.h>

#define MAX_NODES 10   // Maximum number
of nodes (routers)
#define INFINITY INT_MAX // Representation
of no direct link

// Structure to store router information
typedef struct {
   int
routing_table[MAX_NODES][MAX_NODES];
// Routing table for each router
} Router;
```

```
// Function declarations
void initialize_router_and_network(int
network[MAX_NODES][MAX_NODES], int
nodes);
void simulate_OSPF(Router *routers, int
network[MAX_NODES][MAX_NODES], int
nodes);
void update_routing_table(Router *router, int
network[MAX_NODES][MAX_NODES], int
nodes);
void print_routing_table(Router *routers, int
nodes);

int main() {
    int network[MAX_NODES][MAX_NODES];
    int nodes;
```

```
    printf("Enter the number of routers (max %d): ",
MAX_NODES);
    scanf("%d", &nodes);

    if (nodes > MAX_NODES || nodes <= 0) {
        printf("Invalid number of routers!\n");
        return 1;
    }

    Router routers[MAX_NODES]; // Array of
routers

    // Step 3: Initialize the network and routers
    initialize_router_and_network(network, nodes);

    // Step 4: Simulate OSPF to update routing
tables
    simulate_OSPF(routers, network, nodes);
```

```
    // Print final routing tables
    printf("\nFinal Routing Tables:\n");
    print_routing_table(routers, nodes);

    return 0;
}

// Step 3: Initialize router and network links
void initialize_router_and_network(int
network[MAX_NODES][MAX_NODES], int
nodes) {
    printf("Enter the network cost matrix (use %d for
no direct link):\n", INFINITY);
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            scanf("%d", &network[i][j]);
            if (i == j) {
```

```
            network[i][j] = 0; // Cost to itself is always
0
        }
    }
  }
}

// Step 4 & 5: Simulate OSPF and update routing
tables
void simulate_OSPF(Router *routers, int
network[MAX_NODES][MAX_NODES], int
nodes) {
  for (int i = 0; i < nodes; i++) {
    // Initialize each router's routing table
    update_routing_table(&routers[i], network,
nodes);
  }
}
```

```
// Step 6: Update routing table for a specific router
void update_routing_table(Router *router, int
network[MAX_NODES][MAX_NODES], int
nodes) {
    for (int i = 0; i < nodes; i++) {
        for (int j = 0; j < nodes; j++) {
            if (network[i][j] == INFINITY) {
                router->routing_table[i][j] = INFINITY;
// No direct link
            } else {
                router->routing_table[i][j] =
network[i][j]; // Direct link cost
            }
        }
    }
}
```

```
// Step 7: Print routing tables for all routers
void print_routing_table(Router *routers, int
nodes) {
   for (int i = 0; i < nodes; i++) {
      printf("\nRouting Table for Router %d:\n", i);
      printf("Destination\tCost\n");
      for (int j = 0; j < nodes; j++) {
         if (routers[i].routing_table[i][j] ==
INFINITY) {
            printf("%d\t\t%s\n", j, "INFINITY");
         } else {
            printf("%d\t\t%d\n", j,
routers[i].routing_table[i][j]);
         }
      }
   }
}
```

Output:

Enter the number of routers (max 10): 4

Enter the network cost matrix (use 2147483647 for no direct link):

1 4 2147483647 3

2147483647 3 4 5

1 2147483647 5 4

1 2 3 2147583647

Final Routing Tables:

Routing Table for Router 0:

Destination Cost

0 0

1 4

2 Infinity

3 3

Routing Table for Router 1:

Destination Cost

0 Infinity

1 0

2 4

3 5

Routing Table for Router 2:

Destination Cost

0 1

1 Infinity

2 0

3 4

Routing Table for Router 3:

Destination Cost

0 1

1 2

2 3

3 4

RESULT:

Thus, the program has been executed and Output is verified successfully.