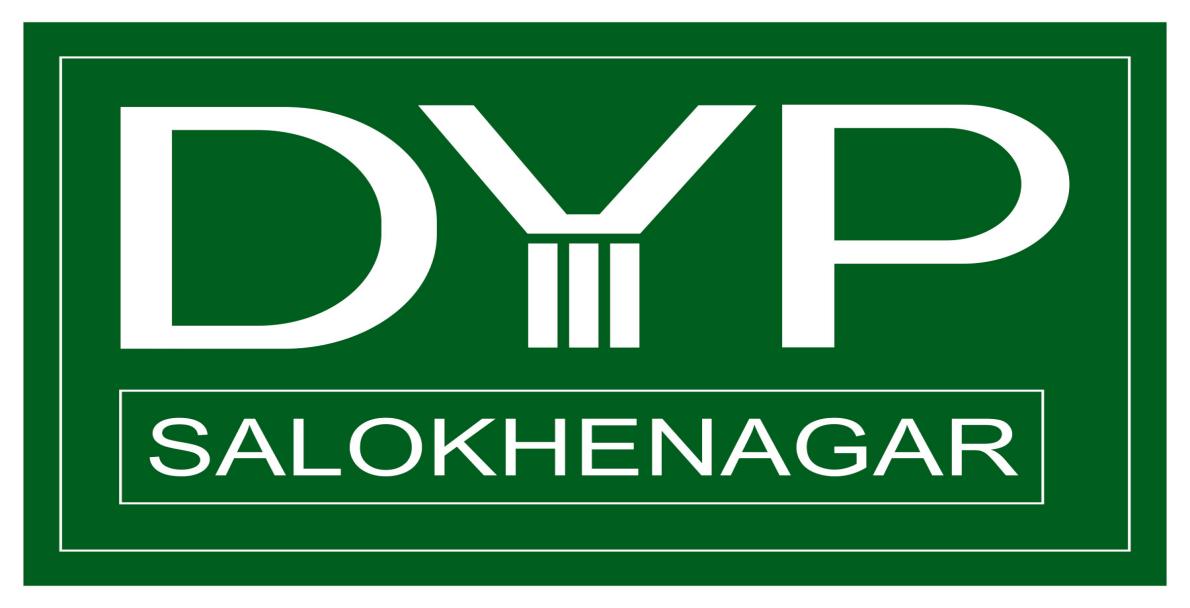
**Dr.D. Y. PATIL PRATISHTHAN’S COLLEGE OF ENGINEERING, SALOKHENAGAR, KOLHAPUR**



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING (DATA SCIENCE)

LABORATORY MANUAL

COMPUTER NETWORKS

SECOND YEAR

SEM-III

YEAR: 2022-23

|  |  |  |
| --- | --- | --- |
|  | Dr. D. Y. Patil Pratishthan’s College of Engineering, Salokhe Nagar, Kolhapur. | |
| Doc. No.: DYP-ACAD-FRM | Rev. No:00 |
| PAGE 1 of | Rev. Date: |
| Department of CSE(Data Science) | Year:2022-23 |
| Subject-Computer Networks | Class- SY SEM-III |

**Lab Name** :Computer Networks  **Lab In-charge**: Prof. P. R. Kamble

**Computer Networks Experiment List**

|  |  |
| --- | --- |
| **Sr.**  **No** | **Title of Experiment** |
| 01 | Study and design the types of Computer Networks (LAN,WAN etc.) and various connecting devices and components required for it (LAN,WAN etc.) |
| 02 | Study of network connectivity test tools: ping, nslookup, ipconfig, arp, route, tracert, nmap, netstat, finger. |
| 03 | Study of Framing Methods & Implementation of Character count framing method. |
| 04 | Study of Framing Methods & Implementation of Bit stuffing methods. |
| 05 | Implementation of Error detecting & correcting code-Hamming Code |
| 06 | Implementation of Cyclic Redundancy code (CRC) Error Detection method |
| 07 | Implementation of Sliding Window Protocol. |
| 08 | Study & Implementation of the Shortest Path Routing Algorithm (Dijkstra,s algorithm) . |
| 09 | Study and Installation of Cisco Packet tracer tool. |
| 10 | Implementation of Classful and classless(CIDR) Subnetting by using Cisco Packet Tracer tool and classful addressing by using program. |
| 11 | Study and Installation of network analyzer tool Wireshark. |
| 12 | Implementation of TCP client-server socket programming. |
| 13 | Implementation of UDP client-server socket programming. |

**Experiment No. 1**

**Title**:- Study and design the types of Computer Networks (LAN,WAN etc.) and various connecting devices and components required for it (LAN,WAN etc.).

**Aim:-** Study and design the types of computer networks and its connecting devices.

**Objective:-** To know the basics of computer networks and networking control devices**.**

**Relevance:-**Types of Computer Networks and Networking Control Devices.

**Theory**:-

**Introduction**:

A network is a collection of computers connected to each other. The network allows

computers to communicate with each other and share resources and information

* **Network Classification:**

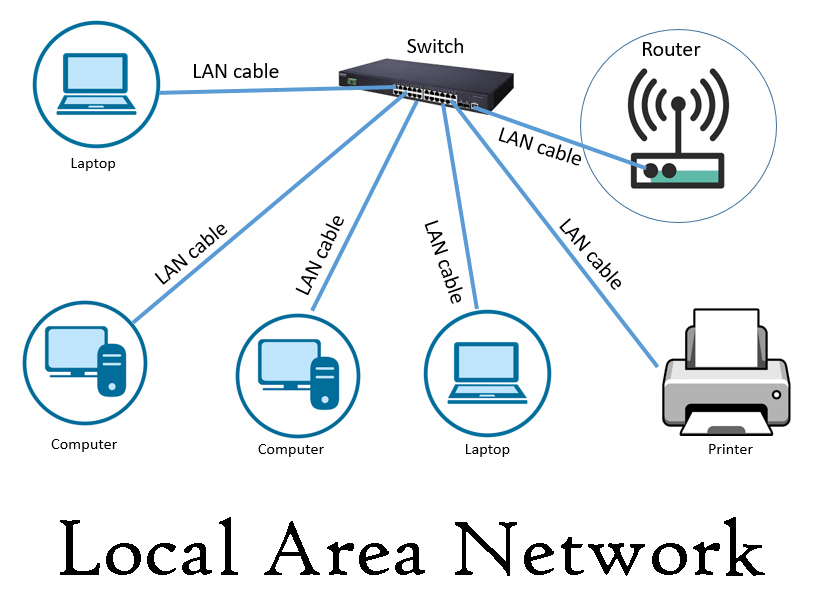
Based on their scale, networks can be classified as Local Area Network (LAN),

Wide Area Network (WAN), Metropolitan Area Network (MAN), Personal Area

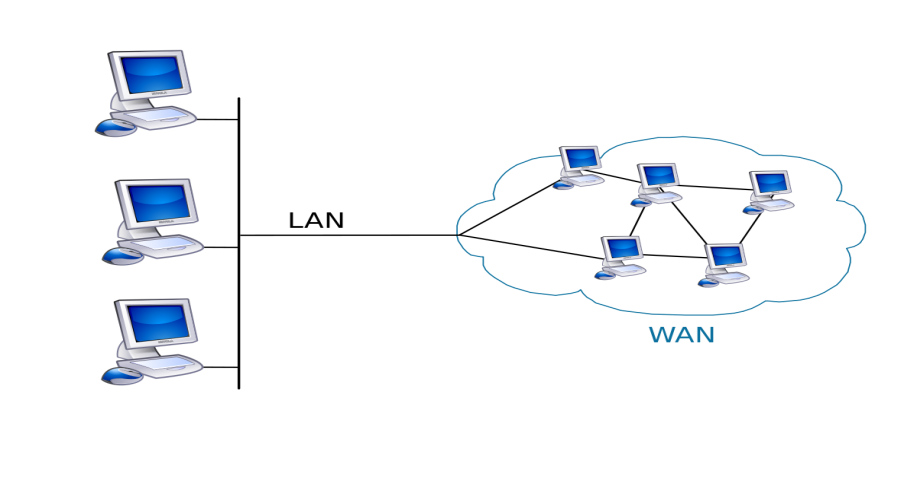
Network ( PAN) etc.

**LAN:**

A local area network (LAN) is a collection of devices connected together in one physical location, such as a building, office, or home. A LAN can be small or large, ranging from a home network with one user to an enterprise network with thousands of users and devices in an office or school.

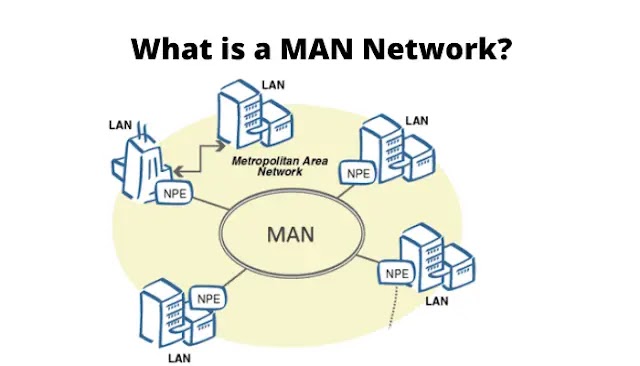


**WAN:**

In its simplest form, a wide-area network (WAN) is a collection of [local-area networks](https://www.cisco.com/c/en_in/products/switches/what-is-a-lan-local-area-network.html) (LAN's) or other networks that communicate with one another.  A WAN is essentially a network of networks, with the Internet the world's largest WAN.

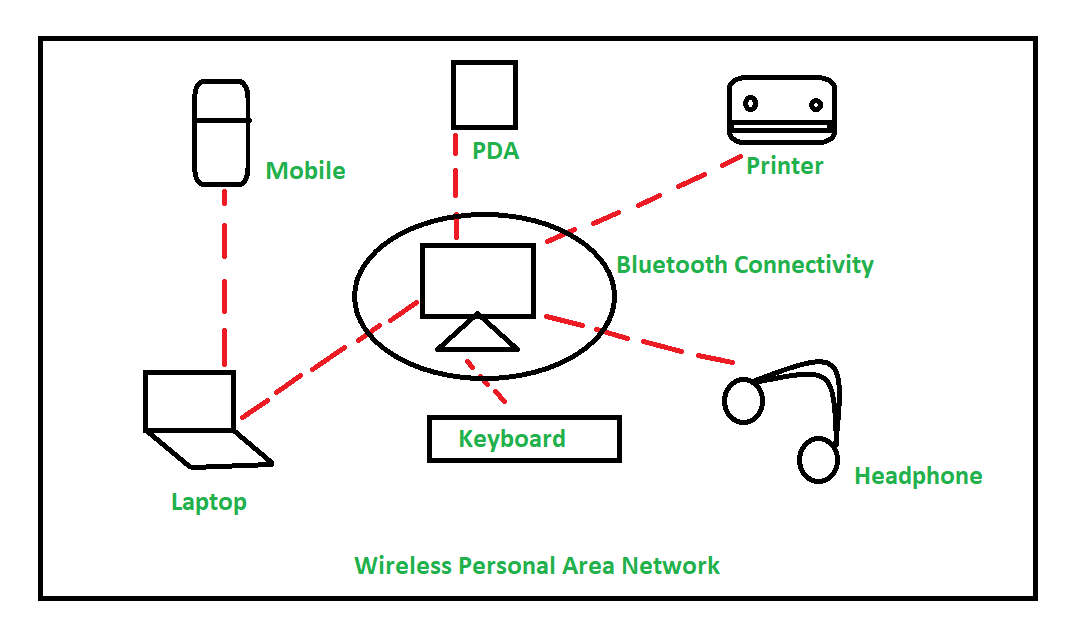
**MAN:**

A metropolitan area network (MAN) is a computer network that connects computers within a metropolitan area, which could be a single large city, multiple cities and towns, or any given large area with multiple buildings. A MAN is larger than a [local area network (LAN)](https://www.cloudflare.com/learning/network-layer/what-is-a-lan/) but smaller than a [wide area network (WAN)](https://www.cloudflare.com/learning/network-layer/what-is-a-wan/). MAN do not have to be in urban areas; the term "metropolitan" implies the size of the network, not the demographics of the area that it serves.



**PAN:**

A personal area network (PAN) connects electronic devices within a user's immediate area. The size of a PAN ranges from a few centimeters to a few meters. One of the most common real-world examples of a PAN is the connection between a Bluetooth earpiece and a smartphone. PAN can also connect laptops, tablets, printers, keyboards, and other computerized devices.

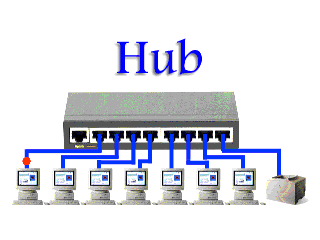


* **Networking devices:**

Different networking devices have different roles to play in a computer network. These network devices also work at different segments of a computer network performing different works. In our new series after network topology, we talk about different networking devices like a switch, router, hub, bridge etc.

**Network Hub**:

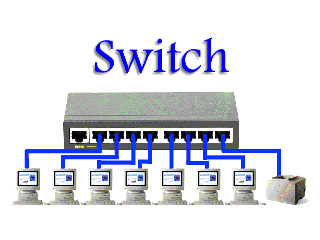
Network Hub is a networking device which is used to connect multiple network hosts. A network hub is also used to do data transfer. The data is transferred in terms of packets on a computer network. So when a host sends a data packet to a network hub, the hub copies the data packet to all of its ports connected to. Like this, all the ports know about the data and the port for whom the packet is intended, claims the packet.



However, because of its working mechanism, a hub is not so secure and safe. Moreover, copying the data packets on all the interfaces or ports makes it slower and more congested which led to the use of network switch.

**Network Switch:**

Like a hub, a switch also works at the layer of LAN (Local Area Network) but you can say that a switch is more intelligent than a hub. While hub just does the work of data forwarding, a switch does ‘filter and forwarding’ which is a more intelligent way of dealing with the data packets.



So, when a packet is received at one of the interfaces of the switch, it filters the packet and sends only to the interface of the intended receiver. For this purpose, a switch also maintains a CAM (Content Addressable Memory) table and has its own system configuration and memory. CAM table is also called as forwarding table or forwarding information base (FIB).

**Modem:**

A modem is a network device that both modulates and demodulates analog carrier signals (called sine waves) for encoding and decoding digital information for processing. Modems accomplish both of these tasks simultaneously and, for this reason, the term modem is a combination of "modulate" and "demodulate."

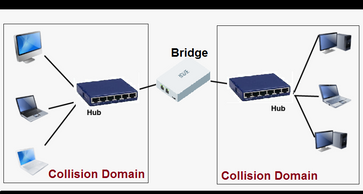


**Network Router:**

A router is a device that connects two or more packet-switched networks or sub-networks. It serves two primary functions: managing traffic between these networks by forwarding [data packets](https://www.cloudflare.com/learning/network-layer/what-is-a-packet/) to their intended [IP addresses](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/), and allowing multiple devices to use the same Internet connection.

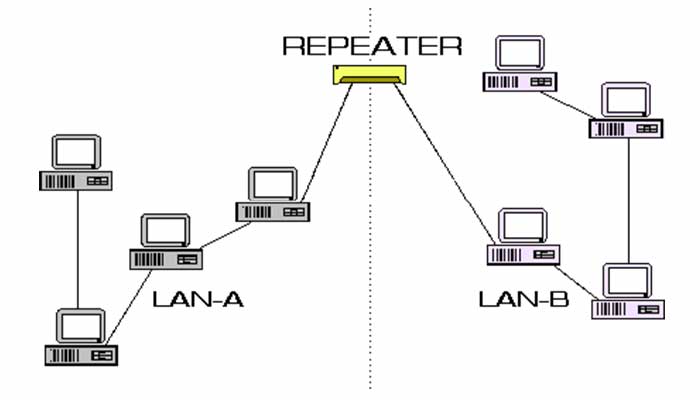
**Bridge:**

A bridge is a type of computer network device that provides interconnection with other bridge networks that use the same protocol.



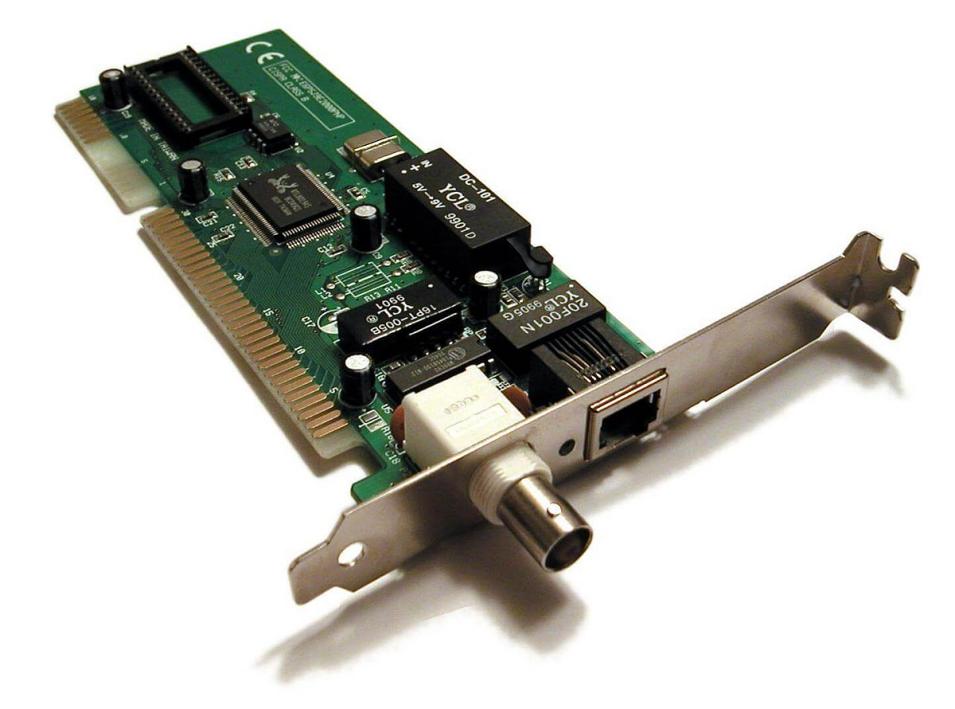
**Repeater:**

A repeater is a network device that re-transmits a received signal with more power and to an extended geographical or topological network boundary than what would be capable with the original signal.



**Network interface card (NIC):**

A NIC provides a computer with a dedicated, full-time connection to a network. It implements the physical layer circuitry necessary for communicating with a data link layer standard, such as Ethernet or Wi-Fi. Each card represents a device and can prepare, transmit and control the flow of data on the network.



**Conclusion**: Thus we have studied practically configure the networking control devices.

**Experiment No. 2**

**Title:-** Study of network connectivity test tools: ping, nslookup, ipconfig, arp, route, tracert, nmap, netstat, finger.

**Aim:-** To study Network connectivity test tools(networking commands) with all its options .

**Objective:-** To make use of Network configuration and troubleshooting commands

**Relevance:-**Networking commands

**Theory**:-

**Introduction:**

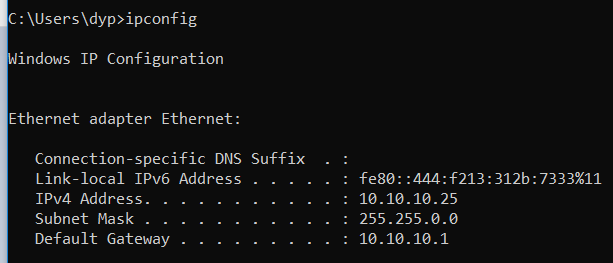
All commands related to Network configuration which includes how to switch to privilege mode and normal mode and how to configure router interface and how to save this

configuration to flash memory or permanent memory. This commands includes

**IP/Networking Commands:**

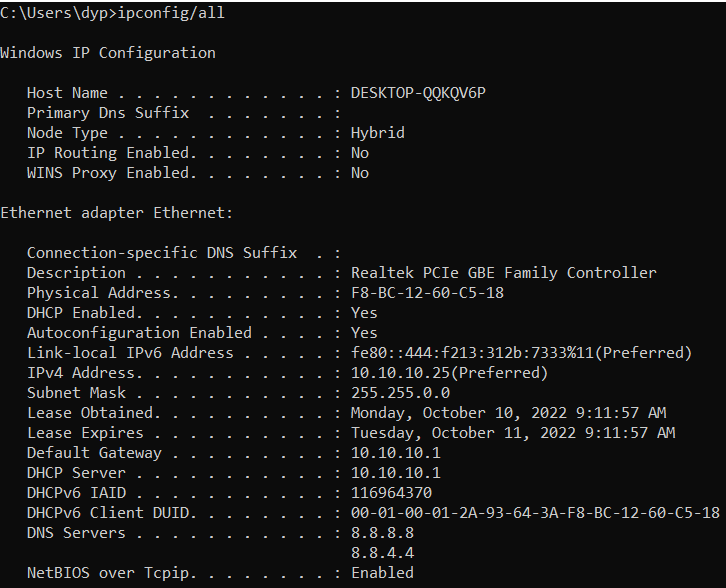
There are a lot of IP commands with short descriptions listed here but you need the ones mentioned below to diagnose and configure your network.

1. **Ipconfig:**

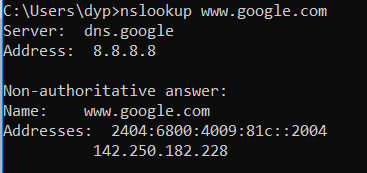
****

ipconfig (standing for "Internet Protocol configuration") is a console application program of some computer operating systems that displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings.Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, ipconfig displays the IP address, subnet mask, and default gateway for all adapters.

1. **ipconfig/all**

**** Ipconfig/all is a Windows command line which is used to find all network connection information like lan info, wireless info, vpn and so on.

1. **nslookup**

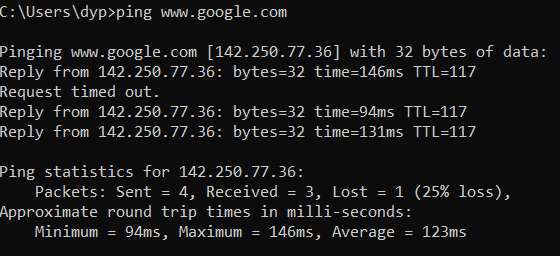
****

Nslookup sends DNS requests to a DNS server

-nslookup [domain] [dns server]

The nslookup command to send DNS requests to a server. By default, if you do not specify the DNS server, the command will use the one that is configured for your network interface (the one you use to surf the internet, for example). Nslookup is the name of a program that lets an Internet server administrator or any computer user enter a host name (for example, "whatis.com") and find out the corresponding IP address or domain name system (DNS) record. The user can also enter a command for it to do a reverse DNS lookup and find the host name for an IP address that is specified.

1. **ping :**

****

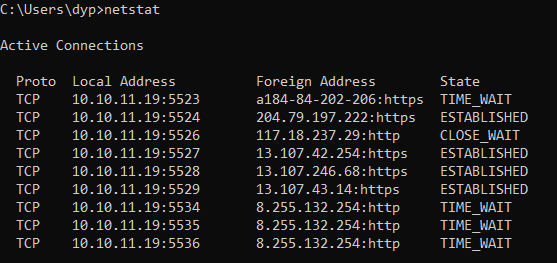
ping is the primary TCP/IP command used to troubleshoot connectivity, reachability, and name resolution. Used without parameters, this command displays Help content. You can also use this command to test both the computer name and the IP address of the computer.

1. **Tracert [www.google.com](../Documents/www.google.com)**

****

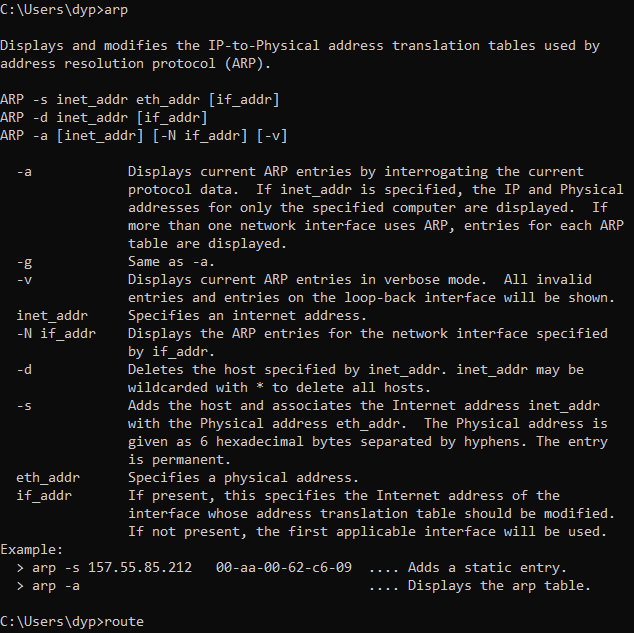
Tracertis a command-line utility that you can use to trace the path that an Internet Protocol (IP) packet takes to its destination.

1. **netstat**

****

The netstat command, meaning network statistics, is a Command Prompt command used to display very detailed information about how your computer is communicating with other computers or network devices.

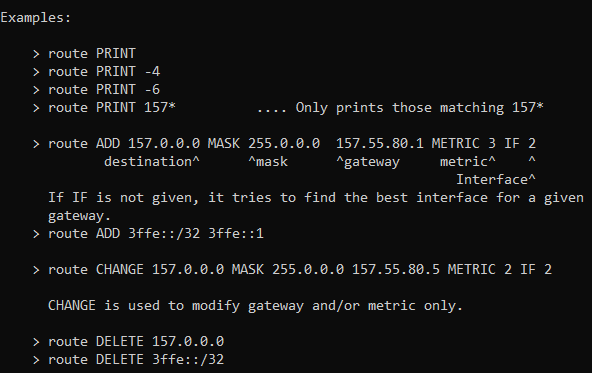
1. **arp**

****

The ARP commands to view, display, or modify the details/information in an ARP table/cache.

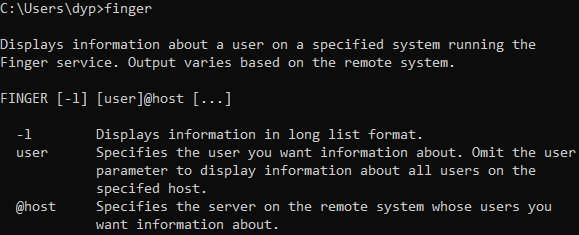
The ARP cache or table has the dynamic list of IP and MAC addresses of those devices to which your computer has communicated recently in a local network. The purpose of maintaining an ARP table is that when you want to communicate with another device, your device does not need to send the ARP request for the MAC address of that device.

1. **Route:**

****

route (command) In computing, route is a command used to view and manipulate the IP routing table in Unix-like and Microsoft Windows operating systems and also in IBM OS/2 and ReactOS. Manual manipulation of the routing table is characteristic of static routing.

1. **Finger**



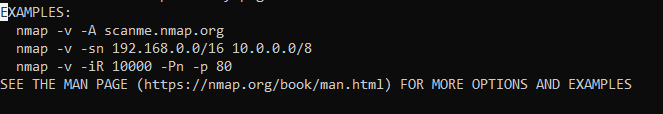
Purpose Shows user information. This command is the same as the f command.

Syntax { finger | f } [ [ -b ] [ -h] [ -l ] [ -p ]]| [ -i ] [ -q ] [ -s ] [ -w ]] ...

Description The /usr/bin/finger command displays information about the users currently logged in to a host. ...

Flags Parameters ...

1. **Nmap:**



Nmap is short for Network Mapper. It is an open-source Linux command-line tool that is used to scan IP addresses and ports in a network and to detect installed applications. Nmap (Network mapper) is an open-source Linux tool for network and security auditing. The tool helps network administrators reveal hosts and services on various systems.

Nmap works both locally and remotely. Typical uses include scanning for open ports, discovering vulnerabilities in a network, network mapping, and maintenance. The tool is valuable from both a security and networking standpoint.

**Conclusion:** Thus we studied and execute all IP commands in any network with the help of

Network configuration and troubleshooting commands.

**Experiment No. 3**

**Title**:-Framing Method : Character- count method.

**Aim**:-To study different framing methods & implementation of character count method.

**Objective**:- To see how synchronization between sender & receiver by character count method.

**Relevance**:- Synchronization between sender & receiver

**Theory**:

**Framing:-**

The physical layer accepts data as raw bits .This bit stream is not guaranteed & Error free .The number of bits received may be less than, equal to or more than the number of bits transmitted & have different values. It is up to the data link layer to detect & if necessary ,correct error. Breaking the bit stream up into frames is more difficult .One way to achieve this framing is to insert time gaps between frames like the space between words in ordinary text.

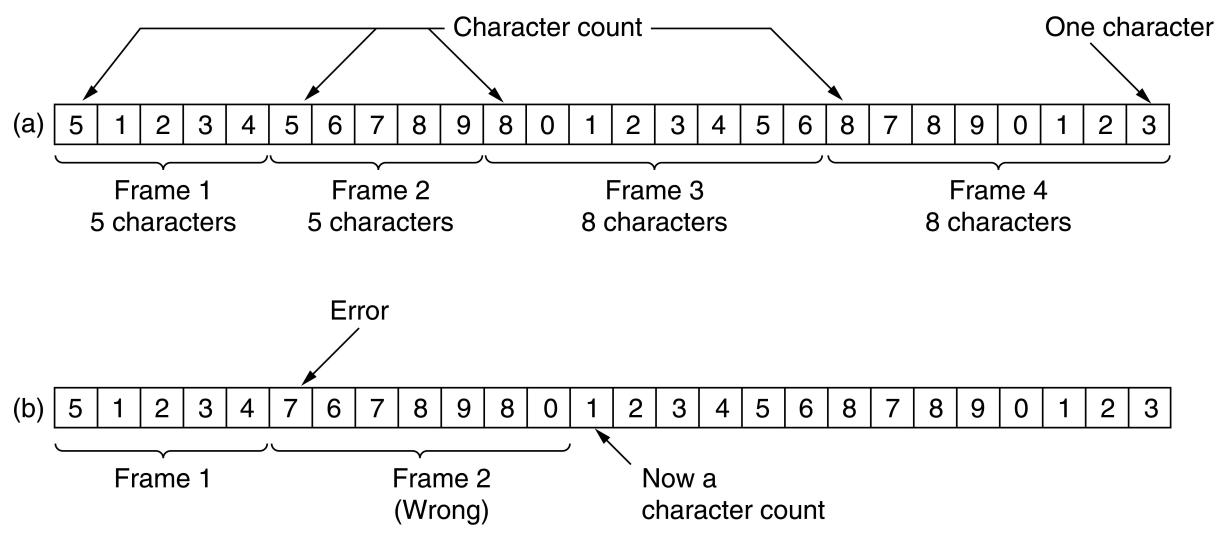
Following are the methods of framing :

* Character count.
* Starting &ending character with character stuffing.
* Starting & ending flags with bit stuffing.
* Physical layer coding violation.

1. **CHARACTER COUNT:**

It uses a field in the header that specifies the number of character in the frames . When the data link layer at the destination see the character count ,it knows how many characters follow &hence when the end of frame is as shown

Field in header gives no. of chars in frame.   
Shown in (a) below. Char count includes the counting character itself:

[](http://www.computing.dcu.ie/~humphrys/Notes/Networks/tanenbaum/3-04.jpg)

### Errors

Problem shown in (b). Transmission error changed 5 to 7. All frames now out of synch. Even if we detect error, we have no way of recovering - of finding where next frame starts.

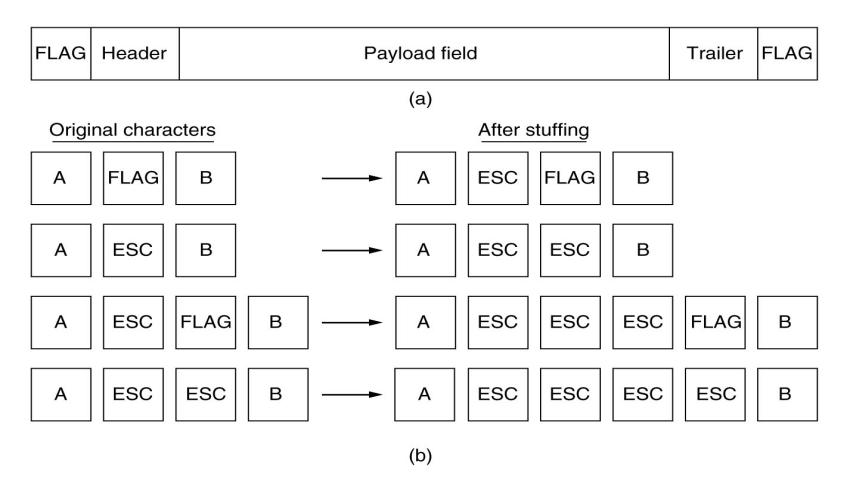
The trouble with this algorithm is that the count can be garbled by a transmission error. In this case, sending a frame back to the source asking for a retransmission doesn’t help, since the destination doesn’t know how many characters to skip over to get to the start of retransmission so character count method is rarely used.

1. **CHARACTER STUFFING:**

This framing method gets around the problem of synchronization after an error by having each frame start with ASCII character sequence DLE ,STX & END with the sequence DLE ,ETX .

* DLE: Data link escape.
* STX: Start of text.
* ETX: End of text.

Each frame starts with special start and end bytes (flag bytes). Here will imagine it as same byte, FLAG. After error, can always find start of next frame.  
See (a) below:

[](http://www.computing.dcu.ie/~humphrys/Notes/Networks/tanenbaum/3-05.jpg)

Q. What if flag byte itself is in the data?  
Probably won't happen for text data, but could easily happen with [binary data](http://www.computing.dcu.ie/~humphrys/Notes/OS/files.html" \l "binary).   
A. Insert special escape byte (ESC) before each FLAG in data. Removed at far end. This is called **byte stuffing** or **character stuffing**.

Q. What if ESC itself is in data?   
A. Insert another ESC before it.   
See (b) above. This framing method gets the problem of resynchronization .A serious problem occurs within this method when binary data such as object programs or floating point numbers are being transmitted .It may happen that the characters for DLE STX or DLE ETX occur in data ,which will interfere framing .One way to solve this problem is to have sender’s data link layer insert ASCII DLE character just for each accidental DLE character in the data .The data link layer on receiving end removes the DLE before the data are given to network layer .This technique is called character stuffing.

* **Disadvantage:**

It is closely tied to 8-bit character in general &ASCII character code in particular.

1. **BIT STUFFING:**

The new techniques allows the data frames to contain an arbitrary number of bits & allows character codes with an arbitrary number of bits per character.

Each character begins & ends with a special bit pattern 01111110 called a flag byte .When the sender’s data link layer encounter five consecutive ones in data ,it automatically stuffs a 0 bit into the outgoing bit stream. his bit stuffing is analogous to characters stuffing in which DLE is stuffed into outgoing bit stream. Bore DLE in the data.

When receiver see five consecutive incoming 1 bits , followed by a 0 bit, it automatically destuffs the 0 bit .Just as character stuffing is completely transparent to N/W layer in both computers so is bit stuffing. f the user data contain flag pattern, 01111110 this flag is transmitted an 011111010 but stored in receiver memory as

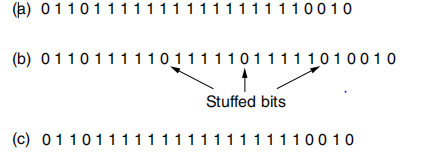


fig :*BIT STUFFING*

a)The original data

b)The data as they appear on the line

c)The data as they are stored in the receiver’s memory after destuffing.

With all stuffing ,the boundary between two frames be recognized by flag pattern .Thus if receiver looses track, all it has to do is scan the input for flag sequences ,since they can occur at frame boundaries never within data.

1. **PHYSICAL LAYER CODING VIOLATION:**

It is only applicable to networks in which the encoding on the physical medium contains some redundancy. e.g.. some LAN’s encode 1 bit of data by using 2 physical bits. Normally 1 bit is a high-low pair .The combination is high-high &low-low are not used for data .The scheme means that every data bits has transition in the middle ,making it east for the receiver to locate the bit boundaries.

#include <stdio.h>

#include <string.h>

int main()

{

char data[50],stuff[50],dstuff[50];

int i,j,fsize,noofframes,tframes,dsize,s;

printf("Enter Data:");

scanf("%s",data);

dsize=strlen(data);

printf("Enter fsize:\n");

scanf("%d",&fsize);

s=fsize;

tframes=fsize-1;

noofframes=(dsize/tframes+1);

j=0;

for(i=0;i<(dsize+noofframes);i++)

{

if(i%fsize==0)

{

if((dsize-j)<fsize)

{

fsize=dsize-j+1;

}

stuff[i]=(char)(48+fsize);

fsize=s;

}

else

{

stuff[i]=data[j];

j++;

}

} stuff[dsize+noofframes]='\0';

printf("stuffed data is: %s",stuff);

**//Destuffing:**

j=0;

for(i=0;i<(dsize+noofframes);i++)

{

if(i%fsize!=0)

{

dstuff[j]=stuff[i];

j++;

}

}

printf("\nDestuffed data is:%s",dstuff);

return 0;

}

**OUTPUT:**

**Enter Data:hello**

**Enter fsize:**

**3**

**stuffed data is: 3he3ll2o**

**Destuffed data is:hello**

**Conclusion:**

Thus we studied the different framing methods and implement Character- count framing method.

**Experiment No. 4**

**Title**:-Framing Method : bit stuffing method.

**Aim**:-To study different framing methods & implementation of bit stuffing method.

**Objective**:- To see how synchronization between sender & receiver by character count method.

**Relevance**:- synchronization between sender & receiver

**Theory**:

**Framing:-**

The physical layer accepts data as raw bits .This bit stream is not guaranteed &

Error free .The number of bits received may be less than, equal to or more than the number of bits transmitted & have different values. It is up to the data link layer to detect & if necessary ,correct error.

Breaking the bit stream up into frames is more difficult .One way to achieve this framing is to insert time gaps between frames like the space between words in ordinary text.

Following are the methods of framing :

* Character count.
* Starting &ending character with character stuffing.
* Starting & ending flags with bit stuffing.
* Physical layer coding violation.

.

**BIT STUFFING:**

The new techniques allows the data frames to contain an arbitrary number of bits & allows character codes with an arbitrary number of bits per character.

Each character begins & ends with a special bit pattern 01111110 called a flag byte .When the sender’s data link layer encounter five consecutive ones in data ,it automatically stuffs a 0 bit into the outgoing bit stream. his bit stuffing is analogous to characters stuffing in which DLE is stuffed into outgoing bit stream. Bore DLE in the data.

When receiver see five consecutive incoming 1 bits , followed by a 0 bit, it automatically de tuffs the 0 bit .Just as character stuffing is completely transparent to N/W layer in both computers so is bit stuffing. f the user data contain flag pattern, 01111110 this flag is transmitted an 011111010 but stored in receiver memory as

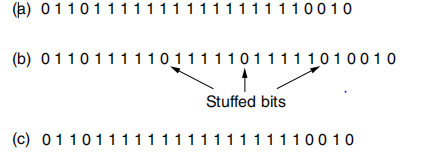


fig :*BIT STUFFING*

a)The original data

b)The data as they appear on the line

c)The data as they are stored in the receiver’s memory after destuffing.

With all stuffing ,the boundary between two frames be recognized by flag pattern .Thus if receiver looses track, all it has to do is scan the input for flag sequences ,since they can occur at frame boundaries never within data.

#include<stdio.h>

#include<string.h>

int main()

{

char data[50],stuff[50],dstuff[50];

int i,j,cnt,len;

printf("Enter the Data(in '0' and '1' form):");

scanf("%s",data);

len=strlen(data);

cnt=0;

j=0;

for(i=0;i<len;i++)

{

if(data[i]=='1')

{

cnt++;

}

else

{

cnt=0;

}

stuff[j]=data[i];

j++;

if(cnt==5 && data[i+1]=='1')

{

stuff[j]='0';

j++;

cnt=0;

}

}

stuff[j]='\0';

printf("Stuffed data with flag byte(01111110):\n 01111110 %s 01111110 \n",stuff);

**//Destuffing:**

len=strlen(stuff);

cnt=0;

j=0;

for(i=0;i<len;i++)

{

if(stuff[i]=='1')

{

cnt++;

}

else

{

cnt=0;

}

dstuff[j]=stuff[i];

j++;

if(cnt==5 && stuff[i+1]=='0')

{

cnt=0;

i++;

}

}

dstuff[j]='\0';

printf("\nDestuffed data is: %s",dstuff);

return 0;

}

**/\*output:**

**Enter the Data(in '0' and '1' form):111111**

**Stuffed data with flag byte(01111110):**

**01111110 1111101 01111110**

**Destuffed data is: 111111**

**----------------------------------------------------**

**Enter the Data(in '0' and '1' form):01111110111110**

**Stuffed data with flag byte(01111110):**

**01111110 011111010111110 01111110**

**Destuffed data is: 01111110111110**

**---------------------------------------------------**

**Conclusion:**

Thus we have studied the bit stuffing framing method.

**Experiment No. 5**

**Title :-** Error detecting & correcting code-Hamming Code

**Aim:**-Study of Hamming Code (Transmitter & receiver)

**Objective:**- To study how to achieve the integrity & accuracy in data.

**Relevance:**- Error detection & correction

**Theory:-**

**Error Correction and the Hamming Code**

The use of simple parity allows detection of single bit errors in a received message. Correction of such errors requires more information, since the position of the bad bit must be identified if it is to be corrected. (If a bad bit can be found, then it can be corrected by simply complementing its value.) Correction is not possible with one parity bit since any bit error in any position produces exactly the same information - "bad parity".

If more bits are included with a message, and if those bits can be arranged such that different errored bits produce different error results, then bad bits could be identified. In a 7-bit message, there are seven possible single bit errors, so three error control bits could potentially specify not only that an error occurred but also which bit caused the error.

Similarly, if a family of codewords is chosen such that the minimum distance between valid codewords is at least 3, then single bit error correction is possible. This distance approach is "geometric", while the above error-bit argument is 'algebraic'.

Either of the above arguments serves to introduce the **Hamming Code**, an error control method allowing correction of single bit errors.

**The Hamming Code :-**

Consider a message having four data bits (D) which is to be transmitted as a 7-bit codeword by adding three error control bits. This would be called a (7,4) code. The three bits to be added are three EVEN Parity bits (P), where the parity of each is computed on different subsets of the message bits as shown below.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| D | D | D | P | D | P | P | 7-BIT CODEWORD |
| D | - | D | - | D | - | P | (EVEN PARITY) |
| D | D | - | - | D | P | - | (EVEN PARITY) |
| D | D | D | P | - | - | - | (EVEN PARITY) |

|  |  |
| --- | --- |
| **Why Those Bits?** - The three parity bits (**1,2,4**) are related to the data bits (**3,5,6,7**) as shown at right. In this diagram, each overlapping circle corresponds to one parity bit and defines the four bits contributing to that parity computation. For example, data bit **3** contributes to parity bits **1** and **2**. Each circle (parity bit) encompasses a total of four bits, and each circle must have EVEN parity. Given four data bits, the three parity bits can easily be chosen to ensure this condition.  It can be observed that changing any one bit numbered 1..7 uniquely affects the three parity bits. Changing bit 7 affects all three parity bits, while an error in bit 6 affects only parity bits 2 and 4, and an error in a parity bit affects only that bit. The location of any single bit error is determined directly upon checking the three parity circles. | Venn |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| For example, the message 1101 would be sent as 1100110, since:   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 7-BIT CODEWORD | | 1 | - | 0 | - | 1 | - | 0 | (EVEN PARITY) | | 1 | 1 | - | - | 1 | 1 | - | (EVEN PARITY) | | 1 | 1 | 0 | 0 | - | - | - | (EVEN PARITY) |   When these seven bits are entered into the parity circles, it can be confirmed that the choice of these three parity bits ensures that the parity within each circle is EVEN, as shown here. |  |

It may now be observed that if an error occurs in any of the seven bits, that error will affect different combinations of the three parity bits depending on the bit position.

For example, suppose the above message 1100110 is sent and a single bit error occurs such that the codeword 1110110 is received:

Transmitted received

1 1 0 0 1 1 0 ------------> 1 1 1 0 1 1 0

7 6 5 4 3 2 1 7 6 5 4 3 2 1

The above error (in bit 5) can be corrected by examining which of the three parity bits was affected by the bad bit:

|  |
| --- |
|  |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 7-BIT CODEWORD |
| 1 | - | 1 | - | 1 | - | 0 | (EVEN PARITY) | **NOT!** | 1 |
| 1 | 1 | - | - | 1 | 1 | - | (EVEN PARITY) | **OK!** | 0 |
| 1 | 1 | 1 | 0 | - | - | - | (EVEN PARITY) | **NOT!** | 1 |

In fact, the bad parity bits labeled 101 point directly to the bad bit since 101 binary equals 5. Examination of the 'parity circles' confirms that any single bit error could be corrected in this way. The value of the Hamming code can be summarized:

1. Detection of 2 bit errors (assuming no correction is attempted);
2. Correction of single bit errors;
3. Cost of 3 bits added to a 4-bit message.

**Hamming Code Error detection and correction:**

#include<stdio.h>

#include<conio.h>

void main()

{

int data[7],rec[7],i,c1,c2,c3,c;

printf("Hamming code is for message of 4 bits in size \nenter 4 bit message bit one by one:");

scanf("%d%d%d%d",&data[0],&data[1],&data[2],&data[4]);

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("\nThe encoded bits are given below: \n");

for (i=0;i<7;i++)

{

printf("%d ",data[i]);

}

printf("\nEnter the received data bits one by one: ");

for (i=0;i<7;i++)

{

scanf("%d",&rec[i]);

}

c1=rec[6]^rec[4]^rec[2]^rec[0];

c2=rec[5]^rec[4]^rec[1]^rec[0];

c3=rec[3]^rec[2]^rec[1]^rec[0];

c=c3\*4+c2\*2+c1\*1 ;

if(c==0)

{

printf("\ncongratulations there is no error: ");

}

else

{

printf("\nError on the position: %d\nthe corrected bit message is at the location %d is: \n",c,c);

if(rec[7-c]==0)

rec[7-c]=1;

else

rec[7-c]=0;

{

printf("%d ",rec[c]);

}

}

printf("\nThe Corrected message which is recieved by reciever after correcting bit:\n");

for (i=0;i<7;i++)

{

printf("%d ",rec[i]);

}

getch();

}

**OUTPUT 1:**

**-----------------------------------------------------------------------------------------------**

Hamming code is for message of 4 bits in size

enter 4 bit message bit one by one:1 0 0 1

The encoded bits are given below:

1 0 0 1 1 0 0

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

congratulations there is no error:

The Corrected message which is recieved by reciever after correcting bit:

1 0 0 1 1 0 0

--------------------------------

Process exited after 21.9 seconds with return value 13

Press any key to continue . . .

**OUTPUT 2:**

**-----------------------------------------------------------------------------------------------**

Hamming code is for message of 4 bits in size

enter 4 bit message bit one by one:1 0 0 1

The encoded bits are given below:

1 0 0 1 1 0 0

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

Error on the position: 5

the corrected bit message is at the location 5 is:

0

The Corrected message which is recieved by reciever after correcting bit:

1 0 0 1 1 0 0

--------------------------------

Process exited after 18.78 seconds with return value 13

Press any key to continue . . .

**Conclusion:**

Thus we have studied the Hamming code technique of error detection and error correction to achieve the integrity & accuracy in data.

**Experiment No. 6**

**Title :-** Cyclic Redundancy code (CRC) Error Detection method

**Aim:**-Study & Implementation of the cyclic redundancy method(Transmitter & receiver)

**Objective:-** To study how to achieve the integrity & accuracy in data.

**Relevance:**- Error detection

**Theory:-**

**Cyclic Redundancy Method:-**

It is also called as polynomial code .polynomial code are based upon treating bit string as representation of polynomials with co- efficiency of polynomial with co- efficiency of only a k-bit frame is regarded as co-efficiency .for a polynomial with k terms ranging from a x k-1 to x 0 such polynomial is said to be of degree k-1.the length order bit is the coefficient x k-1.the next bit is the coefficient of x k-2 &so on.

Polynomial arithmetic is done by module 2 according to rules of algebraic field theory these both addition , subtraction are identical to EX-OR

e.g

10011001 11110000

- 11001010 + 10100111

--------------- ------------------

01010011 01010111

when polynomial code method is employed the sender & receiver must agree upon a generator polynomial of G(x) .Both low & high order bit of G(x) is 1.

To compute checksum for some frame with m bits, corresponding to the polynomial M(x) frame must be longer than generator polynomial. Idea is to append checksum to end of the frame in such a way that the

polynomials represent by check summed frame is divisible by G(x).when the receiver gets the check summed frame ,it tries dividing it by G(x).if there is no remainder ,there has been a transmission error .

The algorithm for computing the checksum:

1. Let ‘r’ be the degree of g(x).Append ‘r’ zero bit to the low order end of frame no ,so it now contain m + r bit & correspond to polynomial x r M(x).
2. Divide the bit string corresponding to G(x) into the bit string corresponding to x r M(x) using module 2 division.
3. Subtract the remainder from the bit string corresponding to x r M(x) using module 2 subtraction . the result is the check summed frame to be transmitted .call it as polynomial T(x)

**Example:-**

**Frame:-** 1101011011

**Genrator:-**10011

**Message after appending zero’s:-**11010110110000

1100001010

10011 11010110110000

10011

10011

10011

00001

00000

00010

00000

00101

00000

01011

00000

10110

10011

01010

00000

10100

10011

01110

00000

1110 (Remainder)

Transmitted Frame:- 11010110111110

**Program of Cyclic Redundency Code (CRC):**

#include <stdio.h>

#include <conio.h>

#include <string.h>

void main()

{

int i,j,keylen,msglen;

char input[100], key[30],temp[30],quot[100],rem[30],key1[30];

//clrscr();

printf("Enter Data: ");

gets(input);

printf("Enter Key: ");

gets(key);

keylen=strlen(key);

msglen=strlen(input);

strcpy(key1,key);

for (i=0;i<keylen-1;i++)

{

input[msglen+i]='0';

}

for (i=0;i<keylen;i++)

temp[i]=input[i];

for (i=0;i<msglen;i++)

{

quot[i]=temp[0];

if(quot[i]=='0')

for (j=0;j<keylen;j++)

key[j]='0';

else

for (j=0;j<keylen;j++)

key[j]=key1[j];

for (j=keylen-1;j>0;j--)

{

if(temp[j]==key[j])

rem[j-1]='0'; else

rem[j-1]='1';

}

rem[keylen-1]=input[i+keylen];

strcpy(temp,rem);

}

strcpy(rem,temp);

printf("\nQuotient is ");

for (i=0;i<msglen;i++)

printf("%c",quot[i]);

printf("\nRemainder is ");

for (i=0;i<keylen-1;i++)

printf("%c",rem[i]);

printf("\nFinal data is: ");

for (i=0;i<msglen;i++)

printf("%c",input[i]);

for (i=0;i<keylen-1;i++)

printf("%c",rem[i]);

getch();

}

**OUTPU 1:**

**-----------------------------------------------------------------------------**

Enter Data: 1101011011

Enter Key: 10011

Quotient is: 1100001010

Remainder is (CRC): 1110

Final data is transmitted after appending CRC to original message: 11010110111110

--------------------------------

Process exited after 124.1 seconds with return value 13

Press any key to continue . . .

**OUTPU 2:**

**-----------------------------------------------------------------------------**

Enter Data: 100100

Enter Key: 1101

Quotient is: 111101

Remainder is (CRC): 001

Final data is transmitted after appending CRC to original message: 100100001

--------------------------------

Process exited after 24.16 seconds with return value 13

Press any key to continue . . .

**Conclusion:**

Thus we studied cyclic redundancy code (CRC) method for error detection which is used to achieve the integrity & accuracy in data.

**Experiment No. 7**

**Title**:-Sliding Window Protocol.

**Aim**:-Study & Implementation of Sliding Window Protocol.

**Objective**:- To see how synchronization between sender & receiver by Sliding Window Protocol.

**Relevance**:- synchronization between sender & receiver

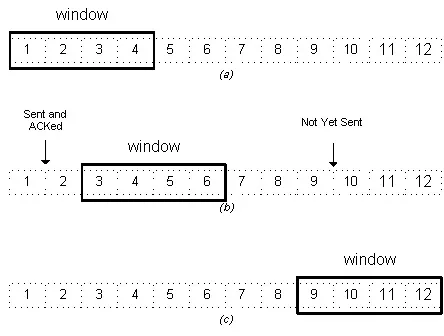
**Theory**:

**Sliding Window Protocol:**

In computer networks sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agrees on some window size. If window size=w then after sending w frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.



In sliding window protocol the receiver has to have some memory to compensate any loss in transmission or if the frames are received unordered.

**Efficiency of Sliding Window Protocol:**

η = (W\*tx)/(tx+2tp)

W = Window Size

tx = Transmission time

tp = Propagation delay

Sliding window works in full duplex mode

It is of two types:

1. **Selective Repeat:** Sender transmits only that frame which is erroneous or is lost.

2. **Go back n**: Sender transmits all frames present in the window that occurs after the error bit including error bit also.

**PROGRAM OF SLIDING WINDOW PROTOCOL:**

#include<stdio.h>

int main()

{

int w,i,f,frames[50];

printf("Enter window size(w): ");

scanf("%d",&w);

printf("\nEnter number of frames to transmit(f): ");

scanf("%d",&f);

printf("\nEnter %d frames: ",f);

for(i=1;i<=f;i++)

scanf("%d",&frames[i]);

printf("\nWith sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)\n\n");

printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);

for(i=1;i<=f;i++)

{

if(i%w==0)

{

printf("%d\n",frames[i]);

printf("Acknowledgement of above frames sent is received by sender\n\n");

}

else

printf("%d\n ",frames[i]);

}

if(f%w!=0)

printf("\nAcknowledgement of above frames sent is received by sender\n");

return 0;

}

**OUTPU 1:**

**-----------------------------------------------------------------------------**

Enter window size: 3

Enter number of frames to transmit: 5

Enter 5 frames: 12 5 89 4 6

With sliding window protocol the frames will be sent in the following manner (assuming no corruption of frames)

After sending 3 frames at each stage sender waits for acknowledgement sent by the receiver

12

5

89

Acknowledgement of above frames sent is received by sender

4

6

Acknowledgement of above frames sent is received by sender \*/

**Conclusion:**

Thus we studied sliding window protocol method for data transmission through network which is used to achieve the synchronization between sender and receiver.

**Experiment No. 8**

**Title:-** Shortest Path Routing Algorithm

**Aim:-** Study & Implementation of the Shortest Path Routing

**Objective:-** To find out shortest path from subnet by dijkstra algo.

**Relevance:-**  For finding the shortest path.

**Theory:-**

**Routing Algorithm:-**

The main function of the Network layer is to route the packet from the source machine to destination machine. In most subnet, packet will require multiple hopes to make the journey. Routing is an issue if the source & destination are not on the same network. The algorithm that choose the route & the Data structure that they use are a major area of n/w layer design.

The routing algo. Is that is part of n/w layer s/w responsible for deciding which o/p line incoming packet should be transmitted on If the subnet uses datagram internally, this decision must changed since last time .If subnet uses virtual circuit internally ,routing decision are only when a new virtual ckt is being set up.

Thereafter ,data packet just follow the previously established route. The latter case is sometime called session routing ,because a route remain in force for an entire user session .Properties that are desirable in a routing algorithm\_ correctness, simplicity robustness, stability, fairness, optimality.

**Static Vs Dynamic routing:-**

Routing algorithm can be grouped into major classes:

Non-adaptive & adaptive .

Non-adaptive algorithm don’t base their routing decision on measurement or estimates of the current traffic & topology.

Instead ,the choice of the rout to use to get from I to J computed in advanced, off\_line ,& downloaded to the router when the n/w is booted. This procedure is called **static routing.**

Adaptive algorithm, in contrast change their routing decision to effect changes in the topology,& usually the traffic as well.

Adaptive algorithm differ in where they get their information .When they change the routs & What metric is used for optimization .This procedure is called as **Dynamic Routing**.

**Shortest Path Routing:-**

The idea is to built a graph of the subnet, with each node of the graph representing a router & each of the graph representing a communication line. To choose a rout between a given pair of routers ,the algo. just finds shortest path between them on the graph.

On way of measuring path length is number of hops .Using the metric , Path ABC & ABE in fig are equally long. Another metric is a geographic distance in km ,in which case ABC is clearly much longer than ABE. The first 5 step used in computing the shortest path from A to D .The arrows indicate working nodes

However ,other metric are also possible besides hopes & physical distance. By changing ,the weighting function ,the algo could then compute the “shortest path “measured according to any one of a number of criteria or combination of criteria Dijkstra algo. For computing shortest path between two nodes of graphs . Each node is labeled with its distance from the source node alog the best known path .Initially no paths are known ,so all nodes are labeled with infinity. As the algo. processed 7 paths are known , found labels may change ,reflecting better paths .A label may be either tentative ,of permanent. Initially all labels are tentative when it is discovered that a label represent the shortest path from the source to that node ,it is made permanent & never changed . Look at the weighted undirected graph in above fig, where the weights represent ,.g, distance. We want to find shortest path from A to D . We start out by a marking node A as permanent ,indicated by a field in circle. Then we examine in turn ,each of the nodes adjacent to A ,relabeling each one with distance to A. whenever a node is relabeled ,we also label it with the node from which the probe was made so we can construct the final path latter.

Having examined each of nodes adjacent to A, We examine all the tentatively labels nodes in the whole graph & make the one with smallest label permanent as shown in fig.b. This one becomes new working node .

We now start at B ,& examine all node adjacent to it .if the sum of label on B & the distance from B to node being considered is less than the label on that node ,we have a shortest path ,so the node is relabeled .after all the nodes adjacent to working node have been inspected & the tentative label changed possible, the entire graph is searched for the tentatively labeled nodes with the smallest value. This node is made permanent & become the working node for the next round. fig shows the next five steps.

**Algorithm:-**

1. start
2. ask the user to enter how many node in subnet.
3. Take distance of node with itself is 0
4. Scan the distance of each node with every other node
5. Take source & destination.
6. Start from source & check the distance with it’s neighbors.
7. The neighbor that has less distance ,make it as permanent.
8. Repeat the same procedure till destination is come.
9. Store all path
10. Display that path as shortest.

**Program:-**

#include<stdio.h>

void main()

{

int a[9][9],b[9][9],min,i,j,k=0,p,c,sum=0,m[9],n=1,s,r,l1,l,h;

char ch[10]={'A','B','C','D','E','F','G','H'};

printf("Enter no of nodes:");

scanf("%d",&h);

printf("Enter path matrix:");

for(i=0;i<h;i++)

{

for(j=0;j<h;j++)

scanf("%d",&b[i][j]);

}

printf("Path Matrix:\n");

for(i=0;i<h;i++)

{

printf("\n");

for(j=0;j<h;j++)

printf(" %d",b[i][j]);

}

j=0;

m[0]=0;

while(k!=3)

{

min=500;

l=-1;

l1=0;

for(i=0;i<4;i++)

{

if((b[k][i]!=0)&&(b[k][i]!=100))

{

if(i==3)

{

min=b[k][i];

p=i;

break;

}

else

{

c=b[k][i];

if(c<min)

min=c;

if(min==c)

{

if(l1!=c)

b[k][l]=100;

l=i;

l1=c;

p=i;

b[i][k]=100;

}

else

{

b[k][i]=100;

b[i][k]=100;

}

}

}

}

m[n]=p;

n++;

k=p;

sum=sum+min;

}

printf("\nANS: Minimum Distance= %d",sum);

printf("\nPath: ");

for(i=0;i<n;i++)

printf(" %c",ch[m[i]]);

}

/\*OUTPUT:

Enter no of nodes:4

Enter path matrix:0 3 2 0 3 0 1 4 2 1 0 3 0 4 3 0

Path Matrix:

0 3 2 0

3 0 1 4

2 1 0 3

0 4 3 0

ANS: Minimum Distance= 5

Path: A C D\*/

**Conclusion:-**

Thus we studied how to find out the shortest path between source and destination node by using dijkstra’s routing algorithm.

**Experiment No 9**

**Title :** Installation of Cisco Packet tracer tool

**Aim:** To study and Installation of Cisco Packet tracer tool

**Objective :** To understand Installation of Cisco Packet tracer tool

**Relevance :** Cisco Packet tracer tool

**Theory :**

**Installation of Cisco Packet tracer tool**

Packet Tracer is a very useful Cisco network simulation tool which allows network administrators and students to experiment with cisco network device behaviour. Packet Tracer provides simulation and visualization capabilities which facilitates the lab testing and learning of complex scenarios and concepts.

Packet Tracer does not require physical equipment. It creates a virtual network with an almost unlimited number of devices, encouraging practice, design scenarios testing and troubleshooting. Packet Tracer Download Link – Link Below is step by step procedure on installing Packet tracer on Windows PC –

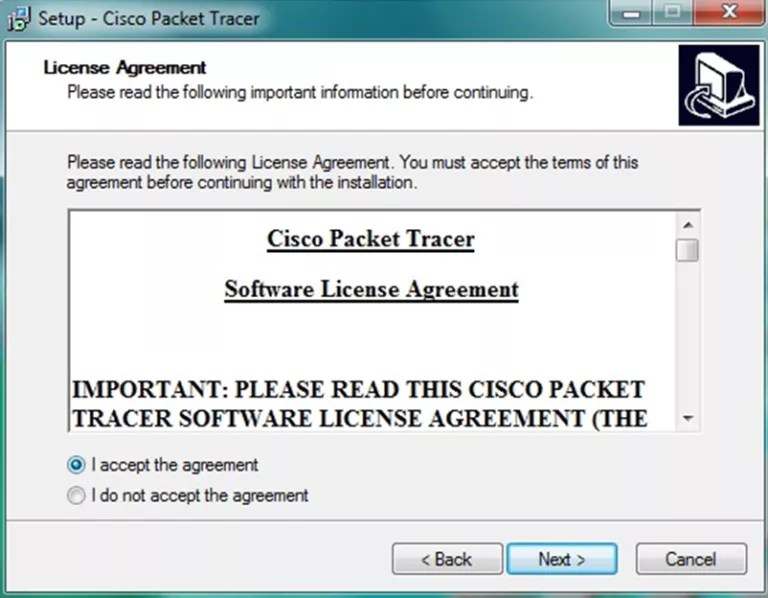
**STEP 1 –**

Once we have downloading the Packet Tracer exe file , Click on the exe file downloaded. Once below Window will appears, click next.



**STEP 2 –**

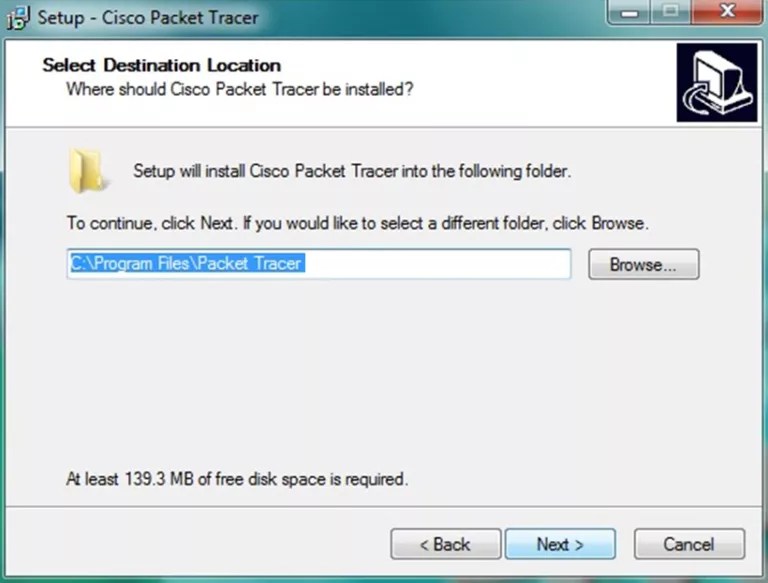
On the next screen, select “I accept the agreement” and click on “Next”.

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-02/)

Step 2 of install packet tracer

****STEP 3 –****

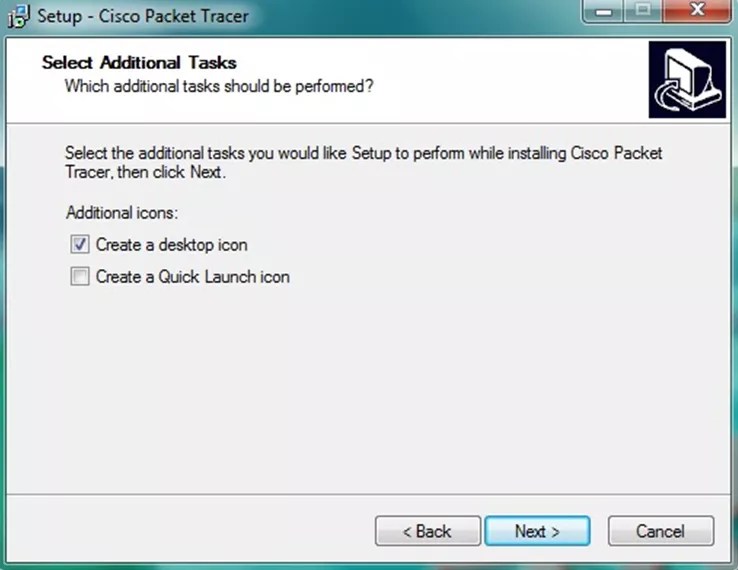
Setup will show the folder in which the program’s shortcuts will be created. If you want to change the folder, you can change it. Click on “Next”.

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-03/)

Step 3 of installing packet tracer

**STEP 4 –**

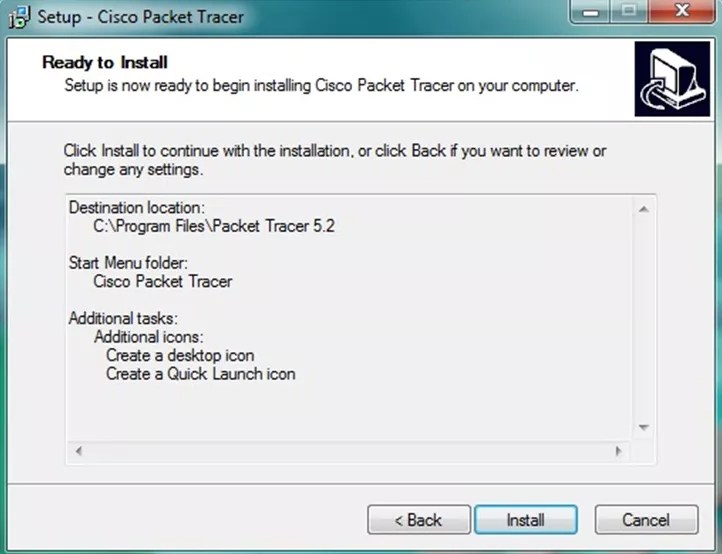
Then the program will ask whether to create a Desktop icon and create a Quick Launch icon. Make your choice and click on “Next”.

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-04/)

Step 4 of installing packet tracer

**STEP 5 –**

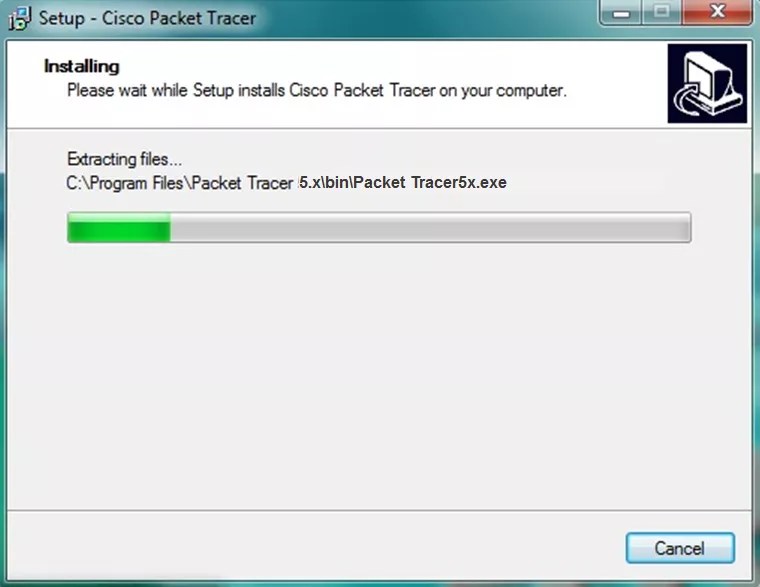
The summary of the settings we selected is displayed. Click on *“Install”*

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-05/)

Step 5 of installing packet tracer

.

**STEP 6 –**The cisco packet tracer installation starts as shown below.

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-06/)

**STEP 7 –**

Cisco packet tracer Installation gets completed and the below screen is shown. Click on “Finish”.

Click “OK” on next popup asking you to close or restart your computer.

[](http://ipwithease.com/how-to-install-packet-tracer-on-windows-system/171how-to-install-packet-tracer-on-windows-system-07/)

**STEP 8 –**

Packet Tracer is installed and ready to be used.

**Conclusion:** Thus we have studied practically installation of packet tracer tool.

**Experiment No 10**

**Title :** Implementation of Classful and classless(CIDR) Subnetting by using Cisco Packet Tracer tool and classful addressing by using program.

**Aim:** To study Classful and classless(CIDR) subnetting of Cisco Packet tracer tool.

**Objective :** To understand Classful and classless(CIDR) subnetting using Cisco Packet tracer tool

**Relevance :** Classful and classless(CIDR) subnetting Cisco Packet tracer tool

**Theory :**

The subnet, or subnetwork, is a part of a larger network. Subnets are a logical part of an IP network into multiple, smaller network components. The Internet Protocol (IP) is the method for transmitting data from one computer to another over the internet network. Each computer, or host, on the internet, has at least one IP address as a unique identifier.

### **Steps to Configure and Verify Three Router Connections in Cisco Packet Tracer:**

**Step 1:** First, open the Cisco packet tracer desktop and select the devices given below:

| **S.NO** | **Device** | **Model-Name** | **Qty.** |
| --- | --- | --- | --- |
| **1.** | PC | pc | 6 |
| **2.** | Switch | PT-Switch | 3 |
| **3.** | Router | PT-Router | 3 |

**IP Addressing Table for PCs:**

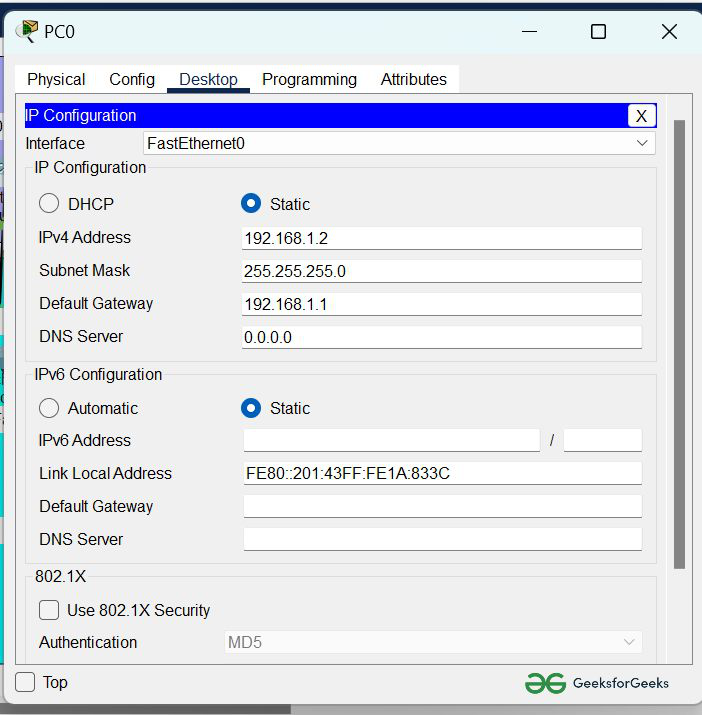
| **S.NO** | **Device** | **IPv4 Address** | **Subnet Mask** | **Default-Gateway** |
| --- | --- | --- | --- | --- |
| **1.** | pc0 | 192.168.1.2 | 255.255.255.0 | 192.168.1.1 |
| **2.** | pc1 | 192.168.1.3 | 255.255.255.0 | 192.168.1.1 |
| **3.** | pc2 | 192.168.2.2 | 255.255.255.0 | 192.168.2.1 |
| **4.** | pc3 | 192.168.2.3 | 255.255.255.0 | 192.168.2.1 |
| **5.** | pc4 | 192.168.3.2 | 255.255.255.0 | 192.168.3.1 |
| **6.** | pc5 | 192.168.3.3 | 255.255.255.0 | 192.168.3.1 |

* Then, create a network topology as shown below the image.
* Use an Automatic connecting cable to connect the devices with other



**Step 2**: Configure the PCs (hosts) with IPv4 address and Subnet Mask according to the IP addressing table given above.

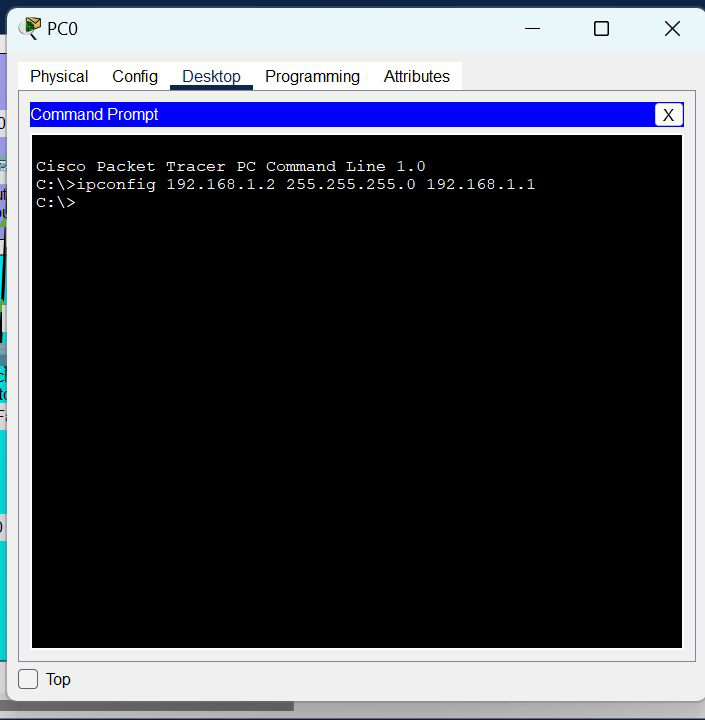
* To assign an IP address in PC0, click on PC0.
* Then, go to desktop and then IP configuration and there you will IPv4 configuration.
* Fill IPv4 address and subnet mask.



Assigning IP address using the ipconfig command.

* Or we can also assign an IP address with the help of a command.
* Go to the command terminal of the PC.
* Then, type ipconfig <IPv4 address><subnet mask><default gateway>(if needed)

Example: ipconfig 192.168.1.2 255.255.255.0 192.168.1.1



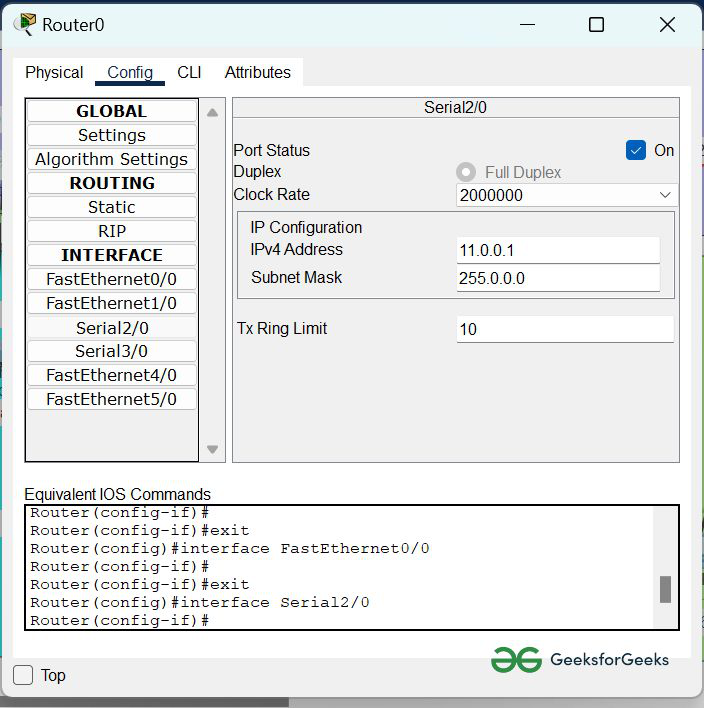
* Repeat the same procedure with other PCs to configure them thoroughly.

**Step 3:** Configure router with IP address and subnet mask.

**IP Addressing Table Router**

| **S.NO** | **Device** | **Interface** | **IPv4 Address** | **Subnet mask** |
| --- | --- | --- | --- | --- |
| **1.** | router0 | FastEthernet0/0 | 192.168.1.1 | 255.255.255.0 |
| Serial2/0 | 11.0.0.1 | 255.0.0.0 |
| **2.** | router1 | Serial 2/0 | 11.0.0.2 | 255.0.0.0 |
| Serial 3/0 | 12.0.0.1 | 255.0.0.0 |
| **3.** | router 3 | FastEthernet0/0 | 192.168.3.1 | 255.255.255.0 |
| Serial2/0 | 12.0.0.2 | 255.0.0.0 |

* To assign an IP address in router0, click on router0.
* Then, go to config and then Interfaces.
* Then, configure the IP address in FastEthernet and serial ports according to IP addressing Table.
* Fill IPv4 address and subnet mask.



* Repeat the same procedure with other routers to configure them thoroughly.

**Step 4:**After configuring all of the devices we need to assign the routes to the routers.

To assign static routes to the particular router:

* First, click on router0 then Go to CLI.
* Then type the commands and IP information given below.

CLI command : ip route <network id> <subnet mask><next hop>

**Static Routes for Router0 are given below:**

Router(config)#ip route 192.168.2.0 255.255.255.0 11.0.0.2

Router(config)#ip route 11.0.0.0 255.0.0.0 11.0.0.2

Router(config)#ip route 192.168.3.0 255.255.255.0 11.0.0.2

Router(config)#ip route 12.0.0.0 255.0.0.0 11.0.0.2

**Static Routes for Router1 are given below:**

Router(config)#ip route 192.168.1.0 255.255.255.0 11.0.0.1

Router(config)#ip route 11.0.0.0 255.0.0.0 11.0.0.1

Router(config)#ip route 192.168.3.0 255.255.255.0 12.0.0.2

Router(config)#ip route 12.0.0.0 255.0.0.0 12.0.0.2

**Static Routes for Router2 are given below:**

Router(config)#ip route 192.168.1.0 255.255.255.0 12.0.0.1

Router(config)#ip route 11.0.0.0 255.0.0.0 12.0.0.1

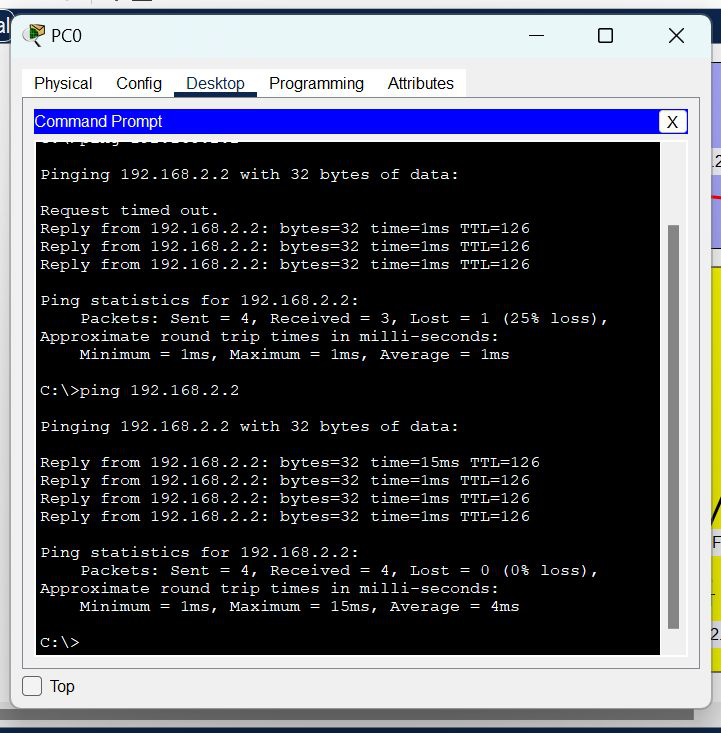
Router(config)#ip route 12.0.0.0 255.0.0.0 12.0.0.1

Router(config)#ip route 192.168.2.0 255.255.255.0 12.0.0.1

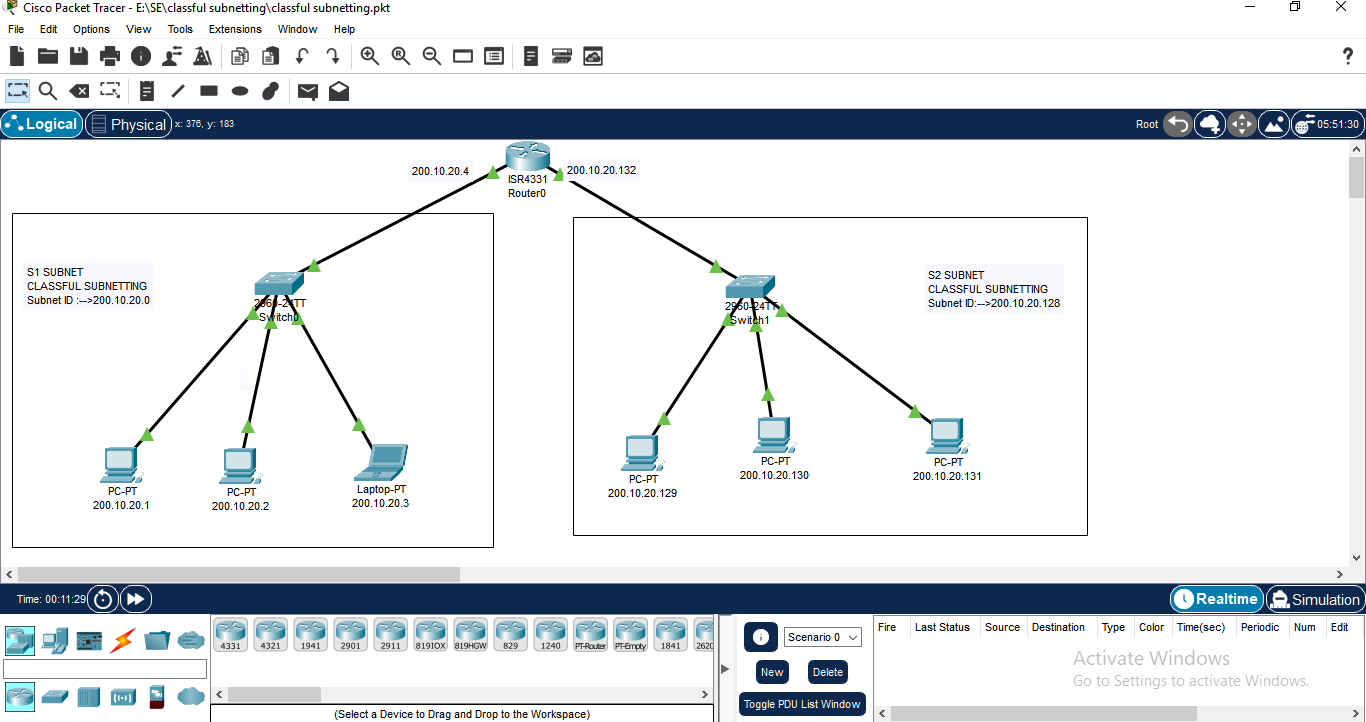
**Step 5:**Verifying the network by pinging the IP address of any PC. We will use the ping command to do so.

* First, click on PC0 then Go to the command prompt
* Then type ping <IP address of targeted node>
* As we can see in the below image we are getting replies which means the connection is working very fine

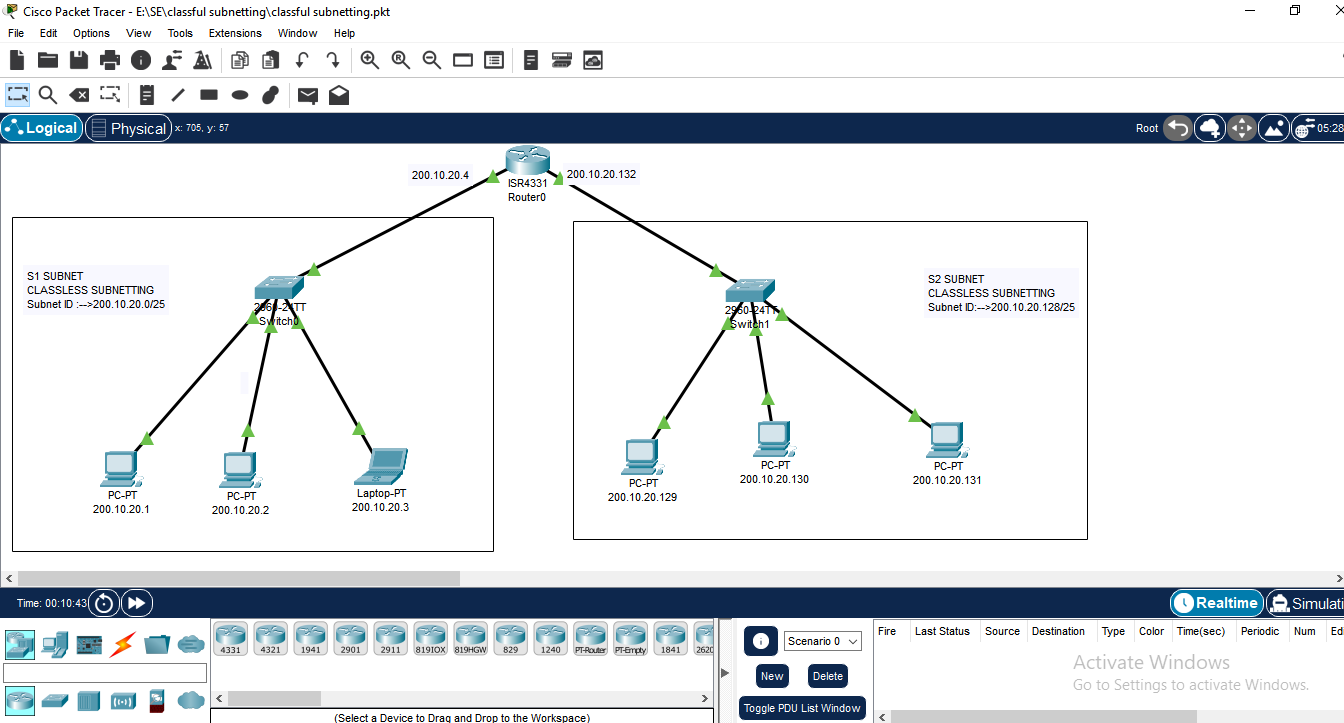
**Example** : ping 192.168.2.2



* **Classful subnetting:**



* **Classless(CIDR) subnetting:**



**/\*Program to understand and Implementation of Dotted Decimal IP address classful addressing.\*/**

#include<stdio.h>

int main()

{

int a,b,c,d;

printf("\nEnter the first byte");

scanf("%d",&a);

printf("\nEnter the secod byte");

scanf("%d",&b);

printf("\nEnter third byte");

scanf("%d",&c);

printf("\nEnter the fourth byte");

scanf("%d",&d);

if (a<=255 && b<=255 && c<=255 && d<=255)

{

printf("\nThe IP address is %d.%d.%d.%d",a,b,c,d);

}

else

{

printf("\n Invalid IP \n");

return 0;

}

if(a<=127)

{

printf("\nClass A");

}

else if(a>127&&a<=191)

{

printf("\nClass B");

}

else if(a>191&&a<=223)

{

printf("\nClass C");

}

else if(a>223&&a<=239)

{

printf("\nClass D");

}

else

{

printf("\nClass E");

}

}

/**\* Output 1:**

Enter the first byte 196

Enter the secod byte 58

Enter third byte 67

Enter the fourth byte 21

The IP address is 196.58.67.21

Class C

**Output 2:**

Enter the first byte 21

Enter the secod byte 56

Enter third byte 87

Enter the fourth byte 23

The IP address is 21.56.87.23

Class A

--------------------------------

Process exited after 14.41 seconds with return value 0

Press any key to continue . . .

\*/

**Conclusion:** Thus we have studied practically subnetting (Classful and Classless subnetting) by using Cisco packet tracer tool and classful addressing by using program.

**Experiment No 11**

**Title :** Installation of network analyzer tool Wireshark

**Aim :** To Study of Installation of network analyzer tool Wireshark

**Objective :** To understand Installation of network analyser tool Wireshark

**Relevance :** Network analyzer tool.

**Theory :**

**In this lab we will learn about**

* Introduction to Wireshark and network traffic analysis
* Introduction to Wireshark
* Installation of Wireshark and basic usage
* Network traffic analysis of HTTP, FTP protocol

**What is Wireshark?**

Wireshark is a network protocol analyzer, also known as a network sniffer. Formerly known as Ethereal, wireshark is computer application that captures and decodes packets of information from a network. “Wireshark can capture live network traffic or read data from a file and translate the data to be presented in a format the user can understand”.

**Why Wireshark:**

Wireshark is a valuable tool for administrators that allow them to monitor all traffic that passes on a network. It is very useful for analyzing, diagnosing and troubleshooting problems that may occur.

**Some features of wireshark.**

1. Data can be captured from a network connection or read from previous records of

captured packets.

2. Live data can be read from Ethernet, FDDI, PPP, token ring, IEEE 802.11, classical IP over ATM, and loopback interfaces (at least on some platforms; not all of those types are supported on all platforms ).

3. Captured files can be programmatically edited or converted via command-line

switches to the “editcap” program.

4. Captured network data can be browsed via a GUI, or via the terminal((command line) version of the utility tshark.

5. Display filters can also be used to selectively highlight and color packet summay information.

6. Data display can be refined using a display filter Hundreds of protocols can be

dissected.

**How to get Wireshark**

Wireshark can be downloaded from website www.wireshark.org



**Getting started with wireshark:**

*  Wireshark has a friendly graphical user interface that makes it easier for

the to analyze and diagnose packets that passing through the network.

No data will initially be displayed when the user runs wireshark.

*  The environment and usage of wireshark will be explained

further in this document.

*  To start capturing packets you need to select the interface which

is connected to the internet. This can be done by choosing

Capture>>Interfaces from the Menu bar

Wireshark is a free application that allows you to capture and view the data traveling back and forth on your network, providing the ability to drill down and read the contents of each packet – filtered to meet your specific needs. It is commonly utilized to troubleshoot network problems as well as to develop and test software. This open-source protocol analyzer is widely accepted as the industry standard, winning its fair share of awards over the years Originally known as Ethereal, Wireshark features a user-friendly interface that can display data from hundreds of different protocols on all major network types. These data packets can be viewed in real-time or analyzed offline, with dozens of capture/trace file formats supported including CAP and ERF. Integrated decryption tools allow you to view encrypted packets for several popular protocols such as WEPand WPA/WPA2.

1. **Downloading and Installing Wireshark**



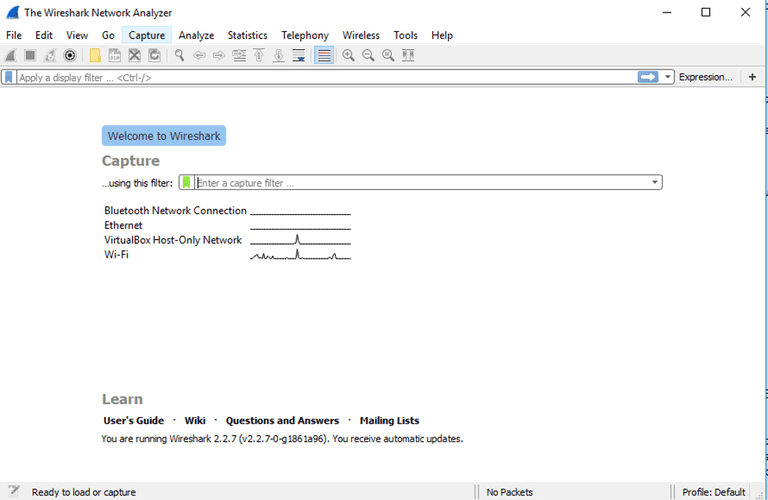
Wireshark can be downloaded at no cost from the Wireshark Foundation website for

both macOS and Windows operating systems. Unless you are an advanced user, it is

recommended that you only download the latest stable release. During the setup process

(Windows only) you should choose to also install WinPcap if prompted, as it includes a library required for live data capture. The application is also available for Linux and most other UNIX-like platforms including Red Hat, Solaris, and FreeBSD. The binaries required for these operating systems can be found towards the bottom of the download page in the Third-Party Packages section.

**2. How to Capture Data Packets**



When you first launch Wireshark a welcome screen similar to the one shown above

should be visible, containing a list of available network connections on your current device. In this example, you'll notice that the following connection types are shown: *Bluetooth Network Connection*, *Ethernet*, *VirtualBox Host-Only Network*, *Wi-Fi*. Displayed to the right of each is an EKG-style line graph that represents live traffic on that respective network. To begin capturing packets, first select one or more of these networks by clicking on your choice(s) and using the *Shift* or *Ctrl* keys if you'd like to record data from multiple networks simultaneously. Once a connection type is selected for capturing purposes, its background will be shaded in either blue or gray. Click on *Capture*from the main menu, located towards the top of the Wireshark interface. When the drop-down menu appears, select the *Start* option.

You can also initiate packet capturing via one of the following shortcuts.

**Keyboard:** Press *Ctrl + E*

 **Mouse:** To begin capturing packets from one particular network, simply double-click on

its name

 **Toolbar:** Click on the blue shark fin button, located on the far left-hand side of the

Wireshark toolbar

The live capture process will now begin, with packet details displayed in the Wireshark window

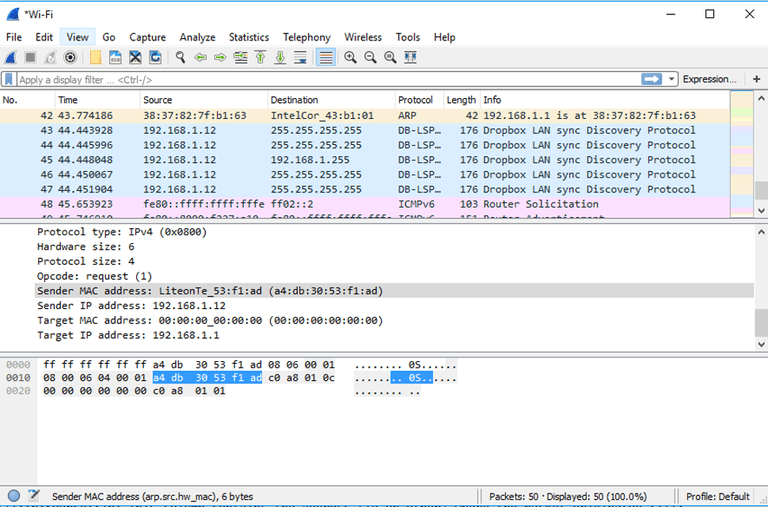
as they are recorded. Perform one of the actions below to stop capturing.

 **Keyboard:** Press *Ctrl + E*

 **Toolbar:** Click on the red stop button, located next to the shark fin on the Wireshark

toolbar

**3. Viewing and Analyzing Packet Contents**



Now that you've recorded some network data it's time to take a look at the captured packets. As shown in the screenshot above, the captured data interface contains three main sections: The packet list pane, the packet details pane, and the packet bytes pane.

* **Packet List**

The packet list pane, located at the top of the window, shows all packets found in the active

capture file. Each packet has its own row and corresponding number assigned to it, along with each of these data points.

* **Time:** The timestamp of when the packet was captured is displayed in this column, with the default format being the number of seconds (or partial seconds) since this specific capture file was first created. To modify this format to something that may be a bit more useful, such as the actual time of day, select the *Time Display Format* option from Wireshark's *View* menu - located at the top of the main interface.

 **Source:** This column contains the address (IP or other) where the packet originated.

 **Destination:** This column contains the address that the packet is being sent to.

 **Protocol:** The packet's protocol name (i.e., TCP) can be found in this column.

 **Length:** The packet length, in bytes, is displayed in this column.

 **Info:** Additional details about the packet are presented here. The contents of this column

can vary greatly depending on packet contents.

When a packet is selected in the top pane, you may notice one or more symbols appear in the first column. Open and/or closed brackets, as well as a straight horizontal line, can indicate whether or not a packet or group of packets are all part of the same back-and-forth conversation on the network. A broken horizontal line signifies that a packet is not part of said conversation.

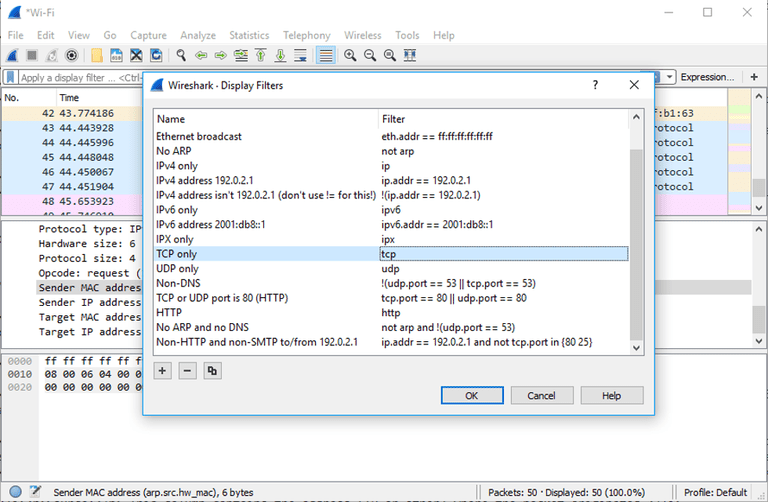
**Packet Details**

The details pane, found in the middle, presents the protocols and protocol fields of the selected packet in a collapsible format. In addition to expanding each selection, you can also apply individual Wireshark filters based on specific details as well as follow streams of data based on protocol type via the details context menu – accessible by right-clicking your mouse on the desired item within this pane

**Packet Bytes**

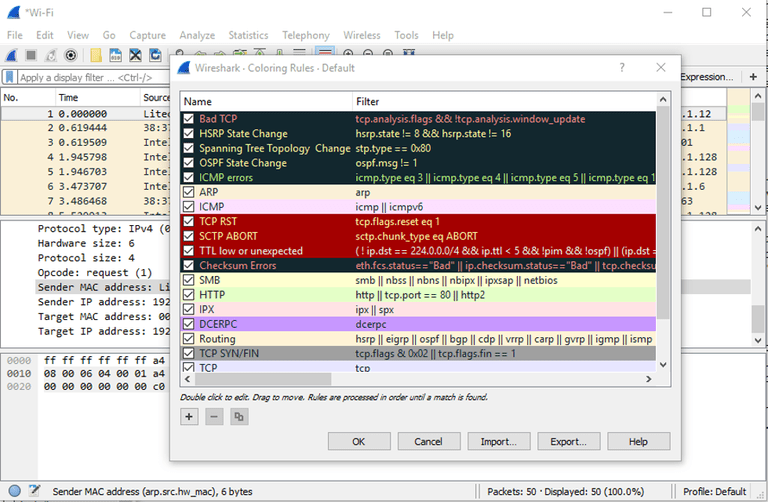
At the bottom is the packet bytes pane, which displays the raw data of the selected packet in a hexadecimal view. This hex dump contains 16 hexadecimal bytes and 16 ASCII bytes alongside the data offset. Selecting a specific portion of this data automatically highlights its corresponding section in the packet details pane and vice versa. Any bytes that cannot be printed are instead represented by a period. You can choose to show this data in bit format as opposed to hexadecimal by right-clicking anywhere within the pane and selecting the appropriate option from the context menu.

**4. Using Wireshark Filters**



One of the most important feature sets in Wireshark is its filter capabilities, especially when you're dealing with files that are significant in size. Capture filters can be set before the fact, instructing Wireshark to only record those packets that meet your specified criteria. Filters can also be applied to a capture file that has already been created so that only certain packets are shown. These are referred to as display filters. Wireshark provides a large number of predefined filters by default, letting you narrow down the number of visible packets with just a few keystrokes or mouse clicks. To use one of these existing filters, place its name in the *Apply a display filter*entry field (located directly below the Wireshark toolbar) or in the *Enter a capture filter* entry field (located in the center of the welcome screen). There are multiple ways to achieve this. If you already know the name of your filter, simply type it into the appropriate field. For example, if you only wanted to display TCP packets you would type *tcp*. Wireshark's autocomplete feature will show suggested names as you begin typing, making it easier to find the correct moniker for the filter you're seeking. Another way to choose a filter is to click on the bookmark-like icon positioned on the left-hand side of the entry field. This will present a menu containing some of the most commonly-used filters as well as an option to *Manage Capture Filters* or *Manage Display Filters*. If you choose to manage either type an interface will appear allowing you to add, remove or edit filters. You can also access previously-used filters by selecting the down arrow, located on the right hand side of the entry field, which displays a history drop-down list. Once set, capture filters will be applied as soon as you begin recording network traffic. To apply a display filter, however, you'll need to click on the right arrow button found on the far-right hand side of the entry field.

**5. Coloring Rules**



While Wireshark's capture and display filters allow you to limit which packets are

recorded or shown on the screen, its colorization functionality takes things a step further by

making it easy to distinguish between different packet types based on their individual hue. This handy feature lets you quickly locate certain packets within a saved set by their row's color scheme in the packet list pane. Wireshark comes with about 20 default coloring rules built in; each which can be edited, disabled or deleted if you wish. You can also add new shade-based filters through the coloring rules interface, acessible from the *View* menu. In addition to defining a name and filter criteria for each rule, you are also asked to associate both a background color and a text color. Packet colorization can be toggled off and on via the *Colorize Packet List* option, also found within the *View* menu.

**6. Statistics**



In addition to the detailed information about your network's data shown in Wireshark's main window, several other useful metrics are available via the *Statistics* drop-down menu found towards the top of the screen. These include size and timing information about the capture file itself, along with dozens of charts and graphs ranging in topic from packet conversation breakdowns to load distribution of HTTP requests. Display filters can be applied to many of these statistics via their individual interfaces, and the results can be exported to several common file formats including CSV, XML, and TXT.

To install Wireshark:

1. Open Windows Explorer.

2. Select the Downloads folder.

3. Locate the version of Wireshark you downloaded in Activity 2. Double-click on the file to open it.

4. If you see a User Account Control dialog box, select Yes to allow the program to make

changes to this computer.

5. Select Next > to start the Setup Wizard.

6. Review the license agreement. If you agree, select I Agree to continue.

7. Select Next > to accept the default components.

8. Select the shortcuts you would like to have created. Leave the file extensions selected.

Select Next > to continue.

9. Select Next > to accept the default install location.

10.Select Install to begin installation.

11.Select Next > to install WinPcap.

12.Select Next > to start the Setup Wizard

Select Finish to complete the installation of Wireshark

**Conclusion**: Thus we studied Installation of network analyser tool Wireshark.

**Experiment No 12.**

**Title: -** TCP Client-Server Socket Programming.

**Aim:** -Study & implementation TCP Client-Socket Programming.

**Objective:**- To check how the communication between sender & receiver is done by the TCP connection oriented Protocol.

**Relevance:**-For Network Communication By socket programming.

**Theory:**- The typical scenario that take place for a connection-oriented transfer – First the server is started, and then sometime later a client is started. Connection is established between the client & server.

Following are some primitive that is used for the TCP socket programming.

|  |  |
| --- | --- |
| **Primitive** | **Meaning** |
| SOCKET | Create a new communication end point. |
| BIND | Attach a local address to a socket |
| LISTEN | Announce willingness to accept connection |
| ACCEPT | Block the caller until a connection attempt arrive |
| CONNECT | Actively attempt to establish a connection |
| SEND | Send some data over the connection |
| RECEIVE | Receive some data from the connection |
| CLOSE | Release the connection |

**Server**

Socket()

Data(reply)

Data(req)

Connection established

Read()

Write()

Socket()

Connect()

Block until connection from client

Process request

Write()

Read()

accept()

listen()

bind()

**Client**

Fig:-Socket system call for connection oriented protocol.

Berkely Socket:-

Primitive:

1. SOCKET:-This primitive create a new end point & allocate table space fir it with in the Transport Layer. The parameter of the call specify the addressing format to be used, the type of service desired & protocol.

A successful SOCKET call returns an ordinary file descriptor for use in succeeding calls, the same way an OPEN call done.

2) BIND:- The newly created socket do not have addresses. These are assigned using the BIND primitive. Once the server has bound an address to socket, remote clients are connect to it.

3) LISTEN:-This system call allocate space to queue incoming calls for the case that several clients try to connect at same time.

4) ACCEPT:- To block waiting for an incoming connection ,the server execute an ACCEPT system call. When the TPDU asking for the connection arrive ,the transport entity create a new socket with the same properties as the original one & file descriptor is returned. The server fork the process to handle the connection & go back to waiting for the next connection on the original socket.

At the client side:

Here a socket is created by the SOCKET system call ,but the BIND is not required since the address used does not matter to server. The CONNECT primitive blocks the caller & actively start the connection process. When it completes the client process is unblocked & the connection is established .

Both sides can now use SEND & RECEIVE system call to transmit & receive the data in full duplex manner.

//**TCPServer.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
int main(){  
  
char \*ip = "127.0.0.1";  
int port = 5566;  
  
int server\_sock, client\_sock;  
struct sockaddr\_in server\_addr, client\_addr;  
socklen\_t addr\_size;  
char buffer[1024];  
int n;  
  
server\_sock = socket(AF\_INET, SOCK\_STREAM, 0);  
if (server\_sock < 0){  
perror("[-]Socket error");  
exit(1);  
}  
printf("[+]TCP server socket created.\n");  
  
memset(&server\_addr, '\0', sizeof(server\_addr));  
server\_addr.sin\_family = AF\_INET;  
server\_addr.sin\_port = port;  
server\_addr.sin\_addr.s\_addr = inet\_addr(ip);  
  
n = bind(server\_sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));  
if (n < 0){  
perror("[-]Bind error");  
exit(1);  
}  
printf("[+]Bind to the port number: %d\n", port);  
  
listen(server\_sock, 5);  
printf("Listening...\n");  
  
while(1){  
addr\_size = sizeof(client\_addr);  
client\_sock = accept(server\_sock, (struct sockaddr\*)&client\_addr, &addr\_size);  
printf("[+]Client connected.\n");  
  
bzero(buffer, 1024);  
recv(client\_sock, buffer, sizeof(buffer), 0);  
printf("Client: %s\n", buffer);  
  
bzero(buffer, 1024);  
strcpy(buffer, "HI, THIS IS SERVER. HAVE A NICE DAY!!!");  
printf("Server: %s\n", buffer);  
send(client\_sock, buffer, strlen(buffer), 0);  
close(client\_sock);  
printf("[+]Client disconnected.\n\n");  
  
}  
return 0;  
}

// **TCPClient.c**

#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <unistd.h>  
#include <arpa/inet.h>  
  
int main(){  
  
char \*ip = "127.0.0.1";  
int port = 5566;  
  
int sock;  
struct sockaddr\_in addr;  
socklen\_t addr\_size;  
char buffer[1024];  
int n;  
  
sock = socket(AF\_INET, SOCK\_STREAM, 0);  
if (sock < 0){  
perror("[-]Socket error");  
exit(1);  
}  
printf("[+]TCP server socket created.\n");  
  
memset(&addr, '\0', sizeof(addr));  
addr.sin\_family = AF\_INET;  
addr.sin\_port = port;  
addr.sin\_addr.s\_addr = inet\_addr(ip);  
  
connect(sock, (struct sockaddr\*)&addr, sizeof(addr));  
printf("Connected to the server.\n");  
  
bzero(buffer, 1024);  
strcpy(buffer, "HELLO, THIS IS CLIENT.");  
printf("Client: %s\n", buffer);  
send(sock, buffer, strlen(buffer), 0);  
  
bzero(buffer, 1024);  
recv(sock, buffer, sizeof(buffer), 0);  
printf("Server: %s\n", buffer);  
  
close(sock);  
printf("Disconnected from the server.\n");  
  
return 0;  
}

**Conclusion:-**In this way we have check how client & server are communicate with each other by using TCP.

**Experiment No 13**

**Title :-** UDP Client-Server Socket Programming.

**Aim:**-Study & implementation UDP Client-Socket Programming.

**Objective:**- To check how the communication between sender & receiver is done by the TCP connectionless Protocol.

**Relevance:**-For Network Communication By socket programming.

**Theory:**-

The **User Datagram Protocol** (**UDP**) is one of the core members of the [Internet protocol suite](http://en.wikipedia.org/wiki/Internet_protocol_suite" \o "Internet protocol suite) (the set of network protocols used for the [Internet](http://en.wikipedia.org/wiki/Internet" \o "Internet)). With UDP, computer applications can send messages, in this case referred to as *[datagrams](http://en.wikipedia.org/wiki/Datagram" \o "Datagram)*, to other hosts on an [Internet Protocol](http://en.wikipedia.org/wiki/Internet_Protocol" \o "Internet Protocol) (IP) network without prior communications to set up special transmission channels or data paths. The protocol was designed by [David P. Reed](http://en.wikipedia.org/wiki/David_P._Reed" \o "David P. Reed) in 1980 and formally defined in [RFC 768](http://tools.ietf.org/html/rfc768).

UDP uses a simple transmission model with a minimum of protocol mechanism. It has no [handshaking](http://en.wikipedia.org/wiki/Handshaking" \o "Handshaking) dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. As this is normally [IP](http://en.wikipedia.org/wiki/IP_address" \o "IP address) over unreliable media, there is no guarantee of delivery, ordering, or duplicate protection. UDP provides [checksums](http://en.wikipedia.org/wiki/Checksums" \o "Checksums) for data integrity, and [port numbers](http://en.wikipedia.org/wiki/Port_numbers" \o "Port numbers) for addressing different functions at the source and destination of the datagram.

UDP is suitable for purposes where error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system. If error correction facilities are needed at the network interface level, an application may use the [Transmission Control Protocol](http://en.wikipedia.org/wiki/Transmission_Control_Protocol" \o "Transmission Control Protocol) (TCP) or [Stream Control Transmission Protocol](http://en.wikipedia.org/wiki/Stream_Control_Transmission_Protocol" \o "Stream Control Transmission Protocol) (SCTP) which are designed for this purpose.

|  |  |
| --- | --- |
| **Primitive** | **Meaning** |
| SOCKET | Create a new communication end point. |
| BIND | Attach a local address to a socket |
| LISTEN | Announce willingness to accept connection |
| ACCEPT | Block the caller until a connection attempt arrive |
| CONNECT | Actively attempt to establish a connection |
| SEND | Send some data over the connection |
| RECEIVE | Receive some data from the connection |
| CLOSE | Release the connection |

Berkely Socket:-

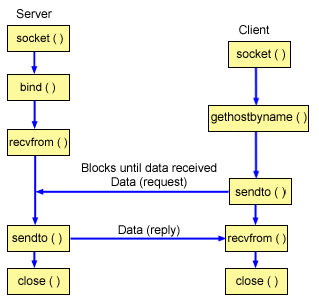
Primitive:

1. SOCKET:-This primitive create a new end point & allocate table space fir it with in the Transport Layer. The parameter of the call specify the addressing format to be used, the type of service desired & protocol.

A successful SOCKET call returns an ordinary file descriptor for use in succeeding calls, the same way an OPEN call done.

1. BIND:- The newly created socket do not have addresses. These are assigned using the BIND primitive. Once the server has bound an address to socket, remote clients are connect to it.

**UDP System calls:**



**Programming with UDP/IP sockets**

There are a few steps involved in using sockets:

1. Create the socket
2. Identify the socket (name it)
3. On the server, wait for a message
4. On the client, send a message
5. Send a response back to the client (optional)
6. Close the socket

Udp\_ser.c

#include<stdio.h>

#include<netinet/in.h>

#include<unistd.h>

#include<sys/socket.h>

#include<sys/types.h>

#include<string.h>

#include<netdb.h>

#include<arpa/inet.h>

int main()

{

struct sockaddr\_in sock;

int fd,i,sa,j;

char str1[20];

int in;

fd = socket(PF\_INET,SOCK\_DGRAM,0);

sock.sin\_family = AF\_INET;

sock.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

sock.sin\_port = htons(3000);

sa = sizeof(sock);

bind(fd,(struct sockaddr\*)&sock,sizeof(sock));

while(1)

{

printf("\n\n\tWaiting for the data.....");

recvfrom(fd,str1,sizeof(str1),0,(struct sockaddr\*)&sock,&sa);

if(strcmp(str1,"###") == 0)

{

printf("\n\n\tConnection closed.....\n");

break;

}

printf("\n\n\tData is : %s",str1);

in = atoi(str1);

j=1;

for(i=1;i<=in;i++)

{

j = j\*i;

}

printf("\n\n\tFactorial of the received data is : %d",j);

}

close(fd);

return 0;

}

Udp\_Client.c

#include<netinet/in.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<stdio.h>

#include<unistd.h>

#include<string.h>

#include<netdb.h>

int fd;

int main()

{

struct sockaddr\_in sock;

char data[20];

fd = socket(PF\_INET,SOCK\_DGRAM,0);

sock.sin\_family = AF\_INET;

sock.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

sock.sin\_port = htons(3000);

while(1)

{

if(strcmp(data,"###") == 0)

{

printf("\n\tConnection closed....\n");

break;

}

printf("\n\tEnter the data to be transmit :");

scanf("%s",data);

printf("\n");

sendto(fd,data,sizeof(data),0,(struct sockaddr\*)&sock,sizeof(sock));

printf("\n\tdata is : %s\n",data);

}

close(fd);

}

**Conclusion:**

In this way we have check how client & server are communicate with each other by using UDP.