Aim : Program to Implementation of the Data Link Layer Framing methods Bit stuffing

Program :

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100
// Function to perform bit stuffing
void bitStuffing(char input[], char output[]) {
   int i, j = 0, count = 0;
   int length = strlen(input);
      for (i = 0; i < length; i++) {
      // Copy the current bit to the output
      output[j] = input[i];
      j++;
            // If the current bit is '1', increase the count
      if (input[i] == '1') {
         count++;
      } else {
         // If the current bit is '0', reset the count
         count = 0;
      }
            // If there are 5 consecutive '1's, insert a '0'
      if (count == 5) {
         output[j] = '0';
         j++;
         count = 0;
      }
   }
      // Null terminate the output string
   output[j] = '\0';
}

int main() {
   char input[MAX_SIZE], output[MAX_SIZE];
      printf("Enter the input bit string: ");
   scanf("%s", input);
      bitStuffing(input, output);
      printf("Bit-stuffed output: %s\n", output);
```

```
    return 0;
}
```

Output :

```
Enter the input bit string: 11111100111111
Bit-stuffed output: 1111101001111101
```

Result: Program to Implementation of the Data Link Layer Framing methods Bit stuffing have been done successfully.

Aim : Program to Implementation of the Data Link Layer Framing methods Character Stuffing

```
#include <stdio.h>
#include <string.h>
#define MAX_SIZE 100
void charStuffing(char input[], char output[])
 {
    int i, j = 0;
    int length = strlen(input);
    char flag = 'F'; // Flag character
    char escape = 'E'; // Escape character
    // Add the starting flag to the output
    output[j++] = flag;
    for (i = 0; i < length; i++) {
        // If the current character is a flag or escape, add an escape character
        if (input[i] == flag || input[i] == escape) {
            output[j++] = escape;
        }
        // Copy the current character to the output
        output[j++] = input[i];
    }
    // Add the ending flag to the output
    output[j++] = flag;
    // Null terminate the output string
    output[j] = '\0';
}
```

```
int main() {
    char input[MAX_SIZE], output[MAX_SIZE];
    printf("Enter the input character string: ");
    scanf("%s", input);
    charStuffing(input, output);
    printf("Character-stuffed output: %s\n", output);
    return 0;
}
```

OUTPUT

Enter the input character string: SVCET

Character-stuffed output: FSVCEETF

Result: Program to Implementation of the Data Link Layer Framing methods Character Stuffing have been done successfully.

Aim : A program to implement CRC PROGRAM (Transmitter and Receiver)

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

int print(int *a,int n)
{
	int i;
	for(i=0;i<n;i++)
	{
		printf("%d",a[i]);
	}
	printf("\n");
	return (0);
}

int crc(int *g,int *q,int *r,int ng,int nq)
{
	int i,j,k;

	k = 0;

	for(i=0;i<nq;i++)
	{
		if(r[i]==0)
```

```
            {

                  for(j=i;j<(i+ng);j++)
                  {
                        r[j]=r[j]^0;
                  }

                  q[i]=0;
            }
            else
            {
                  for(j=i;j<(i+ng);j++)
                  {
                        r[j]=r[j]^g[k];
                        k++;
                  }
                  q[i]=1;
                  k=0;
            }
      }

      return (0);
}

 void main()
{
      int *gx,*tx,*q,*r;
      int i,j,nt,ng,nq,n,flag=1;

      printf("\n Enter no. of bits in message to be transmitted:");
      scanf("%d",&nt);
      printf("\n Enter no. of bits in G(x) :");
```

```
scanf("%d",&ng);


n=nt+ng-1;
nq=nt;
gx=malloc(sizeof(int)*ng);
tx=malloc(sizeof(int)*n);
r=malloc(sizeof(int)*n);
q=calloc(nq,sizeof(int));



printf("\n Enter message :");
for(i=0;i<nt;i++)
{
      scanf("%d",&tx[i]);
      r[i]=tx[i];
}
for(;i<n;i++)
{
      r[i]=0;
}

printf("\n Enter G(x) :");
for(i=0;i<ng;i++)
{
      scanf("%d",&gx[i]);
}

/* *****AT TRANSMITTER***** */
printf("\n CRC at transmitter :");
printf("\n Message to be transmitted :");
print(tx,nt);
printf("\n G(x)=");
```

```
print(gx,ng);
printf("\n Message with '0' appended :");
print(r,n);

crc(gx,q,r,ng,nq);

printf("\n Quotient at transmitter:");
print(q,nq);
printf("\n Remainder at transmitter :");
print(r,n);

for(i=0;i<nt;i++)
{
      r[i]=tx[i];
}

printf("\n Transmitted message :");
print(r,n);

/* ***** AT RECEIVER***** */
printf("\n CRC at receiver :");
printf("\n Message received :");
print(r,n);
printf("\n G(x)=");
print(gx,ng);

crc(gx,q,r,ng,nq);

printf("\n Quotient at receiver :");
print(q,nq);
printf("\n Remainder at receiver :");
print(r,n);
```

```
	for(i=0;i<n;i++)
	{
		if(r[i]!=0)
		{
			flag=0;
			break;
		}
	}

	if(flag)
		printf("\n No error detected -->CRC algorithm implemented
successfully.");
	else
		printf("\n Error detected.");

	getchar();

}
```

Sample Input and Output

Enter no. of bits in message to be transmitted:8

Enter no. of bits in G(x) :6

Enter message :10110111

Enter G(x) :110011

Quotient at transmitter:11010111

Remainder at transmitter :0000000001001

Transmitted message :1011011101001

CRC at receiver :

Message received :1011011101001

G(x)=110011

Quotient at receiver :11010111

Remainder at receiver :0000000000000

No error detected -->CRC algorithm implemented successfully.

...Program finished with exit code 0

Result: **A program to implement CRC PROGRAM (Transmitter and Receiver) have been done successfully.**

Aim : Implementation of Sliding Window Protocol Select Repeat ARQ
Program

```
#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#define WINDOW_SIZE 4

#define FRAME_COUNT 8

// Data structure for frames

typedef struct {

   int sequence_number;

   bool is_acknowledged;

   char data;

} Frame;


// Sender function

void sender(Frame frames[], int total_frames) {

   int base = 0;

   int next_seq = 0;


   while (base < total_frames) {

      // Send frames in the window

      for (int i = base; i < base + WINDOW_SIZE && i < total_frames; i++) {
```

```
            if (!frames[i].is_acknowledged) {
                printf("Sending frame %d: %c\n", frames[i].sequence_number, frames[i].data);
            }
        }

        // Simulate acknowledgment reception
        int ack;
        printf("Enter the last acknowledged frame number (or -1 to exit): ");
        scanf("%d", &ack);

        if (ack == -1) {
            break;
        }

        // Mark frames as acknowledged
        for (int i = base; i <= ack; i++) {
            frames[i].is_acknowledged = true;
        }

        // Slide the window
        while (base < total_frames && frames[base].is_acknowledged) {
            base++;
        }
    }
}

// Receiver function
void receiver(Frame frames[], int total_frames) {
    for (int i = 0; i < total_frames; i++) {
```

```
        printf("Received frame %d: %c\n", frames[i].sequence_number, frames[i].data);
    }
}

int main() {
    Frame frames[FRAME_COUNT];

    // Initialize frames with data and sequence numbers
    for (int i = 0; i < FRAME_COUNT; i++) {
        frames[i].sequence_number = i;
        frames[i].is_acknowledged = false;
        frames[i].data = 'A' + i;
    }

    printf("Sender:\n");
    sender(frames, FRAME_COUNT);

    printf("\nReceiver:\n");
    receiver(frames, FRAME_COUNT);

    return 0;
}
```

```
Sender:
Sending frame 0: A
Sending frame 1: B
Sending frame 2: C
Sending frame 3: D
Enter the last acknowledged frame number (or -1 to exit): 2
Sending frame 3: D
```

Sending frame 4: E

Sending frame 5: F

Sending frame 6: G

Enter the last acknowledged frame number (or -1 to exit): 5

Sending frame 6: G

Sending frame 7: H

Enter the last acknowledged frame number (or -1 to exit): 7

Receiver:

Received frame 0: A

Received frame 1: B

Received frame 2: C

Received frame 3: D

Received frame 4: E

Received frame 5: F

Result: Implementation of Sliding Window Protocol Select Repeat ARQ Program have been done successfully.

AIM

Implement Dijkstra ‘s algorithm to compute the Shortest path thru a graph.

Program:

```
#include<conio.h>
#include<stdio.h>
static int dsp[10][10],nodes,perm,tem;
struct
{
   char src;
   char dest;
   int length;
}  stemp,permanent[10]={' ',' ',0},temp[10]={' ',' ',-1};
void sort()
 {
    int i,j,k;
    for(i=0;i<=tem;i++)
    {
```

```
      k=1;
      for(j=0;j<=tem;j++)
       {
         if((temp[j].length<=temp[j+1].length))
         {
            stemp=temp[j];
            temp[j]=temp[j+1];
            temp[j+1]=stemp;
           k=0;
         }
}
       if(k)
       break;
    }
 permanent[perm++]=temp[tem-1];
 temp[tem-1].src=' ';
 temp[tem-1].dest=' ';
 temp[tem-1].length=-1;
tem--;
       }
void main()
{
int i,j,k,l,m,n=0,point;
char initial,dest,path[10]={' '};
clrscr();
printf("\t\t Shortest path dijkstra's algothrim");
printf("\n ************************************\n");
printf("Enter the number of nodes:\n");
scanf("%d",&nodes);
```

```
printf("\nEnter the adjacency matrix");
for(i=0;i<nodes;i++)
{
   for(j=0;j<nodes;j++)
       scanf("%d",&dsp[i][j]);
}
fflush(stdin);
printf("\n Enter the source node");
scanf("%c",&initial);
fflush(stdin);
printf("\n Enter the destination node\n");
scanf("%c",&dest);
permanent[perm].src=initial;
permanent[perm].dest=initial;
permanent[perm++].length=0;
i=permanent[perm-1].dest-97;
for(j=0;j<nodes;j++)
 {
   if(i!=j)
   {
    if(dsp[i][j]>0)
{
       temp[tem].src=permanent[perm-1].src;
       temp[tem].dest=j+97;
       temp[tem++].length=dsp[i][j];
    }
    }
}
sort();
```

```
  while(tem>=0)
  {
  j=permanent[perm-1].dest-97;
  for(i=0;i<nodes;i++)
{
  if(i!=initial-97)
   {
     if(dsp[j][i]>0)
      {
        l=-1;
       for(k=0;k<perm;k++)
        {
           if(permanent[k].dest==(i+97))
           l=k;
        }



    for(k=0;k<=tem;k++)
     {
       if(temp[k].dest==(i+97))
       l=k;
     }
     if(l<0)
     {
      temp[tem].src=j+97;
      temp[tem].dest=i+97;
      for(m=0;m<perm;m++)
      {
```

```
        if(permanent[m].dest==temp[tem].src)
                n=permanent[m].length;
        }
       temp[tem++].length=dsp[j][i]+n;
     }
    else
    {
    for(m=0;m<perm;m++)
{
     if(permanent[m].dest==j+97)
     {
         n=permanent[m].length+dsp[j][i];
         break;
     }
     else
     n=dsp[j][i];
    }
     if((n<temp[l].length))
       {
         temp[l].length=n;
         temp[l].src=j+97;
        tpaemp[l].dest=i+97;
    }  }  }  }  }
sort();
    }
 printf("\n shortest path :\n\n");
 printf("from %c to %c is:",initial,dest);
 for(i=0;i<perm-1;i++)
 {
```

```
if(permanent[i].dest==dest)
      {
        point=i;
        n=i;
        break;
      }
     }
   i=0;
   for(j=perm;j>0;j--)
   {
     if(permanent[j-1].dest==permanent[point].src)
       {
          path[i++]=permanent[point].dest;
         point=j-1;
     } }
   path[i]=initial;
   for(j=i;j>=0;j--)
     {
        printf("%c",path[j]);
        if(j>0)
          printf("----->");      }
     printf("\t   length=%d",permanent[m].length);
     getch();
  }
```

Output:

```
                Shortest path dijkstra's algothrim
 ****************************************
Enter the number of nodes:
3

Enter the adjacency matrix0
1
0
1
0
1
1
1
0

 Enter the source nodea

 Enter the destination node
c

 shortest path :

from a to c is:a----->b----->c     length=2
```

Result: Implement Dijkstra 's algorithm to compute the Shortest path thru a graph have been done successfully

Aim : Implementation Link State routing algorithm.

Program:

```
#include<stdio.h>
Main()
{
int n, a[10][10],i,j,k;
printf("\n Enter the no.of nodes:");
scanf("%d",&n);
printf("Enter the matrix elements:");
for(i=0;i<n;i++)
{
Printf("\n Enter the distance for node: %d \n",i+1);
For(j=0;j<n;j++)
{
Scanf("%d",&a[i][j]);
}
}
For(i=0;i<n;i++)
{
Printf("The link state packets for node:%d \n",i+1);
```

```
Printf("\n node \t distance \n");
For(j=0;j<n;j++)
{
If(a[i][j]!=0&&a[i][j]!=-1)
{
Printf("%d \t %d \n",j+1,a[i][j]);
}
}
Printf("\n \n");
}
}
```

Output:

```
Enter the no.of nodes:3
Enter the matrix elements:
Enter the distance for node : 1
                    1     2     3
Enter the distance for node:2
                    2     3     4
Enter the distance for node:3
                    3     4     5
The link state packets for node: 1
                    Node        Distance
                    1           1
                    2           2
                    3           3
The link state packets for node: 2
```

Node	Distance
1	2
2	3
3	4

The link state packets for node: 3

Node	Distance
1	3
2	4
3	5

Result: Implementation Link State routing algorithm have been done successfully.

AIM

Obtaining the vector routing table using Distance Vector Routing Algorithm by taking an example subnet graph with weights .

Program:

```
#include"stdio.h"
#include"conio.h"
void main()
{
 int i=0,j=0,nei,n,ah[20];
 int v[50][50],r[20],min=10000,si;
 char s[50],dest,line[50];
 for(i=0;i<50;i++)
   s[i]= NULL;
 clrscr();
 printf("\nEnter the no of nodes:\n");
 scanf("%d",&n);
```

```
printf("\nEnter the destination:\n");
scanf("%s",&dest);
printf("\nEnter the no.of neighbours:\n");
scanf("%d",&nei);
printf("\nEnter the neighbours & slash(0) at end:\n");
for(i=0;i<=nei;i++)
  scanf("%s",s[i]);
for(j=0;j<nei;)
{
  printf("\nThe distance from %c to %s :",dest,s[j]);
  scanf("%d",&ah[j]);
  j++;   }

printf("\nEnter the table of entries:");
for(i=0;i<n;i++)
  for(j=0;j<nei;j++)
    scanf("%d",&v[i][j]);
  for(i=0;i<n;i++)
     {
        min=10000;
        for(j=0;j<nei;j++)
          if(v[i][j]+ah[j]<min)
```

```
          {
            min=v[i][j]+ah[j];
            line[i]=s[j];
          }r[i]=min;
        if(dest-97==i)
          {     r[i]=0;
           line[i]=' ';
        }
    }
 printf("\n new estimated delays from %c    |",dest);
 printf("    Line\n_____________________________________
_____________________\n\n");
 for(i=0;i<n;i++)
 printf("\n \t%d\t\t\t|    %s",r[i],line[i]);
 getch();
 }
```

OUTPUT:

```
Enter the no of nodes:
12

Enter the destination:
J
Enter the no.of neighbours:
4

Enter the neighbours & slash(0) at end:
A
I
H
K

The distance from J to A :8

The distance from J to I :10

The distance from J to H :12

The distance from J to K :6

Enter the table of entries:
0  12 25 40 14 23 18 17 21 9  24 29
24 36 18 27 7  20 31 20 0  11 22 33
20 31 19 8  30 19 6  0  14 7  22 9
21 28 36 24 22 40 31 19 22 10 0  9
```

new estimated delays from J	Line
8	A
22	A
19	I
30	H
15	I
8	I
14	H
6	H
15	I
0	
25	K
12	K

Result: Obtaining the vector routing table using Distance Vector Routing Algorithm by taking an example subnet graph with weights have been done successfully.

AIM

Take a 64 bit playing text and encrypt the same using DES algorithm

. **Program**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
        int i,ch,lp;
        char cipher[50],plain[50];
        char key[50];
        clrscr();
        while(1)
        {
                printf("\n-----MENU-----");
                printf("\n1:Data Encryption\t2:Data Decryption\t3:Exit");
                printf("\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
```

```
        {
            case 1:     printf("\nData Encryption");
                        printf("\nEnter the plain text:");
                        fflush(stdin);
                        gets(plain);
                        printf("\nEnter the encryption key:");
                        gets(key);
                        lp=strlen(key);
                        for(i=0;plain[i]!='\0';i++)
                                cipher[i]=plain[i]^lp;
                        cipher[i]='\0';
                        printf("\nThe encrypted text is:");
                        puts(cipher);
                        break;

                       case 2:printf("\nData decryption");
                        for(i=0;cipher[i]!='\0';i++)
                                plain[i]=cipher[i]^lp;
                        printf("\nDecrypted text is:");
                        puts(plain);
                        break;

                      case 3: exit(0);
        }
    }

}
```

Output:

-------Menu--------

1. **Data Encryption**
2. **Data Decryption**
3. **Exit**

Enter your Choice: 1

Data Encryption

Enter the plain text: rupesh

Enter the encryption key: hsepur

The encrypted text is: tsvcun

-------Menu--------

1. **Data Encryption**
2. **Data Decryption**
3. **Exit**

Enter your Choice: 2

Data Decryption

Decrypted text is: rupesh

-------Menu--------

1. **Data Encryption**
2. **Data Decryption**
3. **Exit**

Enter your Choice: 3

Result: Take a 64 bit playing text and encrypt the same using DES algorithm have been done successfully

AIM

Using RSA algorithm Encrypt a text data and Decrypt the same

Program:

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

int phi,M,n,e,d,C,FLAG;

void check()
{
int i;
for(i=3;e%i==0 && phi%i==0;i+2)
{
FLAG = 1;
return;
}
FLAG = 0;
}

void encrypt()
{
int i;
C = 1;
for(i=0;i< e;i++)
C=C*M%n;
C = C%n;
printf("\n\tEncrypted keyword : %d",C);
```

```
}

void decrypt()
{
int i;
M = 1;
for(i=0;i< d;i++)
M=M*C%n;
M = M%n;
printf("\n\tDecrypted keyword : %d",M);
}

void main()
{
int p,q,s;
clrscr();
printf("Enter Two Relatively Prime Numbers\t: ");
scanf("%d%d",&p,&q);
n = p*q;
phi=(p-1)*(q-1);
printf("\n\tF(n)\t= %d",phi);
do
{
printf("\n\nEnter e\t: ");
scanf("%d",&e);
check();
}while(FLAG==1);
d = 1;
do
{
s = (d*e)%phi;
d++;
}while(s!=1);
d = d-1;
printf("\n\tPublic Key\t: {%d,%d}",e,n);
printf("\n\tPrivate Key\t: {%d,%d}",d,n);
printf("\n\nEnter The Plain Text\t: ");
scanf("%d",&M);
encrypt();
printf("\n\nEnter the Cipher text\t: ");
scanf("%d",&C);
decrypt();
getch();
}
```

OUTPUT :

Enter Two Relatively Prime Numbers : 7 17

F(n) = 96

Enter e : 5

Public Key : {5,119}
Private Key : {77,119}

Enter The Plain Text : 19

Encrypted keyword : 66

Enter the Cipher text : 66

Result : Using RSA algorithm Encrypt a text data and Decrypt the same have been done successfully.

Experiment-9

Aim: implementation of basic network command and Network configuration commands.

Apparatus (Software): Command Prompt And Packet Tracer.

Procedure: To do this EXPERIMENT- follows these steps:

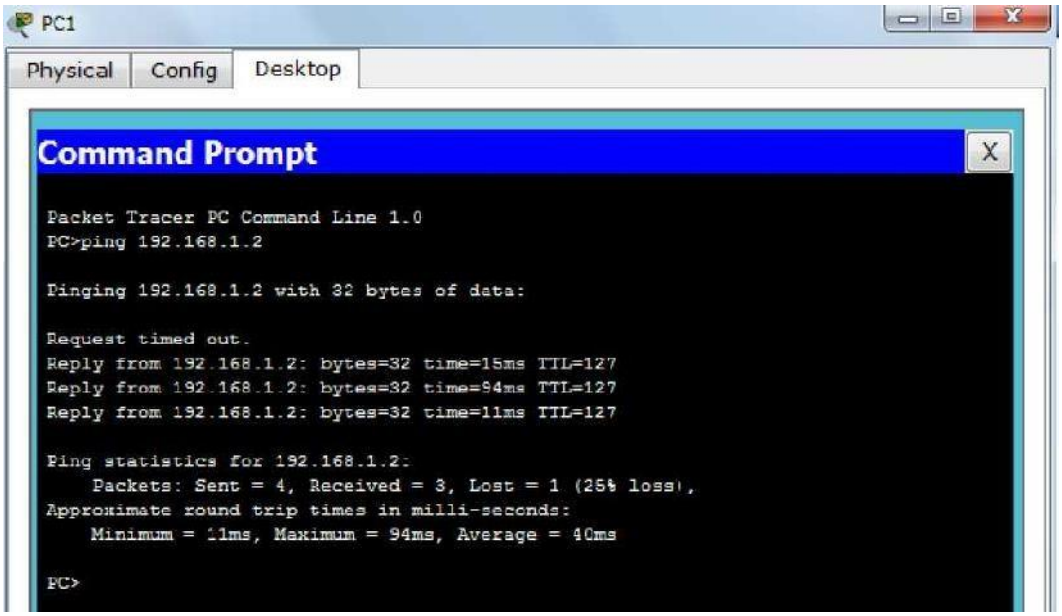
In this EXPERIMENT- students have to understand basic networking commands e.g ping, tracert etc.

All commands related to Network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this configuration to flash memory or permanent memory.

This commands includes

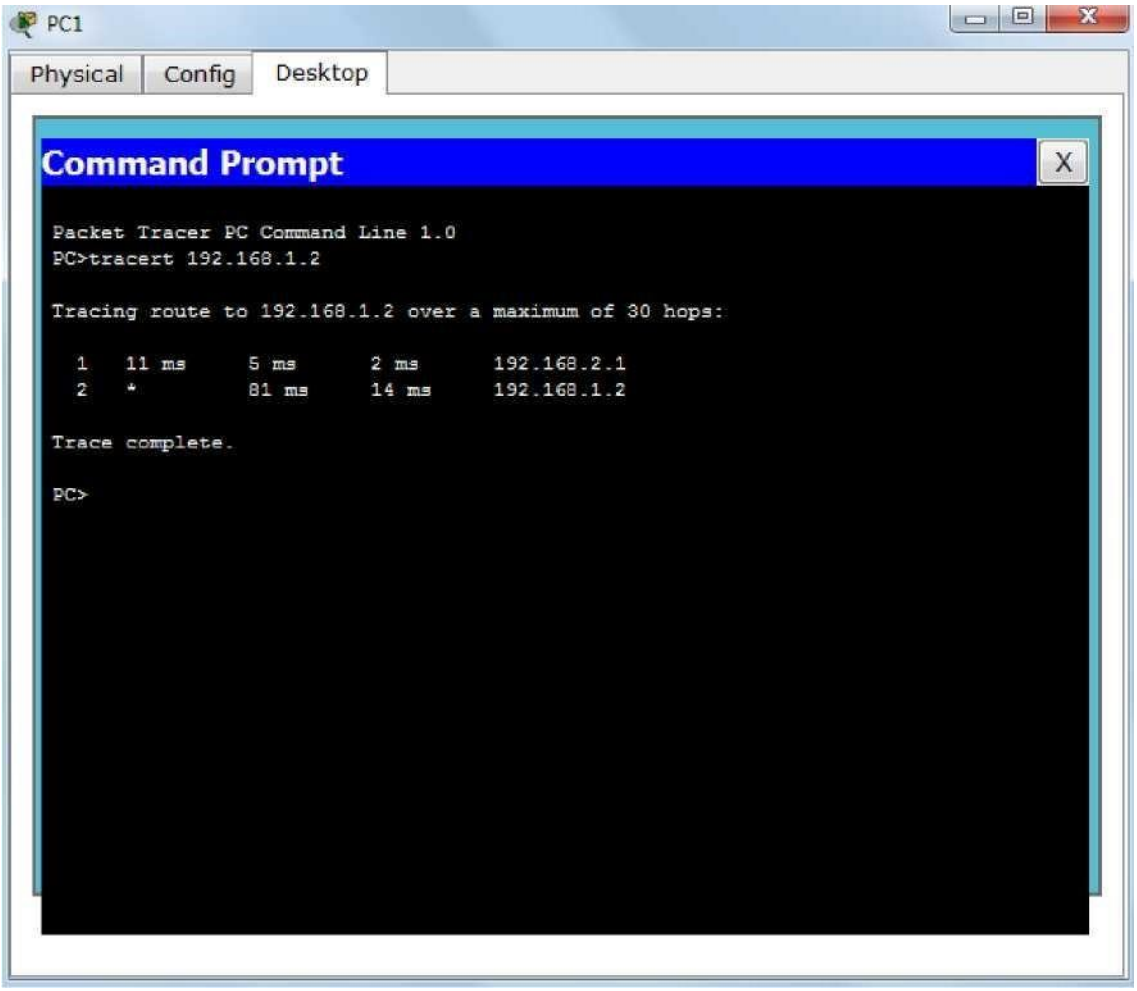
- Configuring the Router commands
- General Commands to configure network
- Privileged Mode commands of a router
- Router Processes & Statistics
- IP Commands
- Other IP Commands e.g. show ip route etc.

1.ping: ping(8) sends an ICMP ECHO_REQUEST packet to the specified host. If the host responds, you get an ICMP packet back. Sound strange? Well, you can “ping” an IP address to see if a machine is alive. If there is no response, you know something is wrong.



2.Traceroute:

Tracert is a command which can show you the path a packet of information takes from your computer to one you specify. It will list all the routers it passes through until it reaches its destination, or fails to and is discarded. In addition to this, it will tell you how long each 'hop' from router to router takes.



3.nslookup:

Displays information from Domain Name System (DNS) name servers.
NOTE If you write the command as above it shows as default your pc's server name firstly.

C:\Users\Thyagarajan>nslookup
Default Server: dns.google
Address: 2001:4860:4860::8888

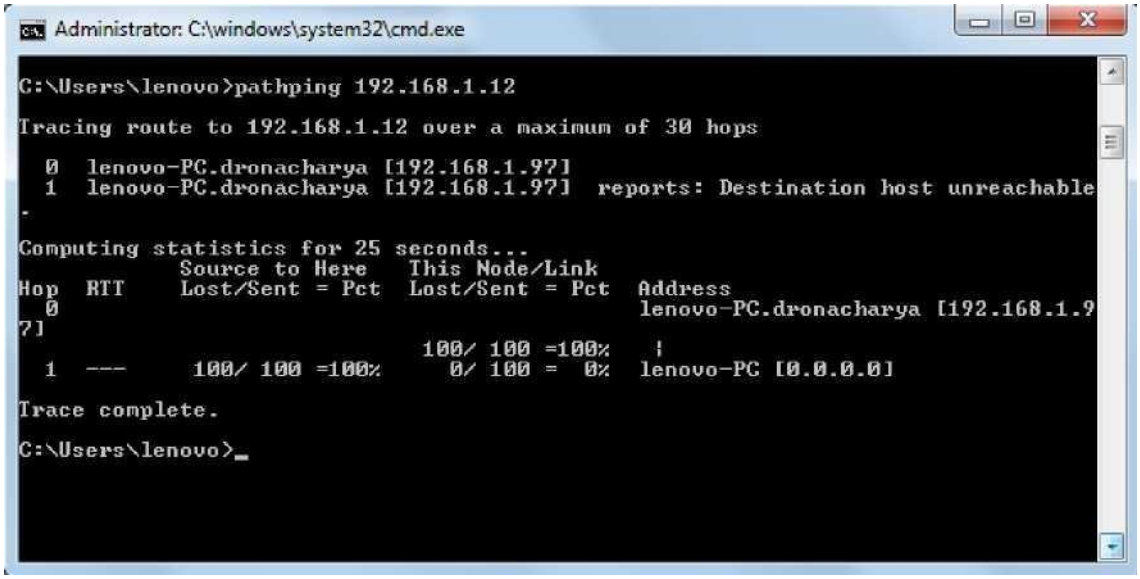
> nslookup www.svcetedu.org
Server: svcetedu.org
Address: 43.225.55.220
Aliases: www.svcetedu.org

*** www.svcetedu.org can't find nslookup: Query refused
> nslookup www.gmail.com
Server: googlemail.l.google.com
Addresses: 2404:6800:4002:80d::2005
        172.217.167.197
Aliases: www.gmail.com
        mail.google.com

DNS request timed out.
   timeout was 2 seconds.
DNS request timed out.
   timeout was 2 seconds.
*** Request to www.gmail.com timed-out

4.pathping:

A better version of tracert that gives you statics about packet lost and latency.



Result : implementation of basic network command and Network configuration commands have be done successfully.

EXPERIMENT-10

Ex. No. 10. Configure Host IP, Subnet Mask and Default Gateway in a System in LAN (TCP/IP Configuration).

Aim:

To Configure IP Address in a system in LAN (TCP/IP Configuration) and Configure DNS to establish

interconnection between systems

Principle: Following is required to be study under this practical.

• Classification of IP address

Class A 1.0.0.1 to 126.255.255.254 Supports 16 million hosts on each of 127 networks.

Class B 128.1.0.1 to 191.255.255.254 Supports 65,000 hosts on each of 16,000 networks.

Class C 192.0.1.1 to 223.255.254.254 Supports 254 hosts on each of 2 million networks.

Class D 224.0.0.0 to 239.255.255.255 Reserved for multicast groups.

Class E 240.0.0.0 to 254.255.255.254 Reserved.

• Sub netting

- Subnet ting is dividing the network into two or more networks is called subnet ting.

Procedure:

(a) Steps to configure IP address, Subnet mask and Default Gateway:

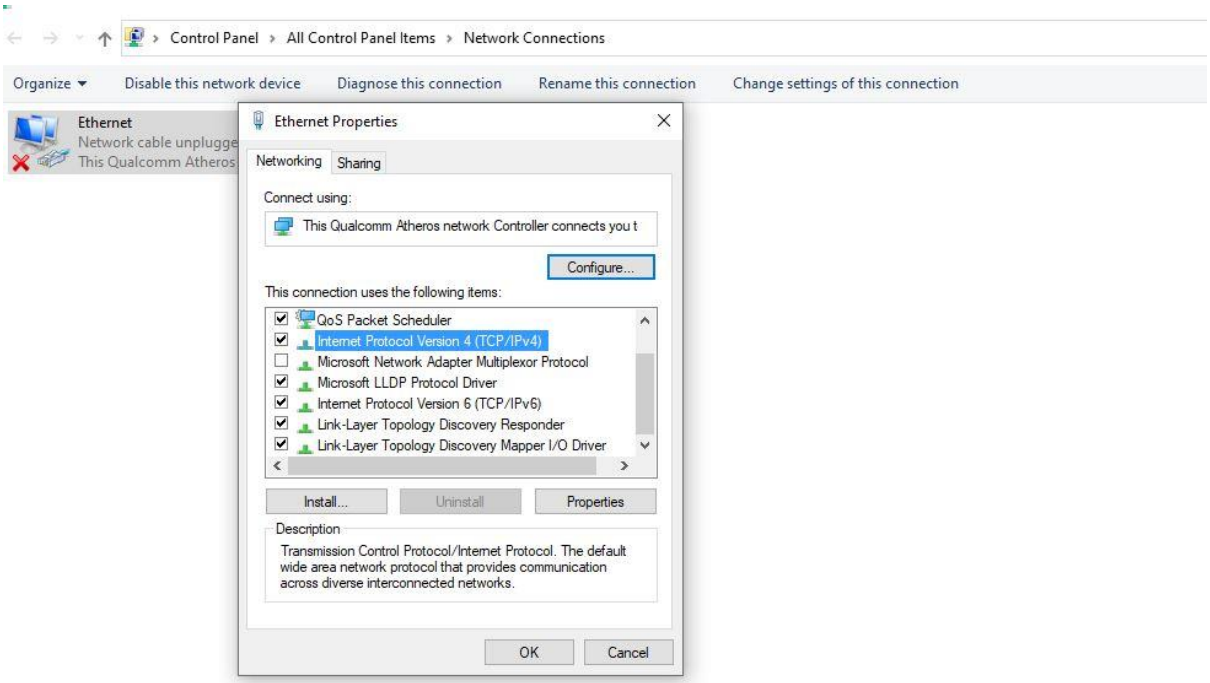
1. Click on the Start button and select Control Panel then Network and Internet Connections.

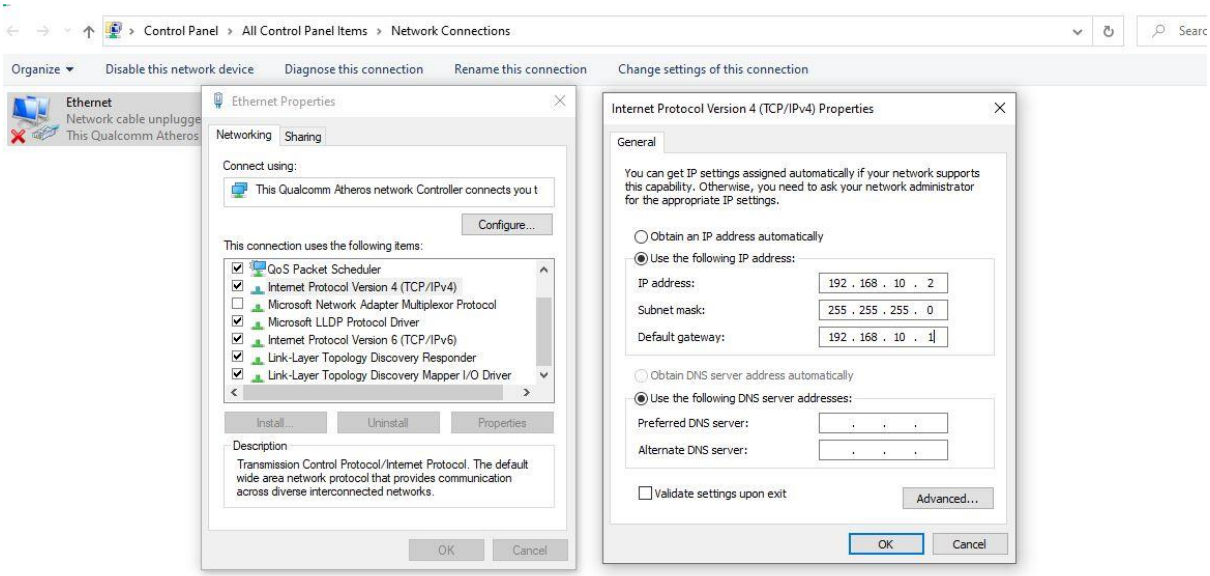
2. Click Network and Internet Connections.

3. Right click on the Local Area Connection icon and select Properties.

4. Select Internet Protocol (TCP/IP).

5. Click on the Properties button.





6. Uncheck that Obtain an IP address automatically and Obtain DNS server address automatically and put IP, Subnet mask & Default Gateways.

7. Click on the Advanced button and select the DNS tab in the Advanced TCP/IP Settings window.

8. Ensure that Register this connection's addresses in DNS is not selected.

9. Click OK, OK, then Close to close all boxes.

Result :

Configuration of IP Address in a system in LAN (TCP/IP Configuration) and Configuration to establish interconnection between systems have been done successfully