III B.Tech-I SEM **Regulation: R23**

Laboratory Manual

For the course of

COMPUTER NETWORKS LAB

Branch: CSE/CSD/CAI/CSM/AIML

VIGNAN'S LARA
INSTITUTE OF TECHNOLOGY & SCIENCE
(AUTONOMOUS)
Approved by AICTE New Delhi & Affiliated to JNTUK Kakinada
Accredited by NAAC 'A+' and NBA | ISO 9001 : 2015
Vadlamudi - 522 213, Guntur District

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

III B.Tech I Semester	COMPUTERNETWORKSLAB	L	T	P	C
		0	0	3	1.5

List of Experiments:

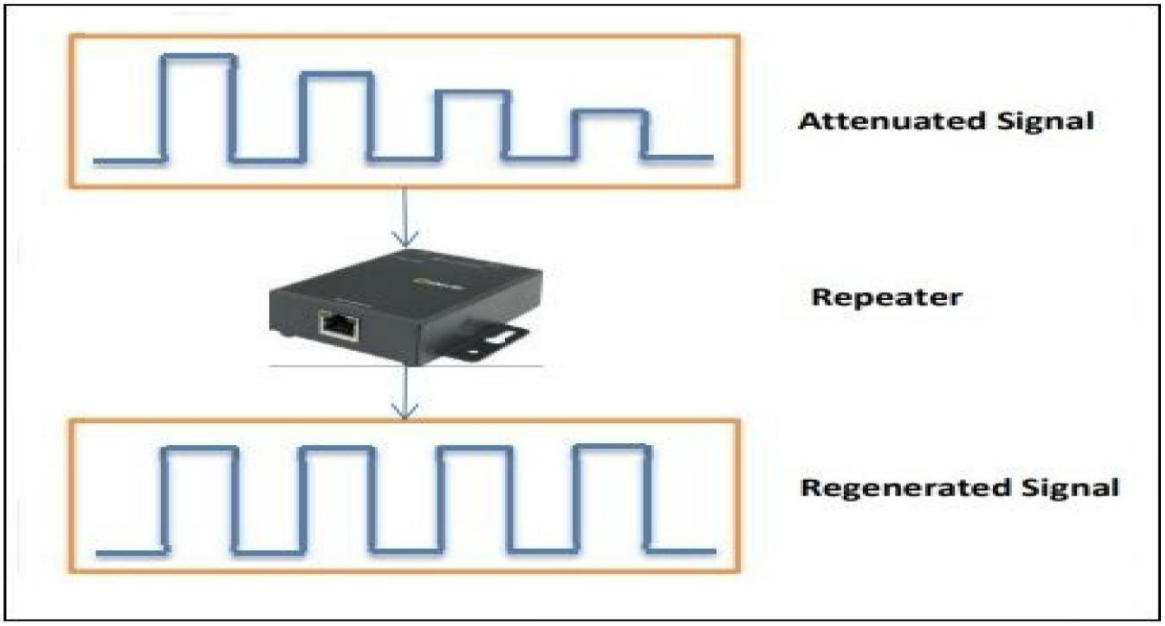
1. Study of Network devices in detail and connect the computers in Local Area Network.
2. Write a Program to implement the data link layer farming methods such as
   i)Character stuffing ii)bit stuffing.
3. Write a Program to implement data link layer farming method checksum.
4. Write a program for Hamming Code generation for error detection and correction.
5. Write a Program to implement on a data set of characters the three CRC polynomials–CRC12, CRC 16 and CRC CCIP.
6. Write a Program to implement Sliding window protocol for GobackN.
7. Write a Program to implement Sliding window protocol for Selective repeat.
8. Write a Program to implement Stop and Wait Protocol.
9. Write a program for congestion control using leaky bucket algorithm
10. Write a Program to implement Dijkstra,,s algorithm to compute the Shortest path through a graph.
11. Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node (Take an example subnet graph with weights indicating delay between nodes).
12. Write a Program to implement Broad cast tree by taking subnet of hosts.
13. Wireshark
    i. Packet Capture Using Wireshark.
    ii. Starting Wireshark.
    iii. Viewing Captured Traffic.
    iv. Analysis and Statistics & Filters.
14. How to run Nmap scan.
15. Operating System Detection using Nmap.
16. Do the following using NS2 Simulator.
    i. NS2 Simulator-Introduction.
    ii. Simulate to Find the Number of Packets Dropped.
    iii. Simulate to Find the Number of Packets Dropped by TCP/UDP..
    iv. Simulate to Find the Number of Packets Dropped due to Congestion.

EXPERIMENT-1

AIM: Study Of Different Network Devices In Detail (Repeater, Hub, Switch, Bridge, Router, Gateway).

Repeaters:

➔ Repeaters are network devices operating at the physical layer of the OSI model that amplify or regenerate an incoming signal before retransmitting it.
➔ They are incorporated in networks to expand its coverage area. They are also known as signal boosters.
➔ Signals that carry information within a network can travel a fixed distance before attenuation endangers the integrity of the data.
➔ A repeater receives a signal and, before it becomes too weak or corrupted, regenerates the original bit pattern.
➔ The repeater then sends the refreshed signal.
➔ A repeater can extend the physical length of a LAN.
➔ The location of a repeater on a link is vital. A repeater must be placed so that a signal reaches it before any noise changes the meaning of any of its bits.
➔ If the corrupted bit travels much farther, however, accumulated noise can change its meaning completely.
➔ At that point, the original voltage is not recoverable, and the error needs to be corrected.
➔ A repeater placed on the line before the legibility of the signal becomes lost can still read the signal well enough to determine the intended voltages and replicate them in their original form.



Types of Repeaters:

➔ According to the types of signals that they regenerate, repeaters can be classified into two categories −
- Analog Repeaters − They can only amplify the analog signal.
- Digital Repeaters − They can reconstruct a distorted signal.

➔ According to the types of networks that they connect, repeaters can be categorized into two types −
- Wired Repeaters − They are used in wired LANs.

● Wireless Repeaters − They are used in wireless LANs and cellular networks.

➔ According to the domain of LANs they connect, repeaters can be divided into two categories −

● Local Repeaters − They connect LAN segments separated by small distance.

● Remote Repeaters − They connect LANs that are far from each other.

Advantages of Repeaters:

● Repeaters are simple to install and can easily extend the length or the coverage area of networks.

● They are cost effective.

● Repeaters don't require any processing overhead. The only time they need to be investigated is in case of degradation of performance.

● They can connect signals using different types of cables.

Disadvantages of Repeaters :

● Repeaters cannot connect dissimilar networks.

● They cannot differentiate between actual signal and noise.

● They cannot reduce network traffic or congestion.

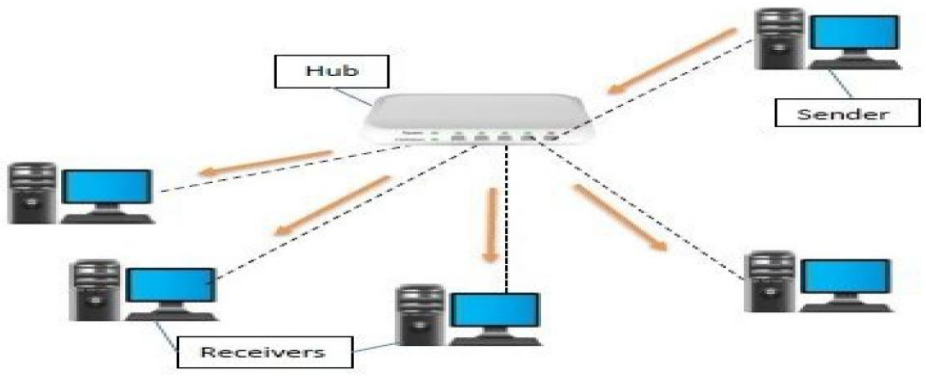
● Most networks have limitations upon the number of repeaters that can be deployed.

Hub:

➔ A hub is a physical layer networking device which is used to connect multiple devices in a network. They are generally used to connect computers in a LAN.

➔ A hub has many ports in it. A computer which intends to be connected to the network is plugged in to one of these ports.

➔ When a data frame arrives at a port, it is broadcast to every other port, without considering whether it is destined for a particular destination or



not.

Types of Hubs:

● Passive Hubs:

➔ A passive hub is just a connector. It connects the wires coming from different branches. In a star-topology Ethernet LAN, a passive hub is just a point where the signals coming from different stations collide; the hub is the collision point.

➔ This type of a hub is part of the media; its location in the Internet model is below the physical layer.

● Active Hubs:

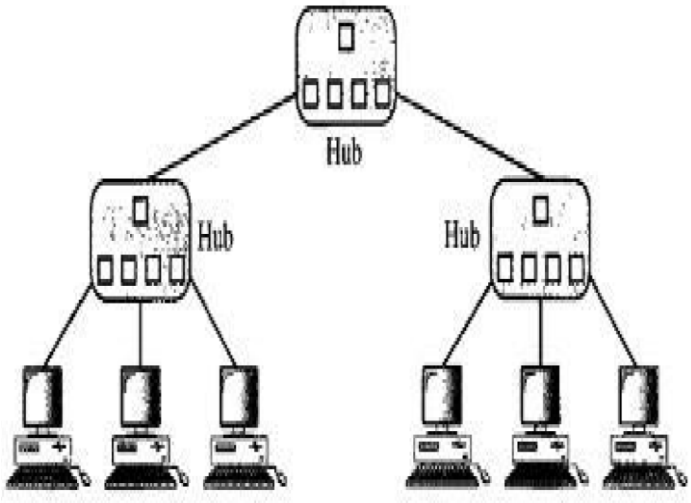
➔ An active hub is actually a multipart repeater. It is normally used to create connections between stations in a physical star topology.

➔ However, hubs can also be used to create multiple levels of hierarchy, as shown in Figure. The hierarchical use of hubs removes the length limitation of 10Base-T (100 m).

Intelligent Hubs:

- Intelligent Hubs:

➔ Intelligent hubs are active hubs that provide additional network management facilities. They can perform a variety of functions of more intelligent network devices like network management, switching, providing flexible data rates



etc

Bridges:

➔ A bridge operates in the physical layer as well as in the data link layer. It can regenerate the signal that it receives and as a data link layer device, it can check the physical addresses of source and destination contained in the frame.

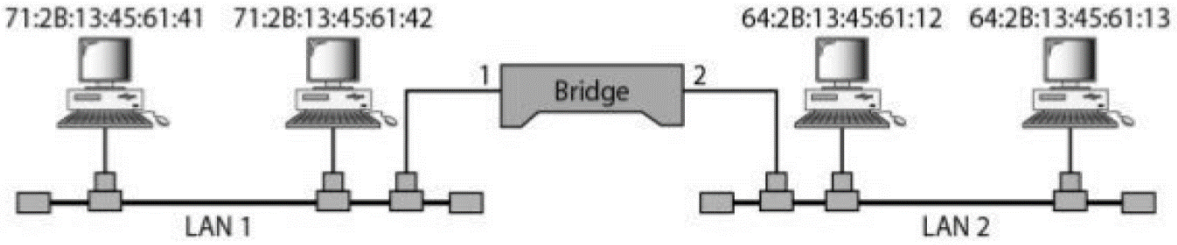
➔ The major difference between the bridge and the repeater is that the bridge and the repeater is that the bridge has a filtering capability.

➔ That means it can check the destination address of a frame and decide if the frame should be forwarded or dropped.

➔ If the frame is forwarded, then the bridge should specify the port over which it should be forwarded.

Address	Port
71:2B:13:45:61:41	1
71:2B:13:45:61:42	1
64:2B:13:45:61:12	2
64:2B:13:45:61:13	2

Bridge Table



Types of Bridges :

● Transparent Bridges:- These are the bridges in which the stations are completely unaware of the bridge's existence i.e. whether or not a bridge is added or deleted from the network, reconfiguration of the stations are unnecessary. These bridges make use of two processes i.e. bridge forwarding and bridge learning.

● Source Routing Bridges:- In these bridges, routing operation is performed by source station and the frame specifies which route to follow. The host can discover frames by sending a special frame called discovery frame, which spreads through the entire network using all possible paths to destination.

Router:

➔ Routers are networking devices operating at layer 3 or a network layer of the OSI model.

➔ They are responsible for receiving, analysing, and forwarding data packets among the connected computer networks.

➔ When a data packet arrives, the router inspects the destination address, consults its routing tables to decide the optimal route and then transfers the packet along this route.

➔ A router is a three-layer device that routes packets based on their logical addresses (host-to-host addressing).

➔ A router normally connects LANs and WANs in the Internet and has a routing table that is used for making decisions about the route.

➔ The routing tables are normally dynamic and are updated using routing protocols. Data is grouped into packets, or blocks of data.

➔ Each packet has a physical device address as well as logical network address. The network address allows routers to calculate the optimal path to a workstation or computer.

➔ The functioning of a router depends largely upon the routing table stored in it. The routing table stores the available routes for all destinations.

➔ The router consults the routing table to determine the optimal route through which the data packets can be sent

➔ A routing table typically contains the following entities −

● IP addresses and subnet mask of the nodes in the network
● IP addresses of the routers in the network
● Interface information among the network devices and channels

➔ Routing tables are of two types −

● Static Routing Table − Here, the routes are fed manually and are not refreshed automatically. It is suitable for small networks containing 2-3 routers.

● Dynamic Routing Table − Here, the router communicates with other routers using routing protocols to determine the available routes. It is suited for larger networks having large numbers of routers.

Types of Routers:

A variety of routers are available depending upon their usages. The main types of routers are as follows :-

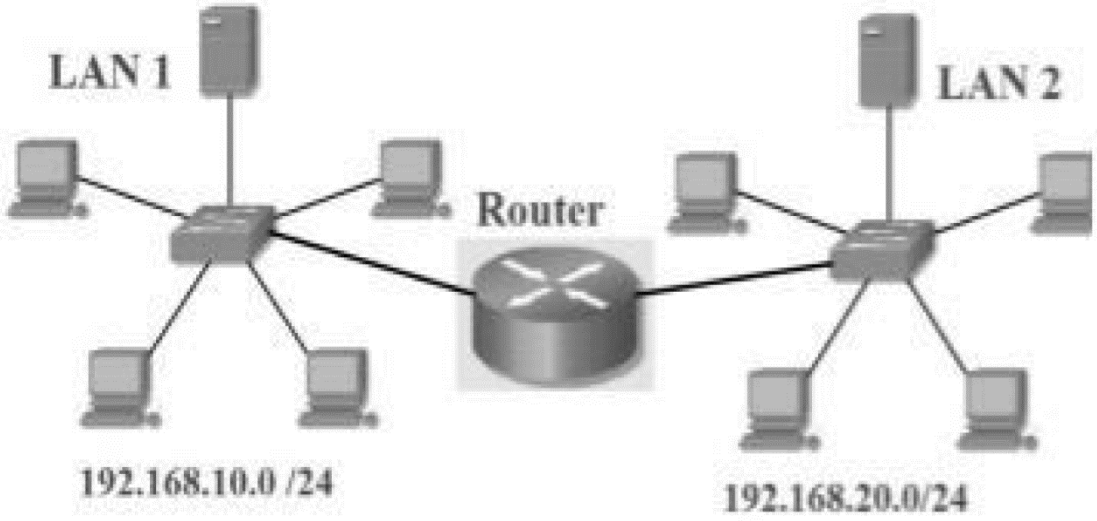
● Wireless Router − They provide WiFi connection WiFi devices like laptops, smartphones etc. They can also provide standard Ethernet routing. For indoor connections, the range is 150 feet while it's 300 feet for outdoor connections.

● Broadband Routers − They are used to connect to the Internet through telephone and to use voice over Internet Protocol (VoIP) technology for providing high-speed Internet access. They are configured and provided by the Internet Service Provider (ISP).

● Core Routers − They can route data packets within a given network, but cannot route the packets between the networks. They help to link all devices within a network thus forming the backbone of the network. It is used by ISP and communication interfaces.

● Edge Routers − They are low-capacity routers placed at the periphery of the networks. They connect the internal network to the external networks, and are suitable for transferring data packets across networks. They use the Border Gateway Protocol (BGP) for connectivity. There are two types of edge routers, subscriber edge routers and label edge routers.

● Brouters − Brouters are specialised routers that can provide the functionalities of bridges as well. Like a bridge, brouters help to transfer data between networks. And like a router, they route the data within the devices of a network.



Gateway:

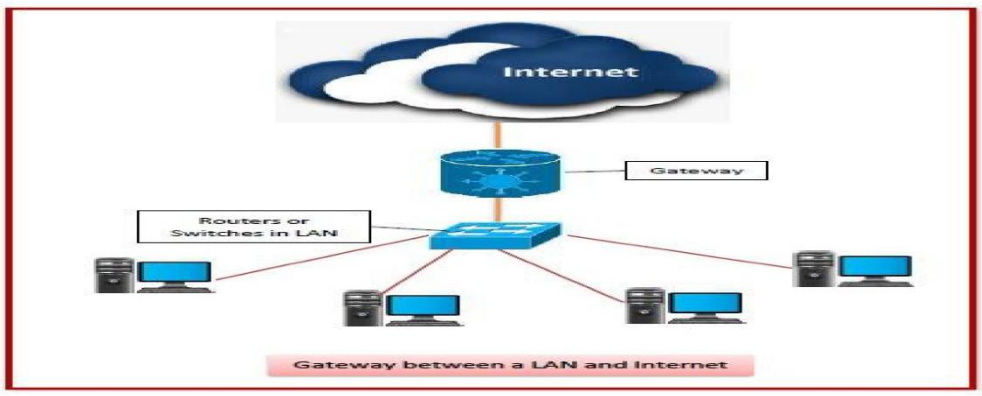
➔ A gateway is a network node that forms a passage between two networks operating with different transmission protocols.

➔ The most common type of gateways, the network gateway operates at layer 3, i.e. network layer of the OSI (open systems interconnection) model.

➔ However, depending upon the functionality, a gateway can operate at any of the seven layers of OSI model.

➔ It acts as the entry – exit point for a network since all traffic that flows across the networks should pass through the gateway.

➔ Only the internal traffic between the nodes of a LAN does not pass through the gateway.



➔ Gateway is located at the boundary of a network and manages all data that inflows or outflows from that network.
➔ It forms a passage between two different networks operating with different transmission protocols.
➔ A gateway operates as a protocol converter, providing compatibility between the different protocols used in the two different networks.
➔ The feature that differentiates a gateway from other network devices is that it can operate at any layer of the OSI model.
➔ It also stores information about the routing paths of the communicating networks.
➔ When used in enterprise scenarios, a gateway node may be supplemented as a proxy server or firewall.
➔ A gateway is generally implemented as a node with multiple NICs (network interface cards) connected to different networks. However, it can also be configured using software.
➔ It uses a packet switching technique to transmit data across the networks.
Types of Gateways :

On basis of direction of data flow, gateways are broadly divided into two categories −

- Unidirectional Gateways − They allow data to flow in only one direction. Changes made in the source node are replicated in the destination node, but not vice versa. They can be used as archiving tools.
- Bidirectional Gateways − They allow data to flow in both directions. They can be used as synchronization tools.

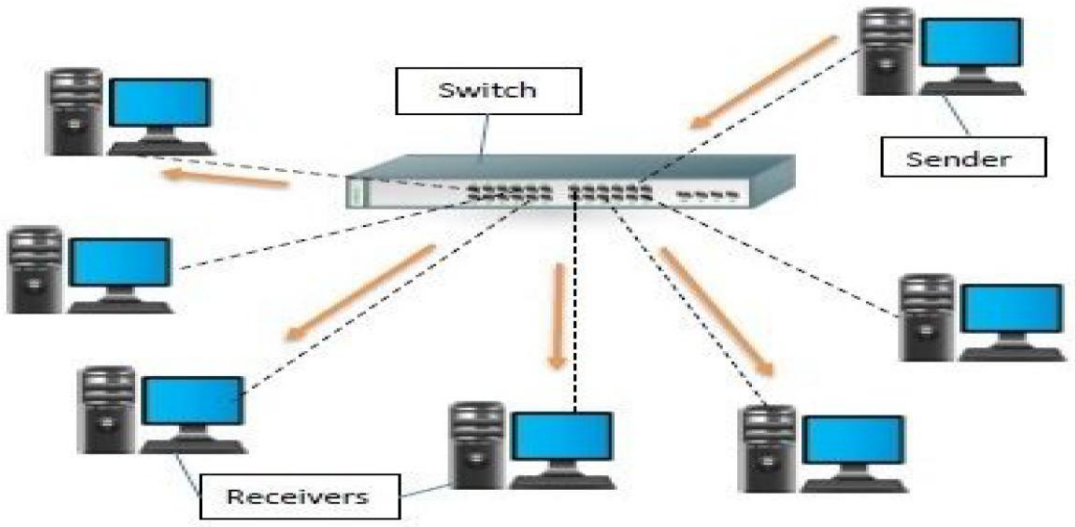
On basis of functionalities, there can be a variety of gateways, the prominent among them are as follows −

- Network Gateway − This is the most common type of gateway that provides an interface between two dissimilar networks operating with different protocols. Whenever the term gateway is mentioned without specifying the type, it indicates a network gateway.
- Cloud Storage Gateway − It is a network node or server that translates storage requests with different cloud storage service API calls, such as SOAP (Simple Object Access Protocol) or REST (REpresentational State Transfer).It facilitates integration of private cloud storage into applications without necessitating transfer of the applications into any public cloud, thus simplifying data communication.
- Internet-To-Orbit Gateway (I2O) − It connects devices on the Internet to satellites and spacecraft orbiting the earth. Two prominent I2O gateways are Project HERMES and Global Educational Network for Satellite Operations (GENSO).
- IoT Gateway − IoT gateways assimilates sensor data from IoT (Internet of Things) devices in the field and translates between sensor protocols before sending it to the cloud network. They connect IoT devices, cloud networks and user applications.
- VoiP Trunk Gateway − It facilitates data transmission between plain old telephone service (POTS) devices like landline phones and fax machines, with VoIP (voice over Internet Protocol) network.

Switch:

➔ A switch is a data link layer networking device which connects devices in a network and uses packet switching to send and receive data over the network.
➔ Like a hub, a switch also has many ports, to which computers are plugged in.
➔ However, when a data frame arrives at any port of a network switch, it examines the destination address and sends the frame to the corresponding device(s).

➔ Thus, it supports both unicast and multicast communications.



➔ We can have a two-layer switch or a three-layer switch.
➔ A three-layer switch is used at the network layer; it is a kind of router.
➔ The two-layer switch performs at the physical and data link layers.
➔ A two-layer switch is a bridge, a bridge with many ports and a design that allows better (faster) performance.
➔ A bridge with a few ports can connect a few LANs together. A bridge with many ports may be able to allocate a unique port to each station, with each station on its own independent entity.
➔ This means no competing traffic (no collision, as we saw in Ethernet).
➔ A two-layer switch, as a bridge does, makes a filtering decision based on the MAC Address of the frame it received.
➔ However, a two-layer switch can be more sophisticated. It can have a buffer to hold the frames for processing.
➔ It can have a switching factor that forwards the frames faster. Some new two-layer switches, called cut-through switches, have been designed to forward the frame as soon as they check the MAC addresses in the header of the frame.
Types of Switches :

➔ There are variety of switches that can be broadly categorised into 4 types :-
- Unmanaged Switch − These are inexpensive switches commonly used in home networks and small businesses. They can be set up by simply plugging in to the network, after which they instantly start operating. When more devices need to be added, more switches are simply added by this plug and play method. They are referred to as u managed since they do not require to be configured or monitored.
- Managed Switch:-

➔ These are costly switches that are used in organisations with large and complex networks, since they can be customized to augment the functionalities of a standard switch. The augmented features may be QoS (Quality of Service) like higher security levels, better precision control and complete network management. Despite their cost, they are preferred in growing organizations due to their scalability and flexibility. Simple Network Management Protocol (SNMP) is used for configuring managed switches.

- LAN Switch :-

➔ Local Area Network (LAN) switches connect devices in the internal LAN of an organization. They are also referred to as Ethernet switches or data switches. These switches are particularly helpful in reducing network congestion or bottlenecks. They allocate bandwidth in a manner so that there is no overlapping of data packets in a network.

- PoE Switch :-

➔ Power over Ethernet (PoE) switches are used in PoE Gigabit Ethernets. PoE technology combines data and power transmission over the same cable so that devices connected to it can receive both electricity as well as data over the same line. PoE switches offer greater flexibility and simplifies the cabling connections.

TO CONNECT COMPUTERS TO LOCAL AREA NETWORK.

LAN (Local Area Network) is a data communication network that locally connects network devices such as workstations, servers, routers, etc. to share the resources within a small area such as a building or campus. Physical or wireless connections are set up between workstations to share the resources. Ethernet and Wi-fi are the most important technologies of LAN. Personal networks at home, school, office, etc. are examples of LAN. These are generally privately-owned networks.

Requirements to set up LAN Network:

- Workstation/Personal devices: laptop, computer, mobile phones, etc.
- Network devices: router, switch, modem (if not already present in the router)
- Sharing resources: printers, disk drives, etc.
- Cables: Ethernet cables, wires for connecting other devices (in case of wired LAN)
- Internet connection: Wi-Fi (in case of wireless LAN)

Instructions to set up LAN Network:

Following steps should be followed to set up a LAN network:

1. Identify services: Identify the network services such as printers, disk drives, data, etc. that will be shared among workstations.
2. Identify devices: Identify devices such as computers, mobile phones, laptops, etc. with a unique address that will be connected to the network.
3. Plan connections: Design the network by laying out cable wires between network devices or by making wireless connections. Wired LAN is set up using Ethernet cables while wireless LAN is set up using Wi-Fi that connects network devices without making any physical connection. A wired LAN network is more secure than a wireless LAN network but it is difficult to relocate.
4. Select networking device: Select switch or router with enough ports to connect all workstations within the network. The choice of networking device is based on the requirements of the network.
5. Configure ports: Configure WAN ports according to the information provided by ISP (Internet Service Provider). Also, configure LAN ports of cable routers such that there are enough addresses available for all the workstations within the network. A cable router acts as DHCP (Dynamic Host Configuration Server) server that automatically allocates addresses to all the devices connected to the network.
6. Make connections: Connect all the devices using wires to configure a LAN network. Standard Ethernet cables are used to connect workstations and servers while Ethernet crossover cable is used to connect the switch to cable routers by connecting the standard port of the switch with router's LAN port.

7. For wireless LAN, connect all the devices to Wi-Fi with SSID (Service Set Identifier) provided by the router or switch to configure the LAN network.
8. Test the network: Test each of the workstation connected to the network and ensure every workstation have access to network services.

Tips for LAN Set-Up:

1. Make a comprehensive plan about connections before making actual connections to avoid confusion.
2. Carefully identify the requirements and size of the network and plan accordingly.
3. Smartly choose the networking device which provides more flexibility to the network.
4. Ensure the cable length is not more than 100 meters.
5. Avoid laying cables in air ducts unless fire rated.
6. Perform detailed testing after network set up to analyze the actual performance of the network.

Applications of LAN:

1. Resource sharing: LAN network allows workstations connected to the network to share resources such as printers, scanners, CD drives, etc. which reduces the cost of the set up of the network.
2. Software sharing: LAN network allows to share a single copy of licensed software among workstations connected to the network instead of purchasing separate software for each computer.
3. Internet sharing: LAN network facilitates sharing of internet connection among all the devices connected to the network.
4. Data sharing: LAN network allows different workstations to share the data and files with each other. It also allows access to data stored on the central server.
5. Communication: Devices connected to a LAN network can communicate with each other.

Advantages of LAN:

1. It is an easy and cheap way of communication within a small geographical location.
2. It is easy to expand the network by connecting workstations to a central server.
3. It is easy to manage the resources and data from the central server.
4. It is more secure since data is stored on a central server which denies illegal access to data.
5. It has high data transmission rates.

Disadvantages of LAN:

1. It has a high initial setup cost.
2. It violates the privacy of network users as administrators have access to all their data and files.
3. It can face security issues if the central server is not properly secured.
4. It needs regular maintenance to deal with issues such as software installations, hardware failures, cable disturbances, etc.
5. It restricts the size of the network.

EXPERIMENT-2

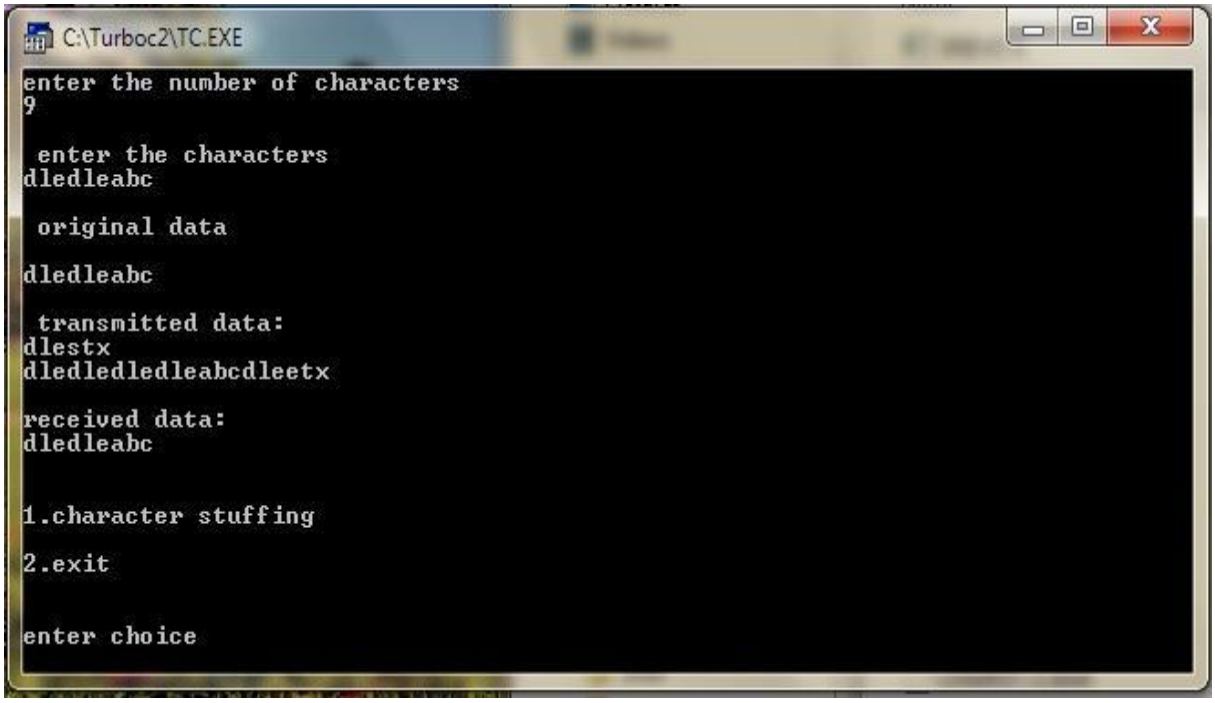
Write a Program to implement the data link layer farming methods such as
i) Character stuffing.

Aim: Write a Program to implement the data link layer farming methods such as

i) Character stuffing

```
#include<stdio.h>

#include<string.h>

main()

{

      char a[30], fs[50] = " ", t[3], sd, ed, x[3], s[3], d[3], y[3], ds[50] = " " ;

int i,len;

printf("Enter characters to be stuffed:");

scanf("%s", a);

printf("\nEnter a character that represents starting delimiter:");

scanf(" %c", &sd);

printf("\nEnter a character that represents ending delimiter:");

scanf(" %c", &ed);

x[0] = s[0] = s[1] = sd;

x[1] = s[2] = '\0';

y[0] = d[0] = d[1] = ed;

d[2] = y[1] = '\0';

strcat(fs, x);

for(i=0; i < strlen(a); i++)

      {

t[0] = a[i];

t[1] = '\0';

if(t[0] == sd)

strcat(fs, s);

else if(t[0] == ed)

strcat(fs, d);

else
```

```
strcat(fs, t);
        }
strcat(fs, y);
printf("\n After stuffing:%s", fs);
     //Character De stuffing from Stuffing
len=strlen(fs);
for(i=2;i<len-1;i++)
          {
                    fs[i-1]=fs[i];
          }
          fs[i-1]='\0';
          printf("\n After De stuffing:%s", fs);
getch();

   }
```



ii) Bit stuffing.

Aim: Write a Program to implement the data link layer farming methods such as

Theory

Security and Error detection are the most prominent features that are to be provided by any application which transfers data from one end to the other end. One of such a mechanism in tracking errors which may add up to the original data during transfer is known as Stuffing. It is of two types namely Bit Stuffing and the other Character Stuffing. Coming to the Bit Stuffing, 01111110 is appended within the original data while transfer of it. The following program describes how it is stuffed at the sender end and de-stuffed at the reciever end.

Program:

```
#include<stdio.h>
      main()
{
          int a[15];
          int i,j,k,n,c=0,pos=0;
          clrscr();
          printf("\n Enter the number of bits");
          scanf("%d",&n);
                printf("\n Enter the bits");
          for(i=0;i<n;i++)
          scanf("%d",&a[i]);
          for(i=0;i<n;i++)
 {
          if(a[i]==1)
{
          c++;
          if(c==5)
{
          pos=i+1;
          c=0;
          for(j=n;j>=pos;j--)
{
          k=j+1;
          a[k]=a[j];
}
          a[pos]=0;
          n=n+1;
}
}
          else
          c=0;
}

                printf("\n DATA AFTER STUFFING \n");
                printf(" 01111110 ");
```

```
        for(i=0;i<n;i++)
 {
                printf("%d",a[i]);
}
                printf(" 01111110 ");
        getch();
}
```

OUTPUT:

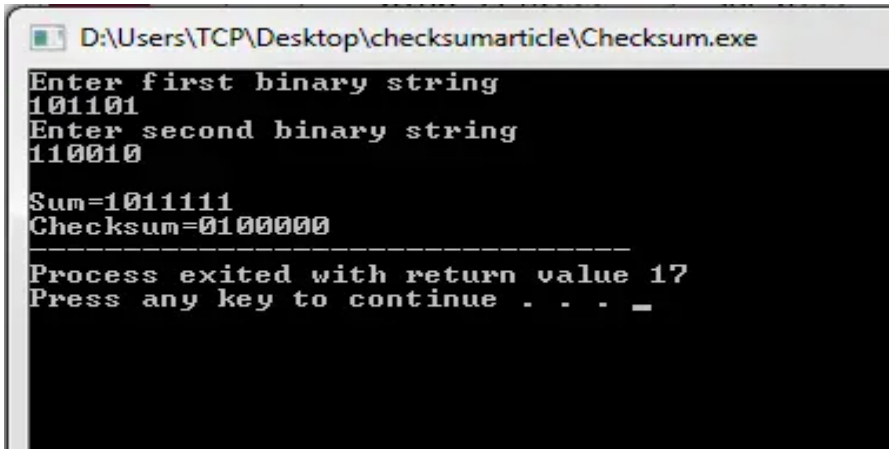
EXPERIMENT-3

AIM: Write a Program to implement data link layer farming method checksum.

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

using namespace std;

int main()
{
   char a[20],b[20];
   char sum[20],complement[20];
   int i;

cout<<"Enter first binary string\n";
   cin>>a;
   cout<<"Enter second binary string\n";
   cin>>b;
   E
if(strlen(a)==strlen(b))
   {
      char carry='0';
      int length=strlen(a);

for(i=length-1;i>=0;i--)
      {
         if(a[i]=='0' && b[i]=='0' && carry=='0')
         {
            sum[i]='0';
            carry='0';
         }
         else if(a[i]=='0' && b[i]=='0' && carry=='1')
         {
            sum[i]='1';
            carry='0';

         }
         else if(a[i]=='0' && b[i]=='1' && carry=='0')
         {
            sum[i]='1';
            carry='0';

         }
         else if(a[i]=='0' && b[i]=='1' && carry=='1')
         {
            sum[i]='0';
            carry='1';
```

```
            }
            else if(a[i]=='1' && b[i]=='0' && carry=='0')
            {
               sum[i]='1';
               carry='0';

            }
            else if(a[i]=='1' && b[i]=='0' && carry=='1')
            {
               sum[i]='0';
               carry='1';

            }
            else if(a[i]=='1' && b[i]=='1' && carry=='0')
            {
               sum[i]='0';
               carry='1';

            }
            else if(a[i]=='1' && b[i]=='1' && carry=='1')
            {
               sum[i]='1';
               carry='1';

            }
            else
               break;
         }
         cout<<"\nSum="<<carry<<sum;

         for(i=0;i<length;i++)
         {
            if(sum[i]=='0')
               complement[i]='1';
            else
               complement[i]='0';
         }

      if(carry=='1')
            carry='0';
         else
            carry='1';

      cout<<"\nChecksum="<<carry<<complement;
      }
      else
         cout<<"\nWrong input strings";

      return 0;
   }
```

OUTPUT:


D:\Users\TCP\Desktop\checksumarticle\Checksum.exe
Enter first binary string
101101
Enter second binary string
110010
Sum=1011111
Checksum=0100000
Process exited with return value 17
Press any key to continue . . .

EXPERIMENT-4

AIM: Write a program for Hamming Code generation for error detection and correction.

HammingCodeExample.jav

```
// import required classes and packages

package javaTpoint.JavaExample;
import java.util.*;

// create HammingCodeExample class to implement the Hamming Code functionality in Java

class HammingCodeExample {

        // main() method start
        public static void main(String args[])
        {
            // declare variables and array
            int size, hammingCodeSize, errorPosition;
            int arr[];
      int hammingCode[];
  // create scanner class object to take input from user
  Scanner sc = new Scanner(System.in);
  System.out.println("Enter the bits size for the data.");
  size = sc.nextInt();
  // initialize array
  arr = new int[size];
  // get data from user which we want to transfer
  for(int j = 0 ; j < size ; j++) {
     System.out.println("Enter " + (size - j) + "-bit of the data:");
     // fill array with user entered data
     arr[size - j - 1] = sc.nextInt();
  }

  // print the user entered data
  System.out.println("The data which you enter is:");
  for(int k = 0 ; k < size ; k++) {
     System.out.print(arr[size - k - 1]);
  }
  System.out.println();   // for next line

  // call getHammingCode() method and store its return value to the hammingCode array
  hammingCode = getHammingCode(arr);
  hammingCodeSize = hammingCode.length;

  System.out.println("The hamming code generated for your data is:");
  for(int i = 0 ; i < hammingCodeSize; i++) {
```

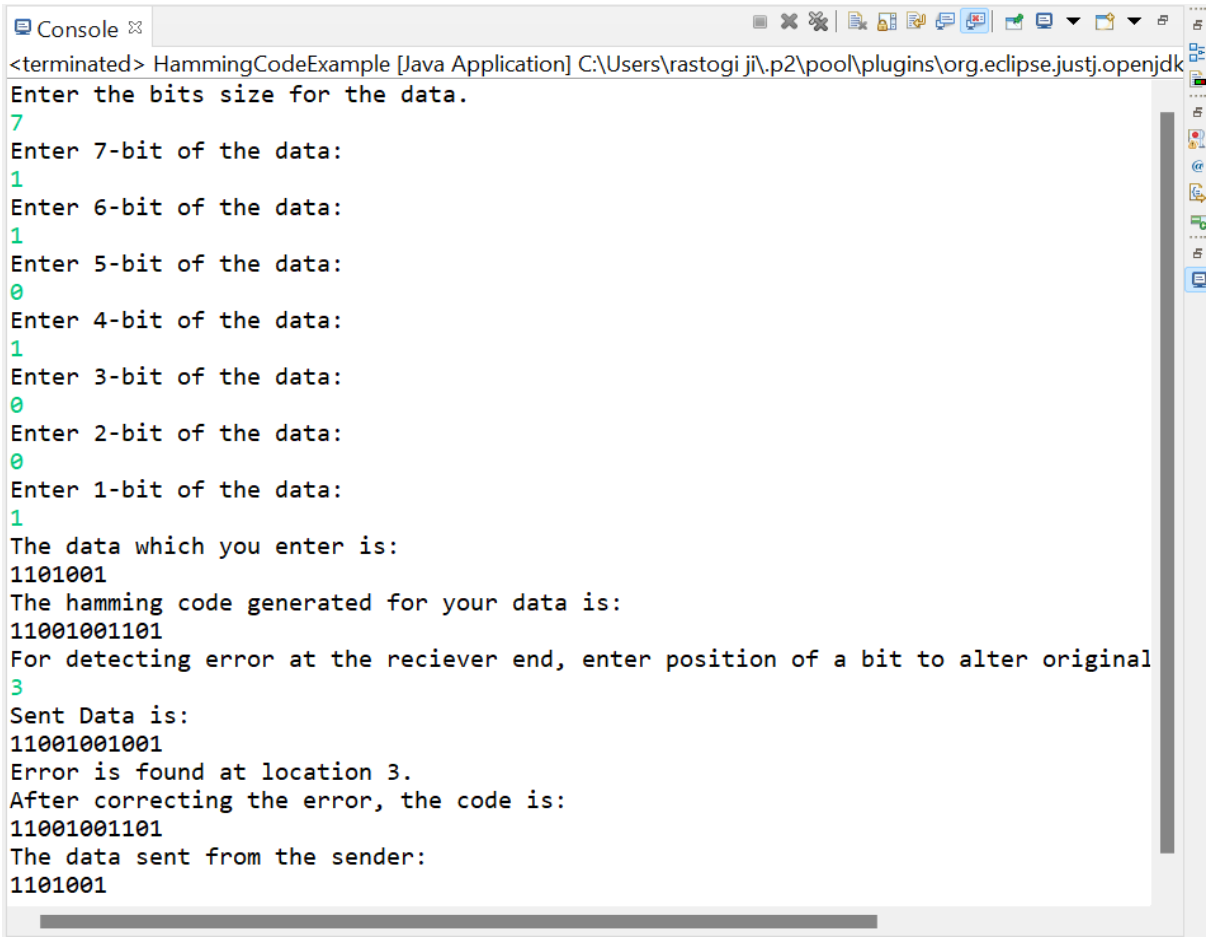
```
            System.out.print(hammingCode[(hammingCodeSize - i - 1)]);
        }
        System.out.println();   // for next line

        // The added parity bits are the difference b/w the original data and the returned hammingCo
de
        System.out.println("For detecting error at the reciever end, enter position of a bit to alter ori
ginal data "
                + "(0 for no error):");
        errorPosition = sc.nextInt();

        // close Scanner class object
        sc.close();

        // check whether the user entered position is 0 or not.
        if(errorPosition != 0) {
            // alter bit of the user entered position
            hammingCode[errorPosition - 1] = (hammingCode[errorPosition - 1] + 1) % 2;
        }

        // print sent data to the receiver
        System.out.println("Sent Data is:");
        for(int k = 0; k < hammingCodeSize; k++) {
            System.out.print(hammingCode[hammingCodeSize - k - 1]);
        }
        System.out.println();   // for next line
        receiveData(hammingCode, hammingCodeSize - arr.length);
    }
    // create getHammingCode() method that returns the hamming code for the data which we wan
t to send
    static int[] getHammingCode(int data[]) {
        // declare an array that will store the hamming code for the data
        int returnData[];
        int size;
        // code to get the required number of parity bits
        int i = 0, parityBits = 0 ,j = 0, k = 0;
        size = data.length;
        while(i < size) {
            // 2 power of parity bits must equal to the current position(number of bits traversed + num
ber of parity bits + 1).
            if(Math.pow(2, parityBits) == (i + parityBits + 1)) {
                parityBits++;
            }
            else {
                i++;
            }
        }

        // the size of the returnData is equal to the size of the original data + the number of parity bit
s.
```

```
    returnData = new int[size + parityBits];

    // for indicating an unset value in parity bit location, we initialize returnData array with '2'

    for(i = 1; i <= returnData.length; i++) {
       // condition to find parity bit location
       if(Math.pow(2, j) == i) {

          returnData[(i - 1)] = 2;
          j++;
       }
       else {
          returnData[(k + j)] = data[k++];
       }
    }
    // use for loop to set even parity bits at parity bit locations
    for(i = 0; i < parityBits; i++) {

       returnData[((int) Math.pow(2, i)) - 1] = getParityBit(returnData, i);
    }

    return returnData;
}

// create getParityBit() method that return parity bit based on the power
static int getParityBit(int returnData[], int pow) {
    int parityBit = 0;
    int size = returnData.length;

    for(int i = 0; i < size; i++) {

       // check whether returnData[i] contains an unset value or not
       if(returnData[i] != 2) {

          // if not, we save the index in k by increasing 1 in its value

          int k = (i + 1);

          // convert the value of k into binary
          String str = Integer.toBinaryString(k);

          //Now, if the bit at the 2^(power) location of the binary value of index is 1,
          // we check the value stored at that location. If the value is 1 or 0,
          // we will calculate the parity value.

          int temp = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
          if(temp == 1) {
             if(returnData[i] == 1) {
                parityBit = (parityBit + 1) % 2;
             }
```

```
            }
        }
    }
    return parityBit;
}

// create receiveData() method to detect error in the received data
static void receiveData(int data[], int parityBits) {

    // declare variable pow, which we use to get the correct bits to check for parity.
    int pow;
    int size = data.length;
    // declare parityArray to store the value of parity check
    int parityArray[] = new int[parityBits];
    // we use errorLoc string for storing the integer value of the error location.
    String errorLoc = new String();
    // use for loop to check the parities
    for(pow = 0; pow < parityBits; pow++) {
        // use for loop to extract the bit from 2^(power)
        for(int i = 0; i < size; i++) {
            int j = i + 1;
            // convert the value of j into binary
            String str = Integer.toBinaryString(j);
            // find bit by using str
            int bit = ((Integer.parseInt(str)) / ((int) Math.pow(10, pow))) % 10;
            if(bit == 1) {
                if(data[i] == 1) {
                    parityArray[pow] = (parityArray[pow] + 1) % 2;
                }
            }
        }
        errorLoc = parityArray[pow] + errorLoc;
    }
    // This gives us the parity check equation values.
    // Using these values, we will now check if there is a single bit error and then correct it.
    // errorLoc provides parity check eq. values which we use to check whether a single bit error
is there or not
    // if present, we correct it
    int finalLoc = Integer.parseInt(errorLoc, 2);
    // check whether the finalLoc value is 0 or not
    if(finalLoc != 0) {
        System.out.println("Error is found at location " + finalLoc + ".");
        data[finalLoc - 1] = (data[finalLoc - 1] + 1) % 2;
        System.out.println("After correcting the error, the code is:");
        for(int i = 0; i < size; i++) {
            System.out.print(data[size - i - 1]);
        }
        System.out.println();
    }
    else {
```

```
            System.out.println("There is no error in the received data.");
        }
        // print the original data
        System.out.println("The data sent from the sender:");
        pow = parityBits - 1;
        for(int k = size; k > 0; k--) {
            if(Math.pow(2, pow) != k) {
                System.out.print(data[k - 1]);
            }
            else {
                // decrement value of pow
                pow--;
            }
        }
        System.out.println();   // for next line
    }
}
```

OUTPUT:



EXPERIMENT-5

AIM: Write a Program to implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

Include header files stdio.h, string.h

```
#define N strlen(g)
char t[28],cs[28],g[28];
int a,e,c,b;
void xor()
{
for(c=1;c<N;c++)
cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc()
{
for(e=0;e<N;e++)
cs[e]=t[e];
do
{ if(cs[0]=='1')
xor();
for(c=0;c<N-1;c++)
cs[c]=cs[c+1];
cs[c]=t[e++];
}while(e<=a+N-1);
} int main()
{ int flag=0;
do{
printf("\n1.crc12\n2.crc16\ncrc ccit\n4.exit\n\nEnter your option.");
scanf("%d",&b);
switch(b)
{ case 1:strcpy(g,"1100000001111");
break;
case 2:strcpy(g,"11000000000000101");
break;
case 3:strcpy(g,"10001000000100001");
break;
case 4:return 0;
}
printf("\n enter data:");
scanf("%s",t);
printf("\n----------------------\n");
printf("\n generating polynomial:%s",g);
a=strlen(t);
for(e=a;e<a+N-1;e++)
t[e]='0';
printf("\n-------------------------\n");
printf("mod-ified data is:%s",t);
printf("\n----------------------\n");
crc();
```

```
printf("checksum is:%s",cs);
for(e=a;e<a+N-1;e++)
t[e]=cs[e-a];
printf("\n----------------------\n");
printf("\n final codeword is : %s",t);
printf("\n----------------------\n");
printf("\ntest error detection 0(yes) 1(no)?:");
scanf("%d",&e);
if(e==0)
{
do{
printf("\n\tenter the position where error is to be inserted:");
scanf("%d",&e);
}
while(e==0||e>a+N-1);
t[e-1]=(t[e-1]=='0')?'1':'0';
printf("\n----------------------\n");
printf("\n\terroneous data:%s\n",t);
} crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++);
if(e<N-1)
printf("error detected\n\n");
else
printf("\n no error detected \n\n");
printf("\n----------------------");
}while(flag!=1);
} crc();
for(e=0;(e<N-1)&&(cs[e]!='1');e++);
if(e<N-1)
printf("error detected\n\n");
else
printf("\n no error detected \n\n");
printf("\n----------------------");
}while(flag!=1);
```

OUTPUT:

```
1.crc12
2.crc16
3.crc ccit
4.exit
Enter your option.1
enter data:1100110011100011
----------------------
generating polynomial:1100000001111
-----------------------
mod-ified data is:11001100111000110000000000001100000001111
----------------------
checksum is:1101110110001
----------------------
final codeword is : 11001100111000111101110110001100000001111
-----------------------
test error detection 0(yes) 1(no)?:1
no error detected
----------------------
1.crc12
2.crc16
3.crc ccit
4.exit
Enter your option.2
enter data:11001100111000
----------------------
generating polynomial:11000000000000101
------------------------
mod-ified data is:11001100111000000000000000000000000000000101
```

EXPERIMENT-6

AIM: Write a Program to implement Sliding window protocol for Goback N.

P.S. Enter the inputs in the Client program after the connection is established with the Server.

/*Server Program*/

```
import java.net.*;
import java.io.*;
import java.util.*;
public class Server
{
public static void main(String args[]) throws Exception
{
ServerSocket server=new ServerSocket(6262);
System.out.println("Server established.");
Socket client=server.accept();
ObjectOutputStream oos=new ObjectOutputStream(client.getOutputStream());
ObjectInputStream ois=new ObjectInputStream(client.getInputStream());
System.out.println("Client is now connected.");
int x=(Integer)ois.readObject();
int k=(Integer)ois.readObject();
int j=0;
int i=(Integer)ois.readObject();
boolean flag=true;
Random r=new Random(6);
int mod=r.nextInt(6);
while(mod==1||mod==0)
mod=r.nextInt(6);
while(true)
{
int c=k;
for(int h=0;h<=x;h++)
{
System.out.print("|"+c+"|");
c=(c+1)%x;
}
System.out.println();
System.out.println();
if(k==j)
{
System.out.println("Frame "+k+" recieved"+"\n"+"Data:"+j);
j++;
System.out.println();
}
```

```
else
System.out.println("Frames recieved not in correct order"+"\n"+" Expected farme:" + j
+"\n"+ " Recieved frame no :"+ k);
System.out.println();
if(j%mod==0 && flag)
{
System.out.println("Error found. Acknowledgement not sent. ");
flag=!flag;
j–;
}
else if(k==j-1)
{
oos.writeObject(k);
System.out.println("Acknowledgement sent");
}
System.out.println();
if(j%mod==0)
flag=!flag;
k=(Integer)ois.readObject();
if(k==-1)
break;
i=(Integer)ois.readObject();
}
System.out.println("Client finished sending data. Exiting");
oos.writeObject(-1);
}
}
```

/*Client Program*/

```
import java.util.*;
import java.net.*;
import java.io.*;
public class Client
{
public static void main(String args[]) throws Exception
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.print("Enter the value of m : ");
int m=Integer.parseInt(br.readLine());
int x=(int)((Math.pow(2,m))-1);
System.out.print("Enter no. of frames to be sent:");
int count=Integer.parseInt(br.readLine());
int data[]=new int[count];
int h=0;
for(int i=0;i<count;i++)
{
System.out.print("Enter data for frame no " +h+ " => ");
data[i]=Integer.parseInt(br.readLine());
```

```
h=(h+1)%x;
}
Socket client=new Socket("localhost",6262);
ObjectInputStream ois=new ObjectInputStream(client.getInputStream());
ObjectOutputStream oos=new ObjectOutputStream(client.getOutputStream());
System.out.println("Connected with server.");
boolean flag=false;
GoBackNListener listener=new GoBackNListener(ois,x);
listener=new GoBackNListener(ois,x);
listener.t.start();
int strt=0;
h=0;
oos.writeObject(x);
do
{
int c=h;
for(int i=h;i<count;i++)
{
System.out.print("|"+c+"|");
c=(c+1)%x;
}
System.out.println();
System.out.println();
h=strt;
for(int i=strt;i<x;i++)
{
System.out.println("Sending frame:"+h);
h=(h+1)%x;
System.out.println();
oos.writeObject(i);
oos.writeObject(data[i]);
Thread.sleep(100);
}
listener.t.join(3500);
if(listener.reply!=x-1)
{
System.out.println("No reply from server in 3.5 seconds. Resending data from frame no " +
(listener.reply+1));
System.out.println();
strt=listener.reply+1;
flag=false;
}
else
{
System.out.println("All elements sent successfully. Exiting");
flag=true;
}
}while(!flag);
oos.writeObject(-1);
```

```
}
}

class GoBackNListener implements Runnable
{
Thread t;
ObjectInputStream ois;
int reply,x;
GoBackNListener(ObjectInputStream o,int i)
{
t=new Thread(this);
ois=o;
reply=-2;
x=i;
}
@Override
public void run() {
try
{
int temp=0;
while(reply!=-1)
{
reply=(Integer)ois.readObject();
if(reply!=-1 && reply!=temp+1)
reply=temp;
if(reply!=-1)
{
temp=reply;
System.out.println("Acknowledgement of frame no " + (reply%x) + " recieved.");
System.out.println();
}
}
reply=temp;
}
catch(Exception e)
{
System.out.println("Exception => " + e);
}
}
}
```

/*__CLIENT OUTPUT__

```
Enter the value of m : 7
Enter no. of frames to be sent:5
Enter data for frame no 0 => 1
Enter data for frame no 1 => 2
Enter data for frame no 2 => 3
```

Enter data for frame no 3 => 4
Enter data for frame no 4 => 5
Connected with server.
|0||1||2||3||4|

Sending frame:0

Acknowledgement of frame no 0 recieved.

Sending frame:1

Sending frame:2

Sending frame:3

Sending frame:4

Sending frame:5
*/

/***SERVER OUTPUT**

Server established.
Client is now connected.
|0||1||2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0|

Frame 0 recieved
Data:0
Acknowledgement sent

|1||2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1|

Frame 1 recieved
Data:1
Error found. Acknowledgement not sent.

|2||3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :2
|3||4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2||3|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :3
|4||5||6||7||8||9||10||11||12||13||14||15||16||17||18||19||20||21||22||23||24||25||26||27||28||29||30||31||32||33||34||35||36||37||38||39||40||41||42||43||44||45||46||47||48||49||50||51||52||53||54||55||56||57||58||59||60||61||62||63||64||65||66||67||68||69||70||71||72||73||74||75||76||77||78||79||80||81||82||83||84||85||86||87||88||89||90||91||92||93||94||95||96||97||98||99||100||101||102||103||104||105||106||107||108||109||110||111||112||113||114||115||116||117||118||119||120||121||122||123||124||125||126||0||1||2||3||4|

Frames recieved not in correct order
Expected farme:1
Recieved frame no :4
*/

EXPERIMENT-7

AIM: Write a Program to implement Sliding window protocol for Selective repeat.

Client Side :

```
//.............CLIENT SIDE (SELECTIVE REPEAT)...........//

import java.lang.System;
import java.net.*;
import java.io.*;
import java.text.*;
import java.util.Random;
import java.util.*;

public class cli {
        static Socket connection;

        public static void main(String a[]) throws SocketException {
                try {
                        int v[] = new int[10];
                        int n = 0;
                        Random rands = new Random();
                        int rand = 0;

                        InetAddress addr = InetAddress.getByName("Localhost");
                        System.out.println(addr);
                        connection = new Socket(addr, 8011);
                        DataOutputStream out = new DataOutputStream(
                                        connection.getOutputStream());
                        DataInputStream in = new DataInputStream(
                                        connection.getInputStream());
                        int p = in.read();
                        System.out.println("No of frame is:" + p);

                        for (int i = 0; i < p; i++) {
                                v[i] = in.read();
                                System.out.println(v[i]);
                                //g[i] = v[i];
                        }
                       rand = rands.nextInt(p);   //FRAME NO. IS RANDOMLY GENERATED
                        v[rand] = -1;
                        for (int i = 0; i < p; i++)
                         {
                        System.out.println("Received frame is: " + v[i]);

                                }
                        for (int i = 0; i < p; i++)
```

```
                        if (v[i] == -1) {
                System.out.println("Request to retransmit from packet no " + (i+1) + " again!!");
                                n = i;
                                out.write(n);
                                out.flush();
                        }

                System.out.println();

                        v[n] = in.read();
                        System.out.println("Received frame is: " + v[n]);



                System.out.println("quiting");
        } catch (Exception e) {
                System.out.println(e);
        }

    }
}
```

OUTPUT

```
[root@localhost sinhgad]# java cli
Localhost/127.0.0.1
No of frame is:8
30
40
50
60
70
80
90
100
Received frame is: 30
Received frame is: 40
Received frame is: 50
Received frame is: -1
Received frame is: 70
Received frame is: 80
Received frame is: 90
Received frame is: 100
Request to retransmit from packet no 4 again!!

Received frame is: 60
quiting
```

*/

SERVER SIDE:

```
//.....................SERVER SIDE (SELECTIVE REPEAT).........//

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.SocketException;

public class ser
{
	static ServerSocket Serversocket;
	static DataInputStream dis;
	static DataOutputStream dos;

	public static void main(String[] args) throws SocketException
	{

		try
		{
			int a[] = { 30, 40, 50, 60, 70, 80, 90, 100 };
			Serversocket = new ServerSocket(8011);
			System.out.println("waiting for connection");
			Socket client = Serversocket.accept();
			dis = new DataInputStream(client.getInputStream());
			dos = new DataOutputStream(client.getOutputStream());
			System.out.println("The number of packets sent is:" + a.length);
			int y = a.length;
			dos.write(y);
			dos.flush();

			for (int i = 0; i < a.length; i++)
			{
				dos.write(a[i]);
				dos.flush();
			}

			int k = dis.read();

			dos.write(a[k]);
			dos.flush();

		}
		catch (IOException e)
		{
			System.out.println(e);
```

```
                }
                finally
                {
                        try
                        {
                                dis.close();
                                dos.close();
                        }
                        catch (IOException e)
                        {
                                e.printStackTrace();
                        }

                }
        }
}
```

OUTPUT

```
Password:
[root@localhost sinhgad]# javac ser.java
[root@localhost sinhgad]# java ser
waiting for connection
The number of packets sent is:8
[root@localhost sinhgad]#
```

EXPERIMENT-8

AIM: Write a Program to implement Stop and Wait Protocol.

//**SENDER//**

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
class stopwaitsender

{

public static void main(String args[]) throws Exception

{

stopwaitsender sws = new stopwaitsender();

sws.run();
}

public void run() throws Exception

{

Scanner sc=new Scanner(System.in);
System.out.println("Enter no of frames to be sent:");
int n=sc.nextInt();

Socket myskt=new Socket("localhost",9999);

PrintStream myps=new PrintStream(myskt.getOutputStream());

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

{

if(i==n)

{

myps.println("exit");
```

```
break;

}

System.out.println("Frame no "+i+" is sent");
myps.println(i);

BufferedReader bf=new BufferedReader(new InputStreamReader(myskt.getInputStream()));

String ack=bf.readLine();

if(ack!=null)

{

System.out.println("Acknowledgement was Received from receiver");
i++;

Thread.sleep(4000);
}

else

{

myps.println(i);

}

}

}

}
```

//RECEIVER//

```
import java.io.*;
import java.net.*;
class stopwaitreceiver

{

public static void main(String args[])throws Exception
```

```
{

stopwaitreceiver swr = new stopwaitreceiver();

swr.run();
}

public void run() throws Exception

{

String temp=”any message”,str=”exit”;

ServerSocket myss=new ServerSocket(9999);

Socket ss_accept=myss.accept();

BufferedReader ss_bf=new BufferedReader(new
InputStreamReader(ss_accept.getInputStream()));

PrintStream myps=new PrintStream(ss_accept.getOutputStream());

while(temp.compareTo(str)!=0)

{

Thread.sleep(1000);
temp=ss_bf.readLine();

if(temp.compareTo(str)==0)

{ break;}

System.out.println(“Frame “+temp+” was received”);
Thread.sleep(500);
myps.println(“Received”);

}

System.out.println(“ALL FRAMES  WERE RECEIVED SUCCESSFULLY”);
}

}
```

OUTPUT FOR SENDER:

C:\javaprog>javac stopwaitsender.java
C:\javaprog>java stopwaitsender
Enter no of frames to be sent:
4
Frame no 0 is sent
Acknowledgement was Received from receiver
Frame no 1 is sent
Acknowledgement was Received from receiver
Frame no 2 is sent
Acknowledgement was Received from receiver
Frame no 3 is sent
Acknowledgement was Received from receiver

OUTPUT FOR RECEIVER:

C:\javaprog>javac stopwaitreceiver.java
C:\javaprog>java stopwaitreceiver
Frame 0 was received
Frame 1 was received
Frame 2 was received
Frame 3 was received
ALL FRAMES WERE RECEIVED SUCCESSFULLY
NOTE:
Create separate java file for client and server
Run the Server.java file first then Client on different CMD or Terminal
Download the file here: SENDER Reciever

EXPERIMENT-9

AIM: Write a program for congestion control using leaky bucket algorithm.

```
import java.util.Scanner;
publicclass Leaky {
    publicstaticvoid main(String args[]){
        Scanner sc=newScanner(System.in);
        int bucket=0,op_rate,i,n,bsize;
        System.out.println("enter number of packets");
        n=sc.nextInt();
        int pkt[]=newint[n];
        System.out.println("enter he output rate of the bucket");
        op_rate=sc.nextInt();
        System.out.println("enter the bucket size");
        bsize=sc.nextInt();
        System.out.println("enter the arriving packets(size)");
        for(i=0;i<n;i++)< span="" style="box-sizing: border-box;">
            pkt[i]=sc.nextInt();
        System.out.println("\nsec\tpsize\tbucketsize\taccept/reject\tpkt_send");
        System.out.println("-----------------------------------------------------");
        for(i=0;i<n;i++)< span="" style="box-sizing: border-box;">
        {
            System.out.print(i+1+"\t"+pkt[i]+"\t");
            if(bucket+pkt[i]<=bsize)
            {
```

```
                    bucket+=pkt[i];

System.out.println(bucket+"\t\taccept\t\t"+min(bucket,op_rate)+"\n");
                    bucket=sub(bucket,op_rate);
                }
                else
                {
System.out.println(bucket+"\t\treject\t"+(bucket+pkt[i]-bsize)
+"\t"+min(bsize,op_rate)+"\n");
                    bucket=bsize;
                    bucket=sub(bucket,op_rate);
                }
            }
            while(bucket!=0)
            {
                System.out.print(++i +"\t 0
\t"+bucket+"\t\taccept\t\t"+min(bucket,op_rate)+"\n");
                bucket=sub(bucket,op_rate);
            }
            sc.close();
<n;i++)< span="" style="box-sizing: border-box;"><n;i++)< span="" style="box-sizing:
border-box;">}
    staticint min(int a,int b)
{
        return(a<b)?a:b;< span="" style="box-sizing: border-box;">
    }
     staticint sub(int a,int b)
```

```
{
        return(a-b)>0?(a-b):0;
    }
}
```

OUTPUT:

```
enter number of packets
4
enter he output rate of the bucket

7
enter the bucket size

8

enter the arriving packets(size)

5

7

8

3

sec     psize   bucketsize      accept/reject   pkt_send

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

1       5       5               accept          5

2       7       7               accept          7

3       8       8               accept          7

4       3       4               accept          4
```

EXPERIMENT-10

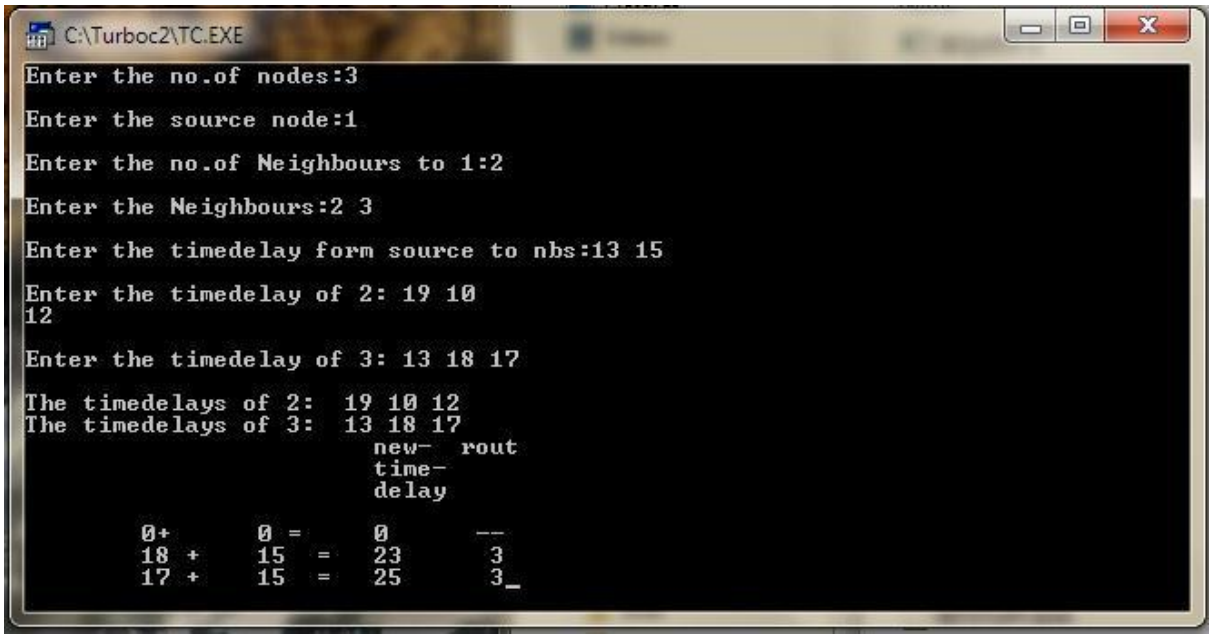
AIM: Write a Program to implement Dijkstra's algorithm to compute the Shortest path through a graph.

Program:

```
#include
#include
int n,s,nb,nbs[15],snbs[15],delay[15][15],i,j,temp[15],ze=0;
void min();
void main()
{
clrscr();
printf("Enter the no.of nodes:");
scanf("%d",&n);
printf("\nEnter the source node:");
scanf("%d",&s);
printf("\nEnter the no.of Neighbours to %d:",s);
scanf("%d",&nb);
printf("\nEnter the Neighbours:");
for(i=1;i<=nb;i++)
scanf("%d",&nbs[i]);
printf("\nEnter the timedelay form source to nbs:");
for(i=1;i<=nb;i++)
scanf("%d",&snbs[i]);
for(i=1;i<=nb;i++)
{
printf("\nEnter the timedelay of %d: ",nbs[i]);
for(j=1;j<=n;j++)
scanf("%d",&delay[i][j]);
}
for(i=1;i<=nb;i++)
{
printf("\nThe timedelays of %d: ",nbs[i]);
for(j=1;j<=n;j++)
printf("%3d",delay[i][j]);
}
min();
getch();
}
void min()
{
int sum,k,y=1,store=1;
printf("\n\t\t\tnew- rout");
printf("\n\t\t\ttime-");
printf("\n\t\t\tdelay");
printf("\n");
```

```
for(i=1;i<=n;i++)
{
sum=0;
k=1;
for(j=1;j<=nb;j++)
{
temp[k++]=delay[j][i];
}

sum=temp[1]+snbs[1];
for(y=2;y<=nb;y++)
{
if(sum>temp[y]+snbs[y])
{

sum=temp[y]+snbs[y];
store=y;
}
}

if(s==i)
printf("\n\t%d+\t%d =\t%d --",ze,ze,ze);
else
printf("\n\t%d +\t%d =\t%d\t%d",temp[store],snbs[store],sum,nbs[store]);
}
}
```

OUTPUT:



EXPERIMENT-11

AIM: Write a Program to implement Distance vector routing algorithm by obtaining routing table at each node.

```
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];

int main()
{
int dmat[20][20];
int n,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
scanf("%d",&n);
printf("\nEnter the cost matrix :\n");
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 {
scanf("%d",&dmat[i][j]);
dmat[i][i]=0;
rt[i].dist[j]=dmat[i][j];
rt[i].from[j]=j;
}
do
{
count=0;
for(i=0;i<n;i++)
 for(j=0;j<n;j++)
 for(k=0;k<n;k++)
 if(rt[i].dist[j]>dmat[i][k]+rt[k].dist[j])
{
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
}
}while(count!=0);
for(i=0;i<n;i++)
 {
printf("\n\nState value for router %d is \n",i+1);
for(j=0;j<n;j++)
 {
printf("\t\nnode %d via %d Distance%d",j+1,rt[i].from[j]+1,rt[i].dist[j]);
}
}
```

```
printf("\n\n");
}
```

OUTPUT:

Enter the number of nodes: 3

Enter the cost matrix:

0 2 7

2 0 1

7 1 0

State value for router 1 is

node 1 via 1 Distance0

node 2 via 2 Distance2

node 3 via 2 Distance3

State value for router 2 is

node 1 via 1 Distance2

node 2 via 2 Distance0

node 3 via 3 Distance1

State value for router 3 is

node 1 via 2 Distance3

node 2 via 2 Distance1

node 3 via 3 Distance0

EXPERIMENT-12

Aim: Write a Program to implement Broadcast tree by taking subnet of hosts.

a)Network Address:

```
#include
#include
void main()
{
unsigned int compad[4];
unsigned int mask[4];
unsigned int netadr[4];
int i;
clrscr();
printf("Enter the ip address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0])
;
printf("Enter the subnet address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&mask[3],&mask[2],&mask[1],&mask[0]);
for(i=0;i<4;i++)
{
netadr[i]= compad[i]&mask[i];
}
printf("\nNetwork address is:\n");
printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
printf("\nsubnet address is:\n");
printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
printf("\nip address is:\n");
printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
getch();
}
```

OUTPUT:

b)Network address with automatic subnet address generation:

```
#include
#include
void main()
{
unsigned int compad[4];
unsigned int mask[4];
unsigned int netadr[4];
unsigned long int ma=0;
int i,pre;
clrscr();
printf("Enter the ip address:\n");
scanf("%u%*c%u%*c%u%*c%u%*c",&compad[3],&compad[2],&compad[1],&compad[0])
;
printf("Enter the prefix:\n");
scanf("%ul",&pre);
for(i=(32-pre);i<32;i++)
ma=ma|(1<<i);
 for(i=0;i<4;i++)
{
mask[i]=ma%256;
ma=ma/256;
}
for(i=0;i<4;i++)
{
netadr[i]= compad[i]&mask[i];
}
printf("\nNetwork address is:\n");
printf("%u.%u.%u.%u",netadr[3],netadr[2],netadr[1],netadr[0]);
printf("\nsubnet address is:\n");
printf("%u.%u.%u.%u",mask[3],mask[2],mask[1],mask[0]);
printf("\nip address is:\n");
printf("%u.%u.%u.%u",compad[3],compad[2],compad[1],compad[0]);
getch();
}

</i);
</n;j++)
</n;i++)
</n;k++)
</n;j++)
</n;i++)
</n;j++)
</n;i++)
</n-1)&&(cs[e]!='1');e++);
</j;i++)
</m-6;i++,j++)
</m;i++)
```

```
</m+1;i++,j++)
</m+1;i++)
</m+1;i++)
</n;i++)
</n;i++)
</n;i++)
```

OUTPUT:

EXPERIMENT-13

13 Aim: Wireshark

i. **Packet Capture Using Wire shark**
ii. **Starting Wire shark**
iii. **Viewing Captured Traffic**
iv. **Analysis and Statistics & Filters.**

What is Wireshark? Wireshark is an open-source packet analyzer, which is used for **education, analysis, software development, communication protocol development, and network troubleshooting**. It is used to track the packets so that each one is filtered to meet our specific needs. It is commonly called as a **sniffer, network protocol analyzer, and network analyzer**. 11It is also used by network security engineers to examine security problems.

Wireshark is a free to use application which is used to apprehend the data back and forth. It is often called as a free packet sniffer computer application. It puts the network card into an unselective mode, i.e., to accept all the packets which it receives.

Uses of Wireshark:

Wireshark can be used in the following ways:

X
It is used by network security engineers to examine security problems.

1. It allows the users to watch all the traffic being passed over the network.
2. It is used by network engineers to troubleshoot network issues.
3. It also helps to troubleshoot latency issues and malicious activities on your network.
4. It can also analyze dropped packets.
5. It helps us to know how all the devices like laptop, mobile phones, desktop, switch, routers, etc., communicate in a local network or the rest of the world.

What is a packet?

A packet is a unit of data which is transmitted over a network between the origin and the destination. Network packets are small, i.e., maximum **1.5 Kilobytes for Ethernet packets and 64 Kilobytes for IP packets**. The data packets in the Wireshark can be viewed online and can be analyzed offline.

History of Wireshark:

In the late 1990's **Gerald Combs**, a computer science graduate of the University of Missouri-Kansas City was working for the small ISP (Internet Service Provider). The protocol at that time did not complete the primary requirements. So, he started writing **ethereal** and released the first version around 1998. The Network integration services owned the Ethernet trademark.

Combos still held the copyright on most of the ethereal source code, and the rest of the source code was re-distributed under the GNU GPL. He did not own the Ethereal trademark, so he changed the name to Wireshark. He used the contents of the ethereal as the basis.

Wireshark has won several industry rewards over the years including eWeek, InfoWorld, PC Magazine and also as a top-rated packet sniffer. Combos continued the work and released the new version of the software. There are around 600 contributed authors for the Wireshark product website.

Functionality of Wireshark:

Wireshark is similar to tcpdump in networking. **Tcpdump** is a common packet analyzer which allows the user to display other packets and TCP/IP packets, being transmitted and received over a network attached to the computer. It has a graphic end and some sorting and filtering functions. Wireshark users can see all the traffic passing through the network.

Wireshark can also monitor the unicast traffic which is not sent to the network's MAC address interface. But, the switch does not pass all the traffic to the port. Hence, the promiscuous mode is not sufficient to see all the traffic. The various network taps or **port mirroring** is used to extend capture at any point.

Port mirroring is a method to monitor network traffic. When it is enabled, the switch sends the copies of all the network packets present at one port to another port.

What is color coding in Wireshark?

The packets in the Wireshark are highlighted with **blue**, **black**, and **green color**. These colors help users to identify the types of traffic. It is also called as **packet colorization**. The kinds of coloring rules in the Wireshark are **temporary rules** and **permanent rules**.

- The temporary rules are there until the program is in active mode or until we quit the program.
- The permanent color rules are available until the Wireshark is in use or the next time you run the Wireshark. The steps to apply color filters will be discussed later in this topic.

Features of Wireshark

- It is multi-platform software, i.e., it can run on Linux, Windows, OS X, FreeBSD, NetBSD, etc.
- It is a standard three-pane packet browser.
- It performs deep inspection of the hundreds of protocols.
- It often involves live analysis, i.e., from the different types of the network like the Ethernet, loopback, etc., we can read live data.
- It has sort and filter options which makes ease to the user to view the data.
- It is also useful in VoIP analysis.
- It can also capture raw USB traffic.
- Various settings, like timers and filters, can be used to filter the output.

- It can only capture packet on the PCAP (an application programming interface used to capture the network) supported networks.
- Wireshark supports a variety of well-documented capture file formats such as the PcapNg and Libpcap. These formats are used for storing the captured data.
- It is the no.1 piece of software for its purpose. It has countless applications ranging from the **tracing down, unauthorized traffic, firewall settings, etc**.

Installation of Wireshark Software

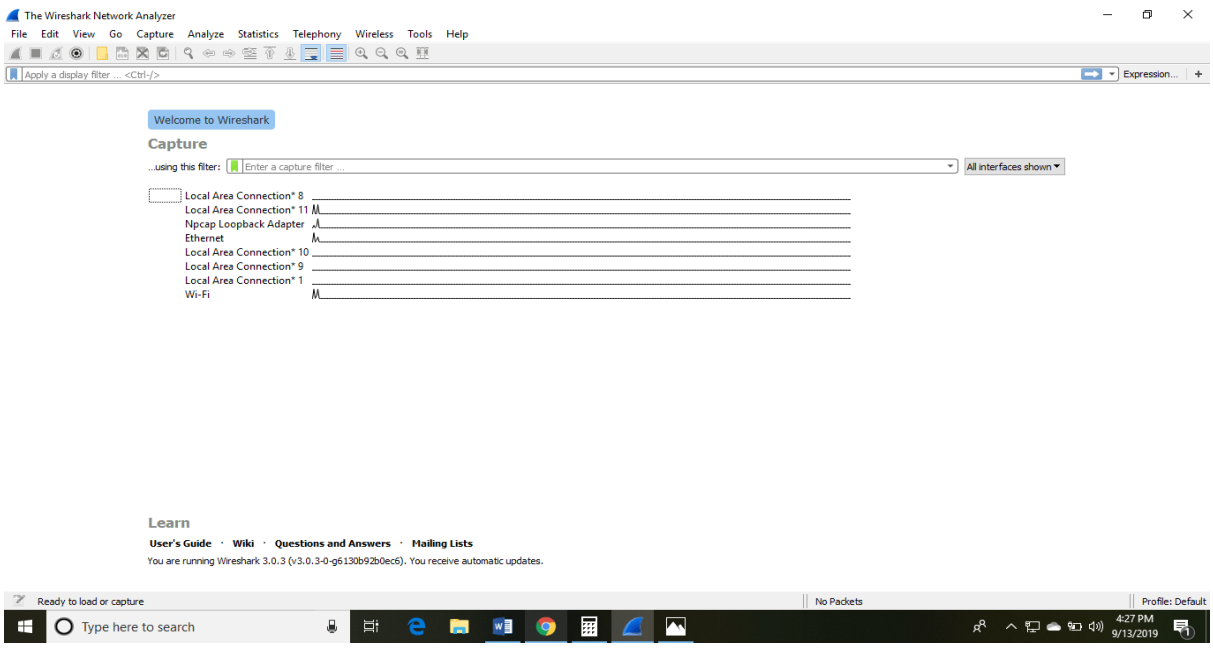
Below are the steps to install the Wireshark software on the computer:

- Open the web browser.
- Search for '**Download Wireshark**.'
- Select the Windows installer according to your system configuration, either 32-bt or 64-bit. Save the program and close the browser.
- Now, open the software, and follow the install instruction by accepting the license.
- The Wireshark is ready for use.

On the network and Internet settings option, we can check the interface connected to our computer.

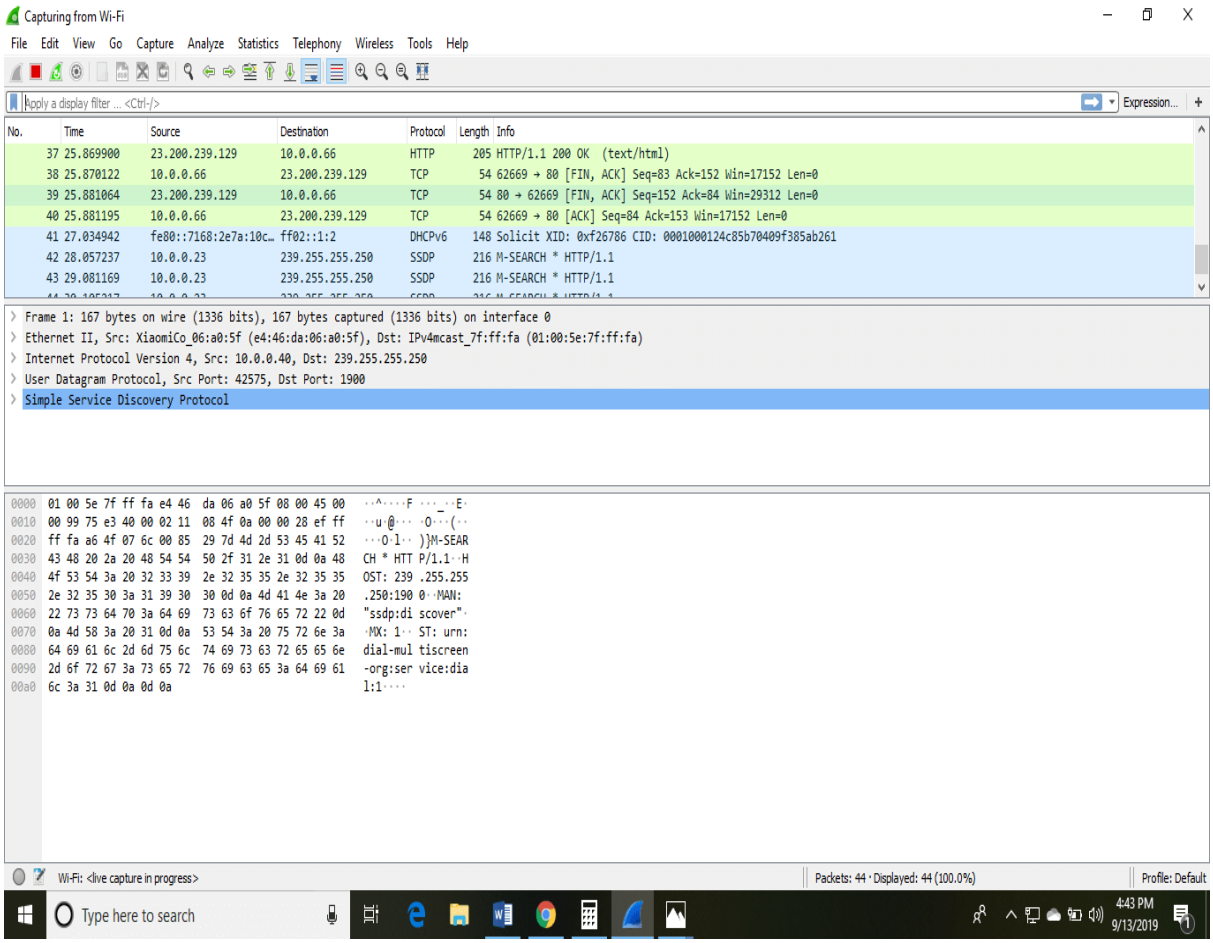
If you are Linux users, then you will find Wireshark in its package repositories.

By selecting the current interface, we can get the traffic traversing through that interface. The version used here is **3.0.3**. This version will open as:



The Wireshark software window is shown above, and all the processes on the network are carried within this screen only.

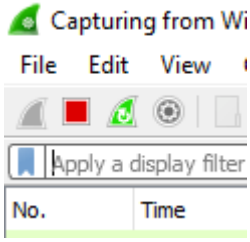
The options given on the list are the Interface list options. The number of interface options will be present. Selection of any option will determine all the traffic. **For example,** from the above fig. select the Wi-Fi option. After this, a new window opens up, which will show all the current traffic on the network. Below is the image which tells us about the live capture of packets and our Wireshark will look like:



The above arrow shows the packet content written in hexadecimal or the ASCII format. And the information above the packet content, are the details of the packet header.

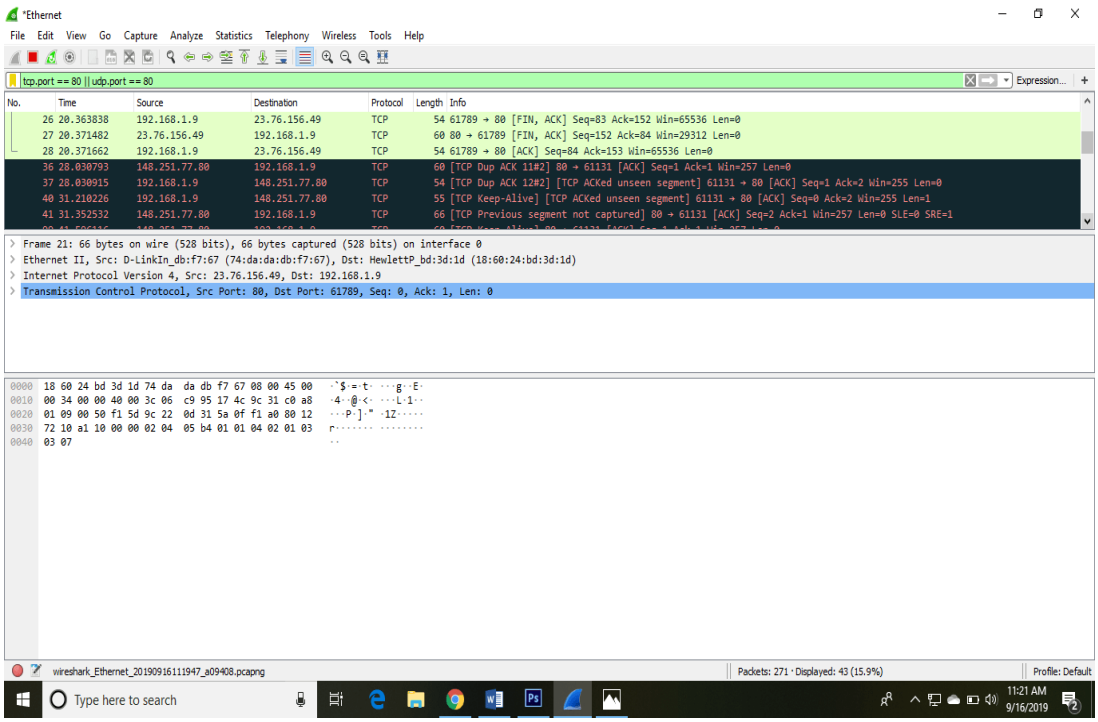
It will continue listening to all the data packets, and you will get much data. If you want to see a particular data, then you can click on the red button. The traffic will be stationary, and you can note the parameters like time, source, destination, the protocol being used, length, and the Info. To view in-depth detail, you can click on that particular address; a lot of the information will be displayed below that.

There will be detailed information on HTTP packets, TCP packets, etc. The red button is shown below:



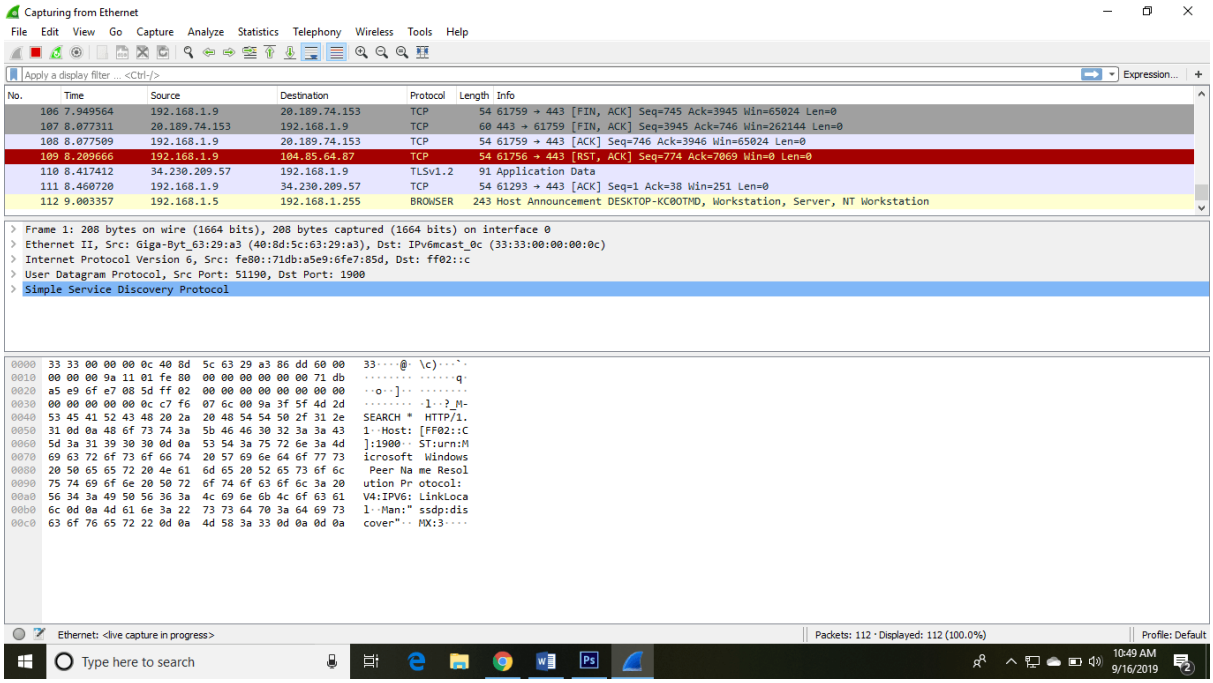
The screen/interface of the Wireshark is divided into five parts:

- First part contains a menu bar and the options displayed below it. This part is at the top of the window. File and the capture menus options are commonly used in Wireshark. The capture menu allows to start the capturing process. And the File menu is used to open and save a capture file.
- The second part is the packet listing window. It determines the packet flow or the captured packets in the traffic. It includes the packet number, time, source, destination, protocol, length, and info. We can sort the packet list by clicking on the column name.
- Next comes the packet header- detailed window. It contains detailed information about the components of the packets. The protocol info can also be expanded or minimized according to the information required.
- The bottom window called the packet contents window, which displays the content in ASCII and hexadecimal format.
- At last, is the filter field which is at the top of the display. The captured packets on the screen can be filtered based on any component according to your requirements. For example, if we want to see only the packets with the HTTP protocol, we can apply filters to that option. All the packets with HTTP as the protocol will only be displayed on the screen, shown below:

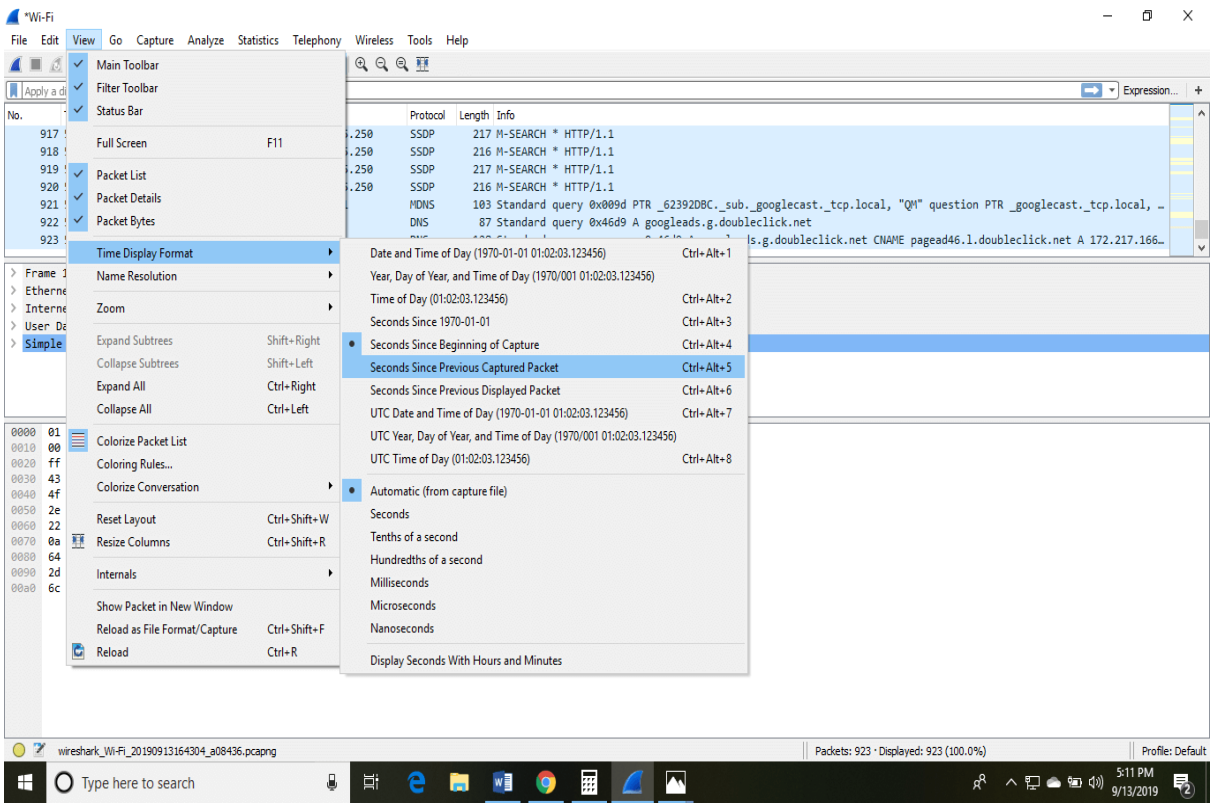


You can also select the connection to which your computer is connected. For example, in this PC, we have chosen the current network, i.e., the ETHERNET.

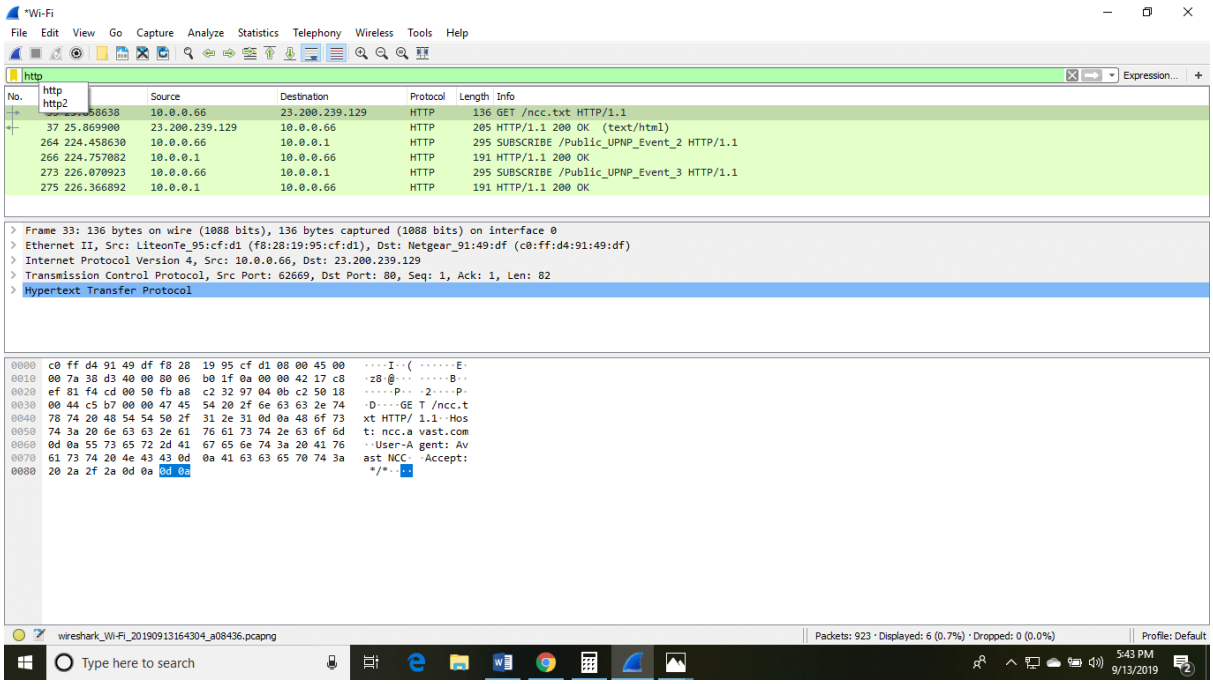
After connecting, you can watch the traffic below:



In view option on the menu bar, we can also change the view of the interface. You can change the number of things in the view menu. You can also enable or disable any option according to the requirements.

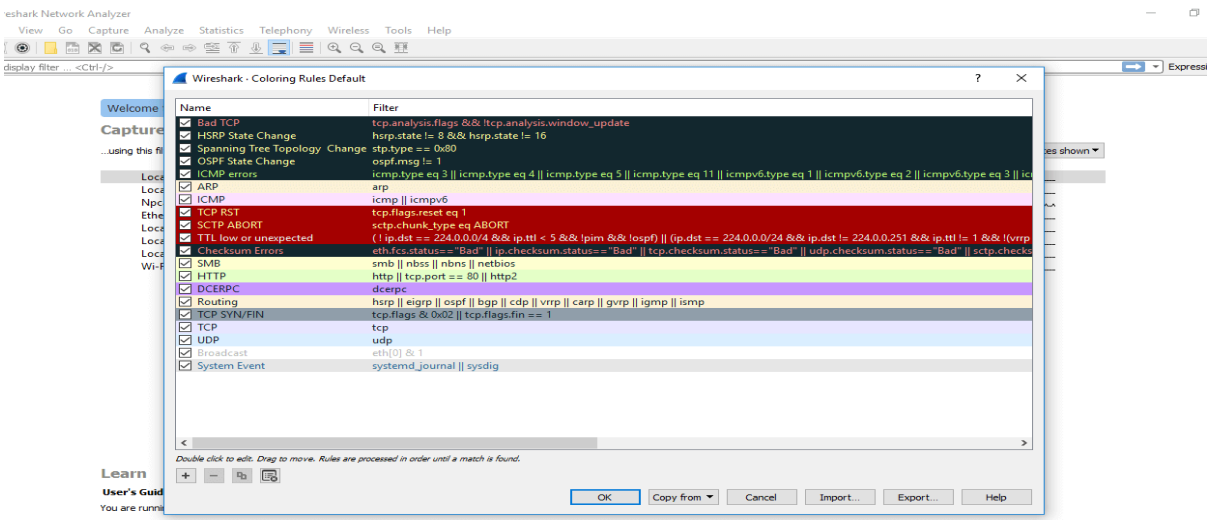


There is a filter block below the menu bar, from where a large amount of data can be filtered. For example, if we apply a filter for HTTP, only the interfaces with the HTTP will be listed.



If you want to filter according to the source, right-click on the source you want to filter and select 'Apply as Filter' and choose '...and filter.'

Steps for the permanent colorization are: click on the 'View' option on the menu bar and select 'Coloring Rules.' The table will appear like the image shown below:



For the network administrator job, advanced knowledge of Wireshark is considered as the requirements. So, it is essential to understand the concepts of the software. It contains these 20 default coloring rules which can be added or removed according to the requirements.

Select the option '**View**' and then choose '**Colorize Packet List**,' which is used to **toggle the color on and off**.

Note: If you are not sure about the version of your desktop or the laptop, then you can download the 32-bit Wireshark which will run almost 99% on every type of computers

Now let's start with this basics-

Basic concepts of the Network Traffic

IP Addresses: It was designed for the devices to communicate with each other on a local network or over the Internet. It is used for host or network interface identification. It provides the location of the host and capacity of establishing the path to the host in that network. Internet Protocol is the set of predefined rules or terms under which the communication should be conducted. The types of IP addresses are **IPv4 and IPv6**.

- IPv4 is a **32-bit address** in which each group represents 8 bits ranging from 0 to 255.
- IPv6 is a 128-bit address.

IP addresses are assigned to the host either dynamically or static IP address. Most of the private users have dynamic IP address while business users or servers have a static IP address. Dynamic address changes whenever the device is connected to the Internet.

Computer Ports: The computer ports work in combination with the IP address directing all outgoing and incoming packets to their proper places. There are well-known ports to work with like **FTP** (File Transfer Protocol), which has port no. 21, etc. All the ports have the purpose of directing all packets in the predefined direction.

Protocol: The Protocol is a set of predefined rules. They are considered as the standardized way of communication. One of the most used protocol is **TCP/IP**. It stands for **Transmission Control Protocol/ Internet Protocol**.

OSI model: OSI model stands for **Open System Interconnect**. OSI model has seven layers, namely, **Application layer, Presentation layer, Session layer, Transport layer, Network layer, Data link layer, and the physical layer**. OSI model gives a detail representation and explanation of the transmission and reception of data through the layers. OSI model supports both connectionless and connection-oriented communication mode over the network layer. The OSI model was developed by ISO (International Standard Organization).

Most used Filters in Wireshark

Whenever we type any commands in the filter command box, it turns **green** if your command is **correct**. It turns **red** if it is **incorrect** or the Wireshark does not recognize your command.

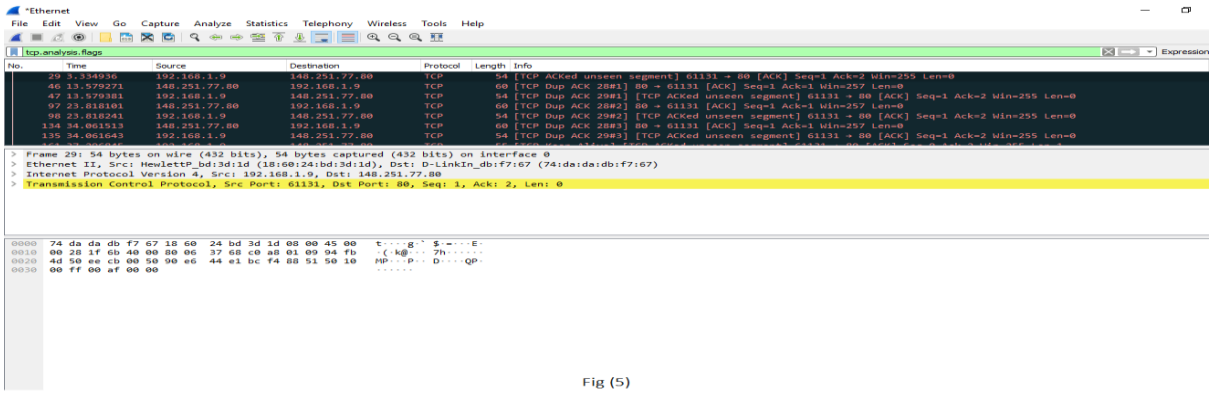

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.analysis.flags
No. Time Source Destination Protocol Length Info
29 3.334936 192.168.1.9 148.251.77.80 TCP 54 [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
46 13.579271 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#1] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
47 13.579381 192.168.1.9 148.251.77.80 TCP 54 [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
97 23.818101 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#2] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
98 23.818241 192.168.1.9 148.251.77.80 TCP 54 [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
134 34.061513 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#3] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
135 34.061643 192.168.1.9 148.251.77.80 TCP 54 [TCP Dup ACK 29#3] [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
Frame 29: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: HewlettP_bd:3d:1d (18:60:24:bd:3d:1d), Dst: D-LinkIn_db:f7:67 (74:da:da:db:f7:67)
Internet Protocol Version 4, Src: 192.168.1.9, Dst: 148.251.77.80
Transmission Control Protocol, Src Port: 61131, Dst Port: 80, Seq: 1, Ack: 2, Len: 0
0000 74 da da db f7 67 18 60 24 bd 3d 1d 08 00 45 00
0010 00 28 1f 6b 40 00 80 06 37 68 c0 a8 01 09 94 fb
0020 4d 50 ee cb 00 50 90 e6 44 e1 bc f4 88 51 50 10
0030 00 ff 00 af 00 00

Fig (5)

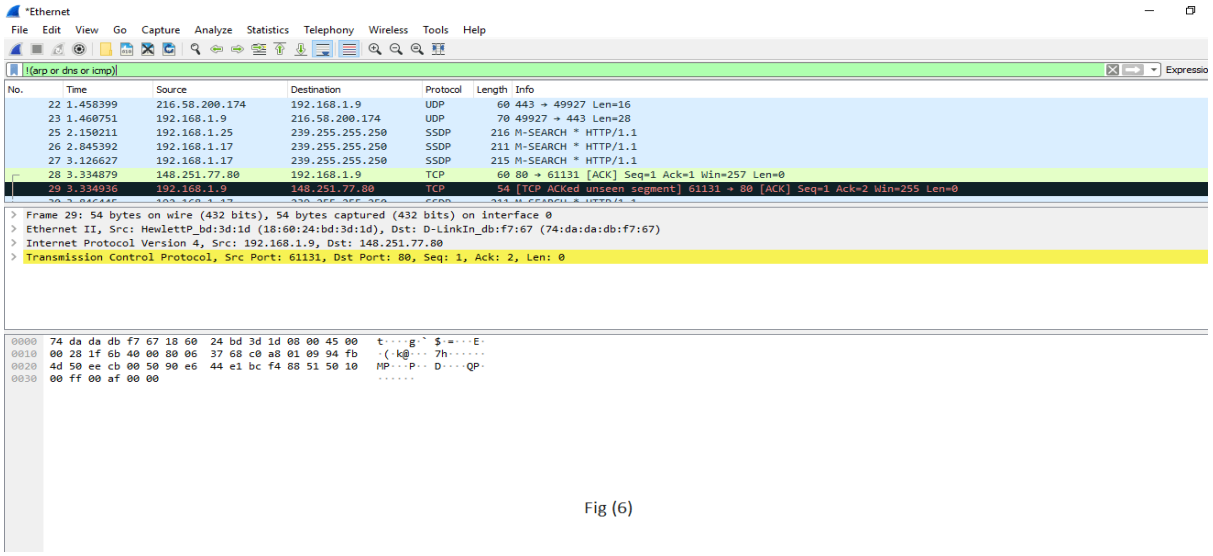

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
!(arp or dns or icmp)
No. Time Source Destination Protocol Length Info
22 1.458399 216.58.200.174 192.168.1.9 UDP 60 443 → 49927 Len=16
23 1.460751 192.168.1.9 216.58.200.174 UDP 70 49927 → 443 Len=28
25 2.150211 192.168.1.25 239.255.255.250 SSDP 216 M-SEARCH * HTTP/1.1
26 2.845392 192.168.1.17 239.255.255.250 SSDP 211 M-SEARCH * HTTP/1.1
27 3.126627 192.168.1.17 239.255.255.250 SSDP 215 M-SEARCH * HTTP/1.1
28 3.334879 148.251.77.80 192.168.1.9 TCP 60 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
29 3.334936 192.168.1.9 148.251.77.80 TCP 54 [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
Frame 29: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: HewlettP_bd:3d:1d (18:60:24:bd:3d:1d), Dst: D-LinkIn_db:f7:67 (74:da:da:db:f7:67)
Internet Protocol Version 4, Src: 192.168.1.9, Dst: 148.251.77.80
Transmission Control Protocol, Src Port: 61131, Dst Port: 80, Seq: 1, Ack: 2, Len: 0
0000 74 da da db f7 67 18 60 24 bd 3d 1d 08 00 45 00
0010 00 28 1f 6b 40 00 80 06 37 68 c0 a8 01 09 94 fb
0020 4d 50 ee cb 00 50 90 e6 44 e1 bc f4 88 51 50 10
0030 00 ff 00 af 00 00

Fig (6)

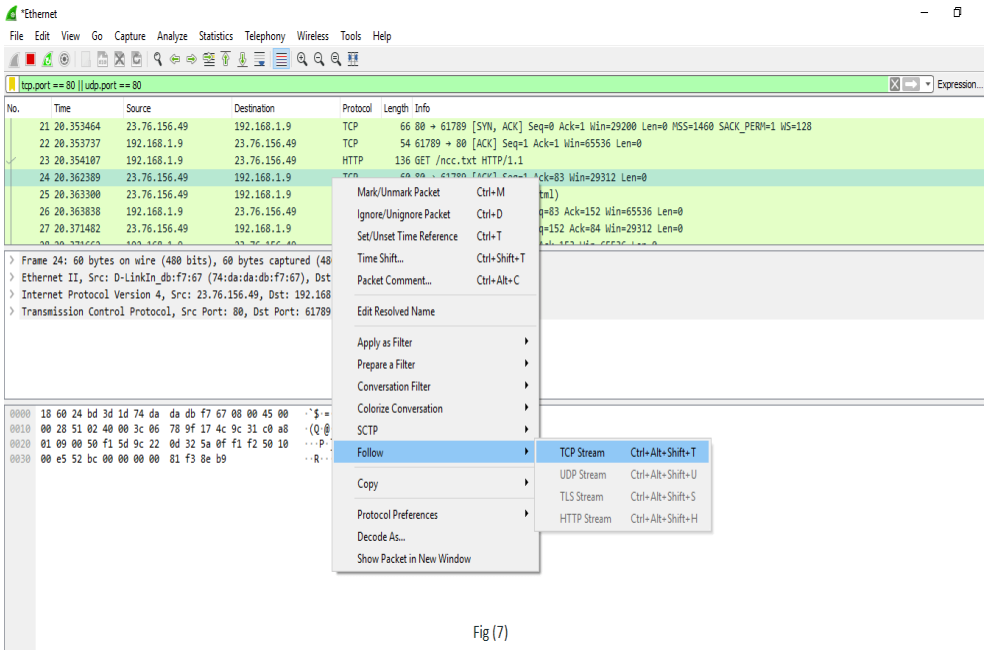


Fig (7)

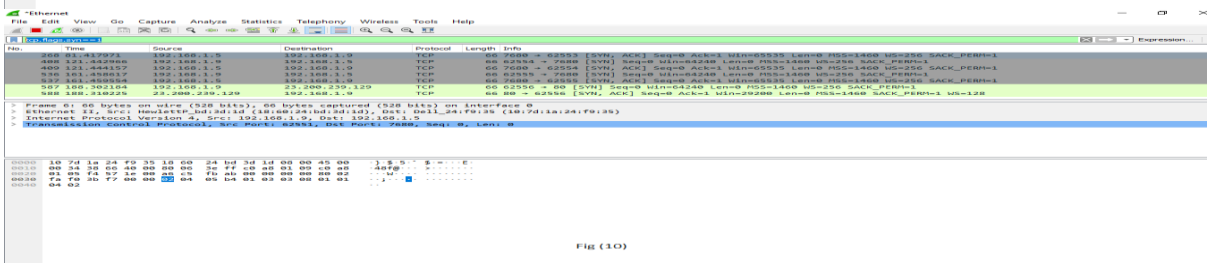


Fig (10)

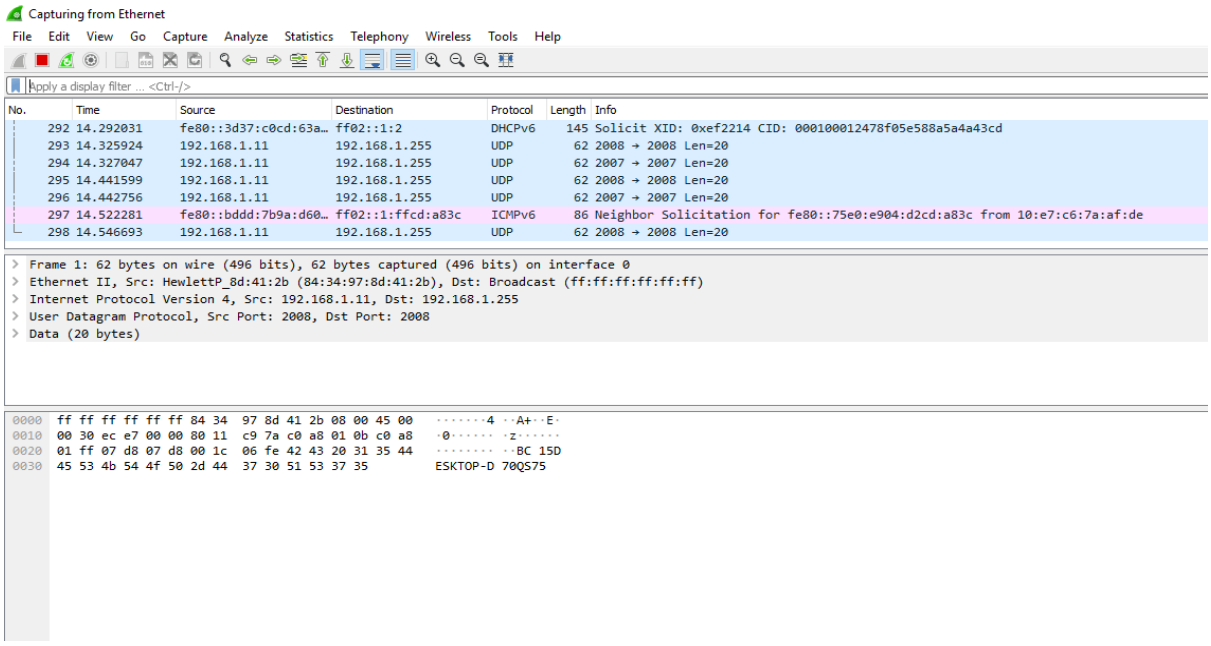
Wireshark packet sniffing

Wireshark is a packet sniffing program that administrators can use to isolate and troubleshoot problems on the network. It can also be used to capture sensitive data like usernames and passwords. It can also be used in wrong way (hacking) to ease drop.

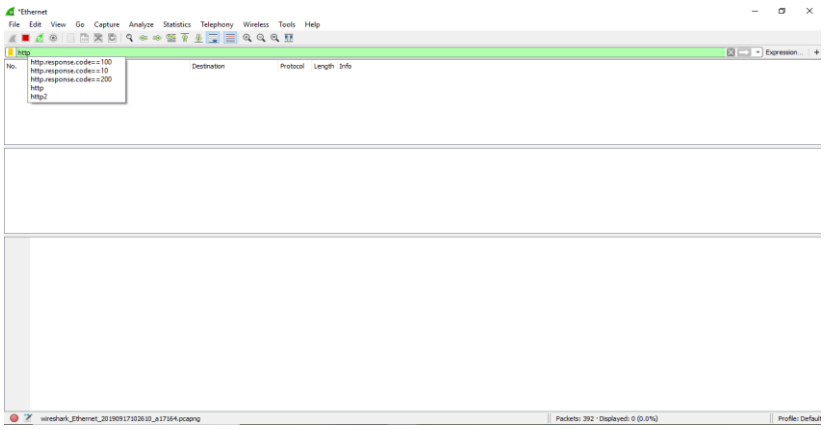
Packet sniffing is defined as the process to capture the packets of data flowing across a computer network. The Packet sniffer is a device or software used for the process of sniffing.

Below are the steps for packet sniffing:

- Open the Wireshark Application.
- Select the current interface. Here in this example, interface is Ethernet that we would be using.
- The network traffic will be shown below, which will be continuous. To stop or watch any particular packet, you can press the red button below the menu bar.



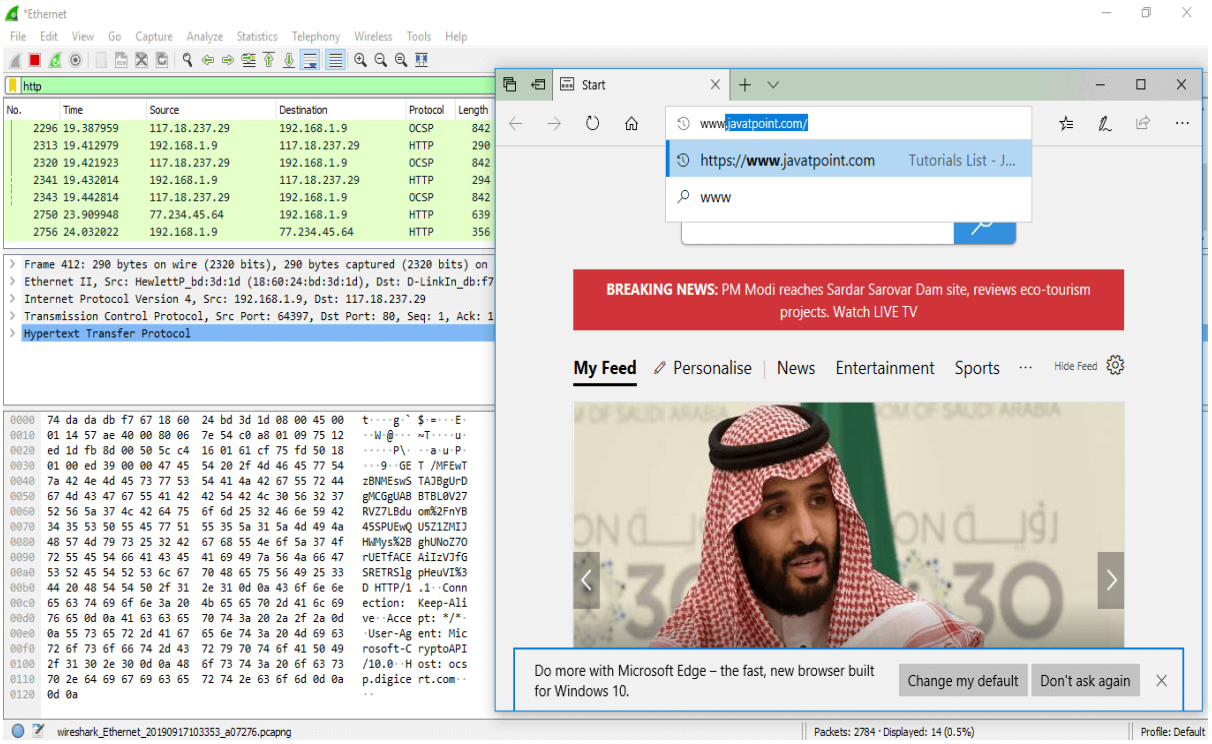
Apply the filter by the name 'http.' After the filter is applied, the screen will look as:

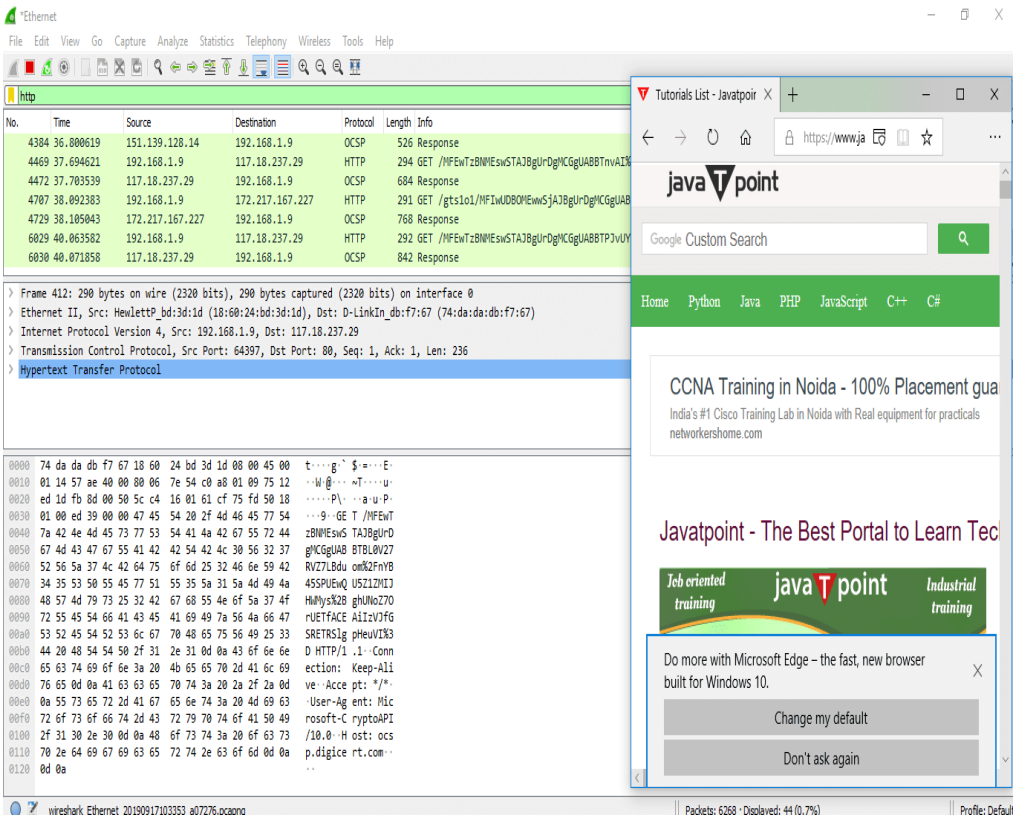


The above screen is blank, i.e.; there is no network traffic as of now.

Open the browser. In this example, we have opened the 'Internet Explorer.' You can choose any browser.

As soon as we open the browser, and type any address of the website, the traffic will start showing, and exchange of the packets will also start. The image for this is shown below:



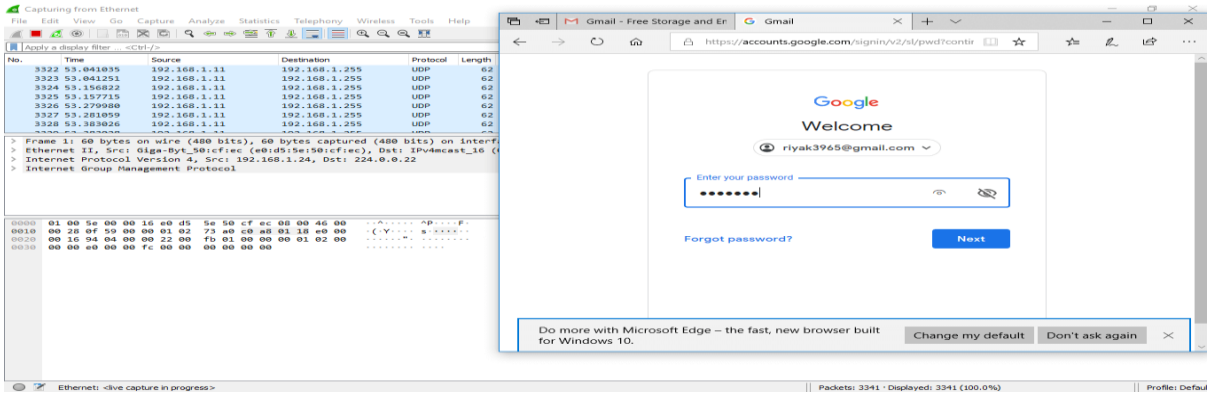


The above process explained is called as **packet sniffing**.

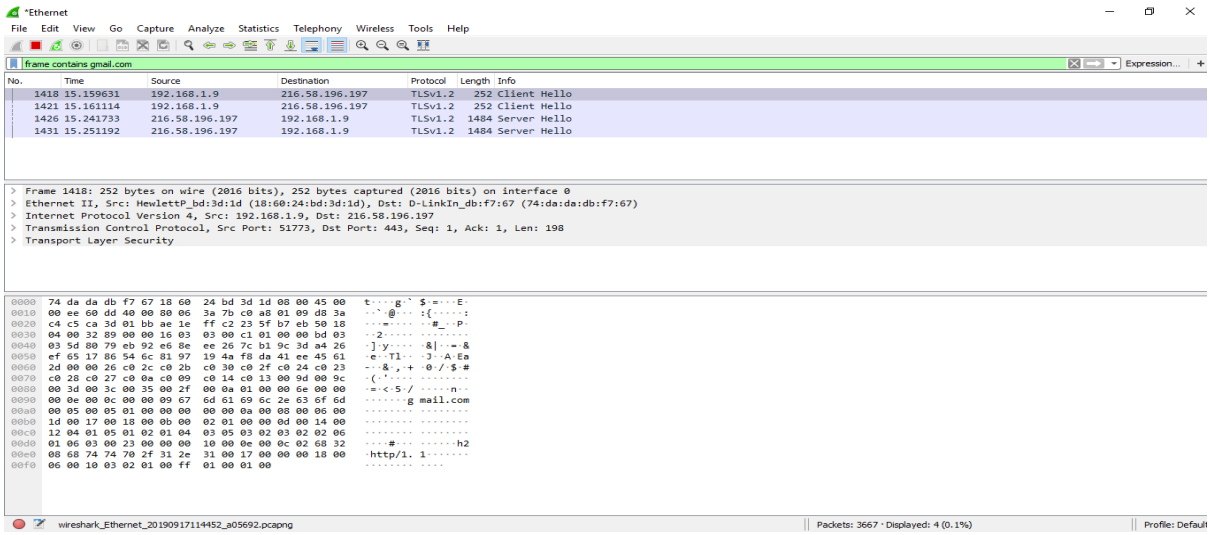
Username and password sniffing

It is the process used to know the passwords and username for the particular website. Let's take an example of gmail.com. Below are the steps:

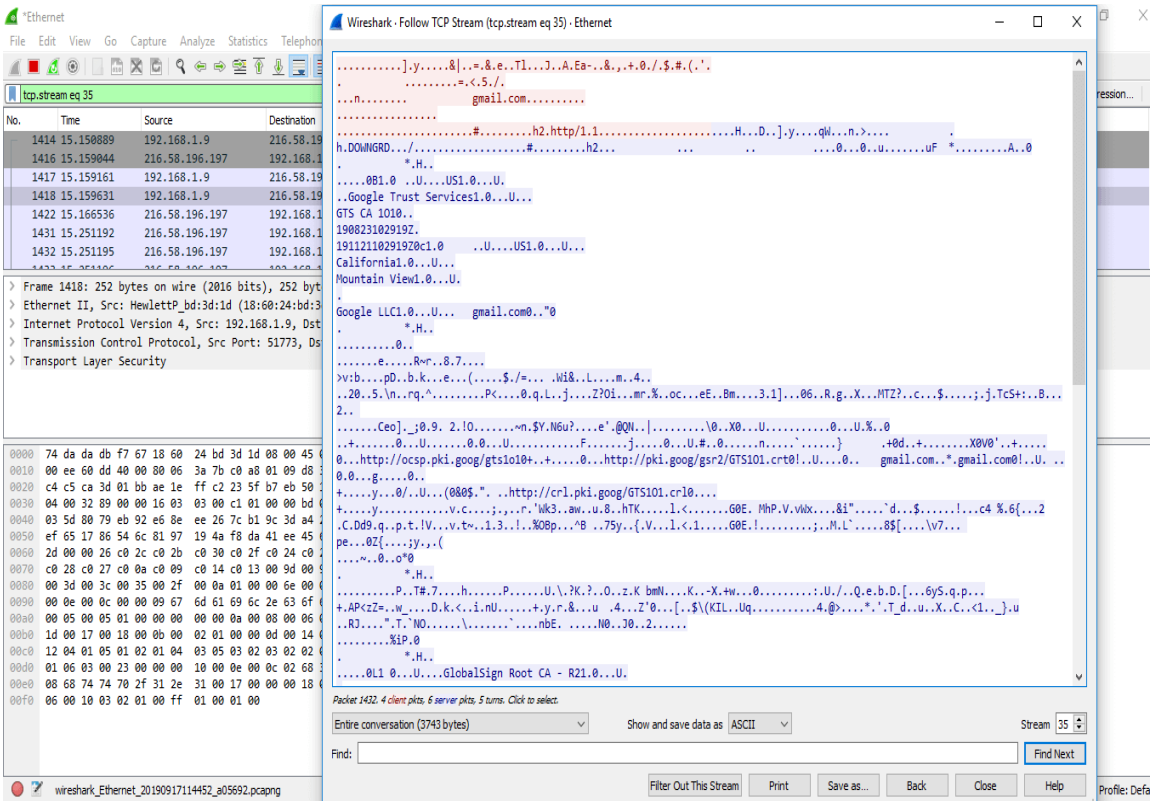
- Open the Wireshark and select the suitable interface.
- Open the browser and enter the web address. Here, we have entered gmail.com, which is highly secured. Enter your email address and the password. The image is shown below:



- Now, go to the Wireshark and on the filters block, enter 'frame contains gmail.com.' Then you can see some traffic.



- Right-click on the particular network and select 'Follow', and then 'TCP Stream.' You can see that all the data is secured in the encrypted form.



In the arrow shown above, the 'show and save data as' has many choices. These options are- **ASCII, C Arrays, EBCDIC (Extended Binary Coded Decimal Interchange Code)**, etc. EBCDIC is used in mainframe and mid-range IBM computer operating systems.

Wireshark Statistics

The Wireshark provides a wide domain of statistics. They are listed below:

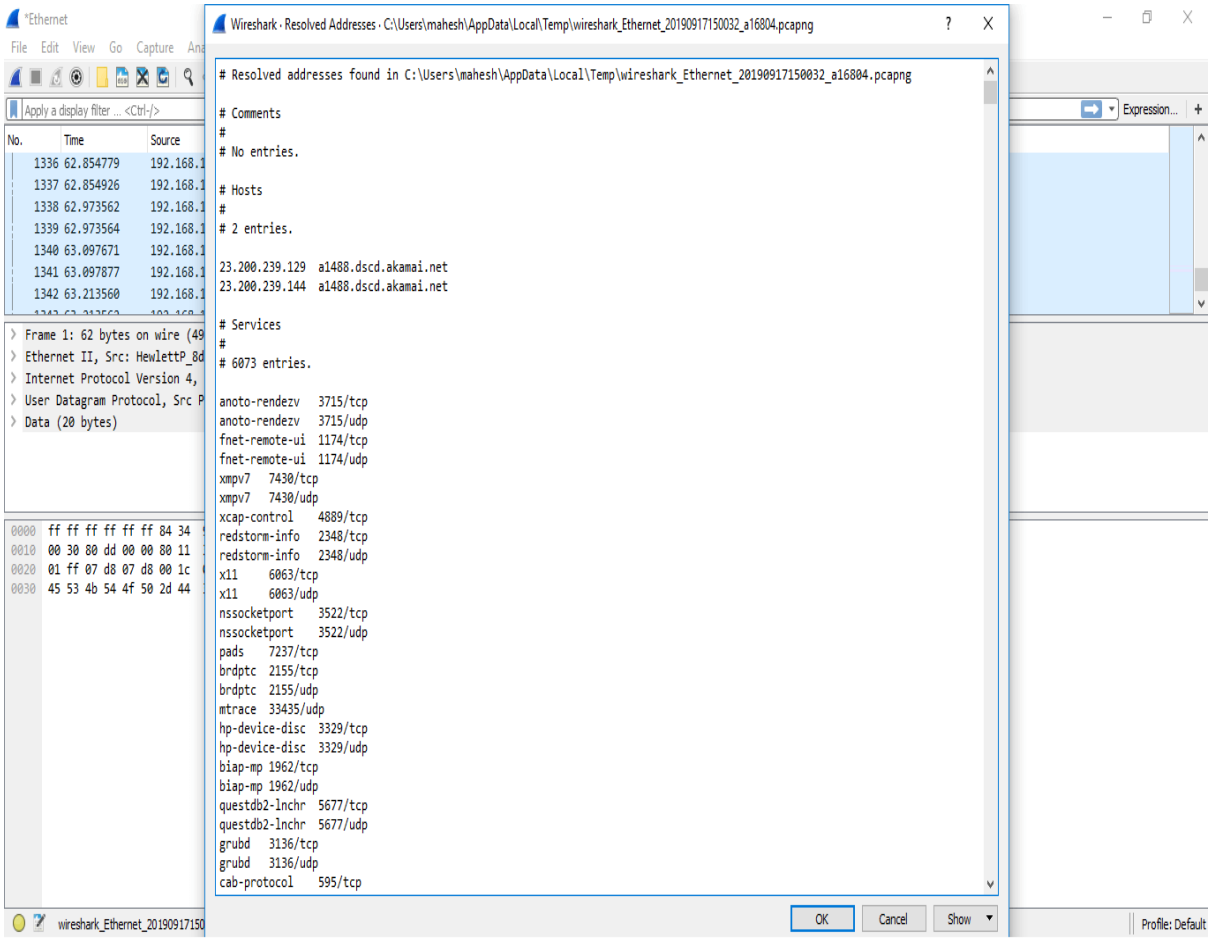

*Ethernet
File Edit View Go Capture
Apply a display filter ... <Ctrl-/>
No. Time Source
1336 62.854779 192.168.1
1337 62.854926 192.168.1
1338 62.973562 192.168.1
1339 62.973564 192.168.1
1340 63.097671 192.168.1
1341 63.097877 192.168.1
1342 63.213560 192.168.1
Frame 1: 62 bytes on wire (49
Ethernet II, Src: HewlettP_8d
Internet Protocol Version 4,
User Datagram Protocol, Src P
Data (20 bytes)
0000 ff ff ff ff ff ff 84 34
0010 00 30 80 dd 00 00 80 11
0020 01 ff 07 d8 07 d8 00 1c
0030 45 53 4b 54 4f 50 2d 44
wireshark_Ethernet_20190917150
Expression...
Profile: Default
Wireshark · Resolved Addresses · C:\Users\mahesh\AppData\Local\Temp\wireshark_Ethernet_20190917150032_a16804.pcapng
Resolved addresses found in C:\Users\mahesh\AppData\Local\Temp\wireshark_Ethernet_20190917150032_a16804.pcapng
Comments
#
No entries.
Hosts
#
2 entries.
23.200.239.129 a1488.dscd.akamai.net
23.200.239.144 a1488.dscd.akamai.net
Services
#
6073 entries.
anoto-rendezv 3715/tcp
anoto-rendezv 3715/udp
fnet-remote-ui 1174/tcp
fnet-remote-ui 1174/udp
xmpv7 7430/tcp
xmpv7 7430/udp
xcap-control 4889/tcp
redstorm-info 2348/tcp
redstorm-info 2348/udp
x11 6063/tcp
x11 6063/udp
nssocketport 3522/tcp
nssocketport 3522/udp
pads 7237/tcp
brdptc 2155/tcp
brdptc 2155/udp
mtrace 33435/udp
hp-device-disc 3329/tcp
hp-device-disc 3329/udp
biap-mp 1962/tcp
biap-mp 1962/udp
questdb2-lnchr 5677/tcp
questdb2-lnchr 5677/udp
grubd 3136/tcp
grubd 3136/udp
cab-protocol 595/tcp
OK
Cancel
Show

Fig (b)

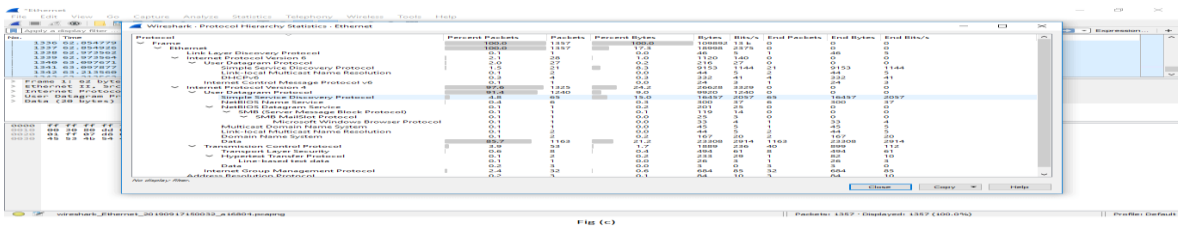


Fig (c)

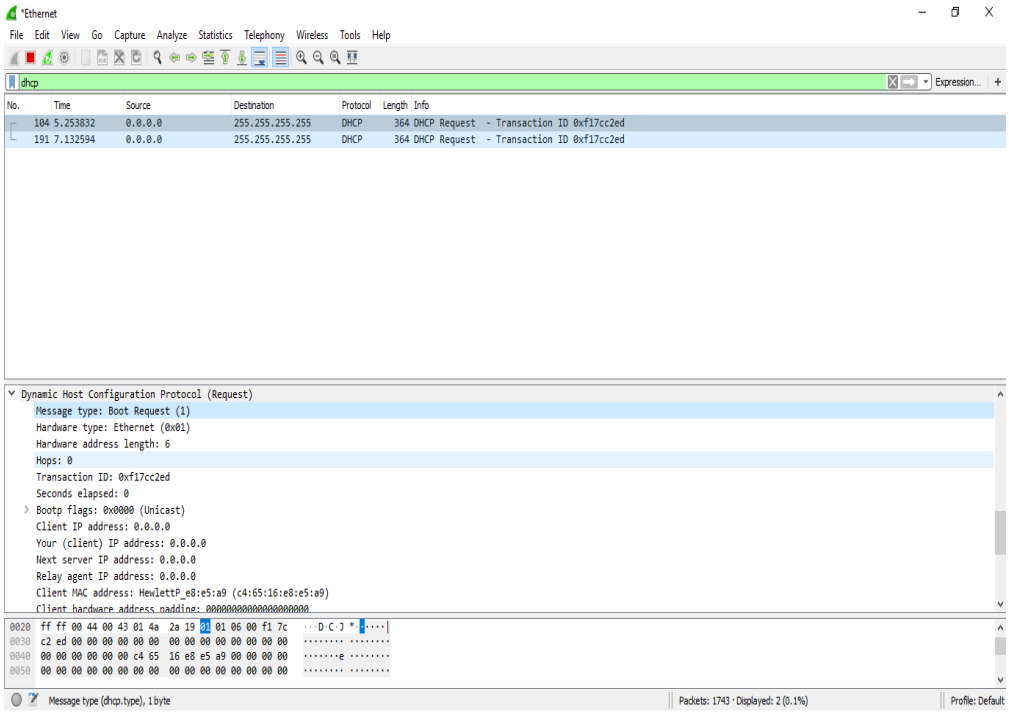
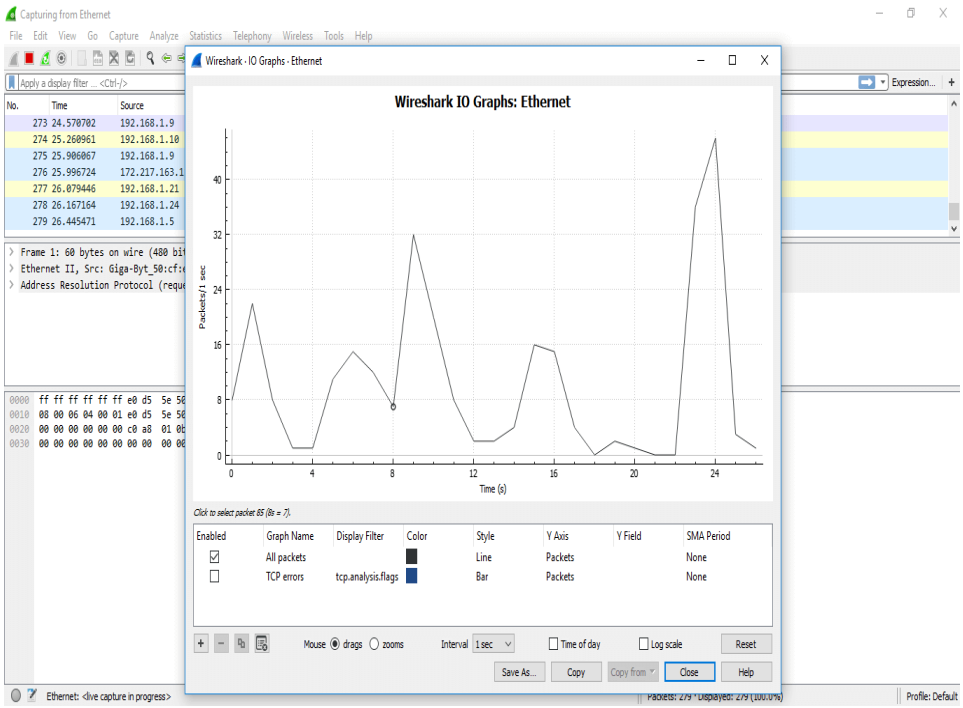


Fig (d)

Below is the list of statistics of Wireshark along with the description:

I/O GRAPHS

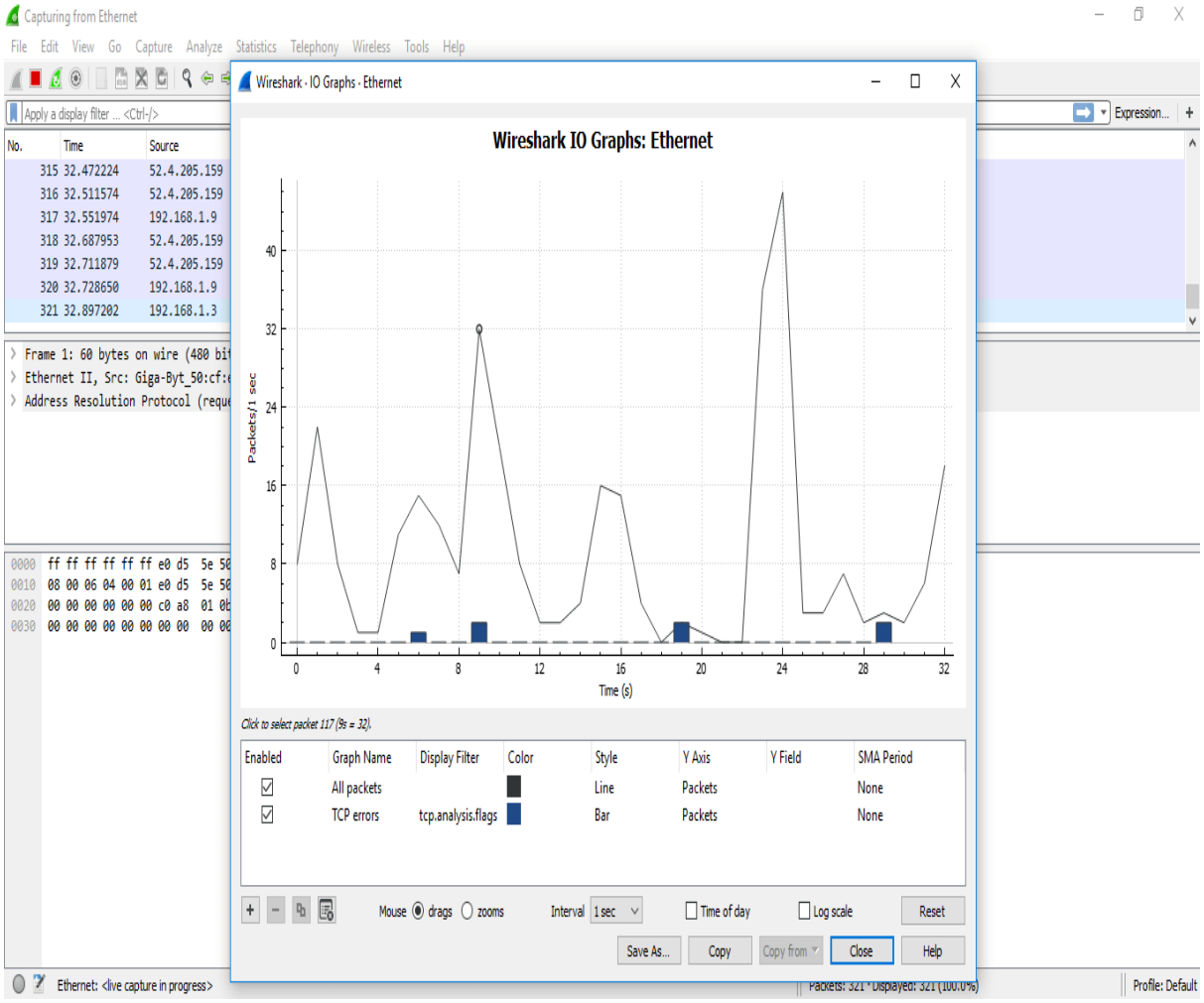


It shows the graph for the network traffic. The graph will look similar but changes as per the traffic involved. There is a table below the figure, which has some filters. Using the '+' sign, you can add more filters and use '-sign you can remove the existing filters. You can also

change the color. For every particular filter, you can add a colored layer, which increases the visibility of the graph.

The tick option under the 'Enabled,' displays the layer according to your requirements.

For example, we have applied the filter 'TCP errors' and the changes can be viewed easily. The image is shown below:



If you click on the particular point on the graph, you can watch the corresponding packet will be shown on the screen of the network traffic. You can also apply a filter on the particular port.

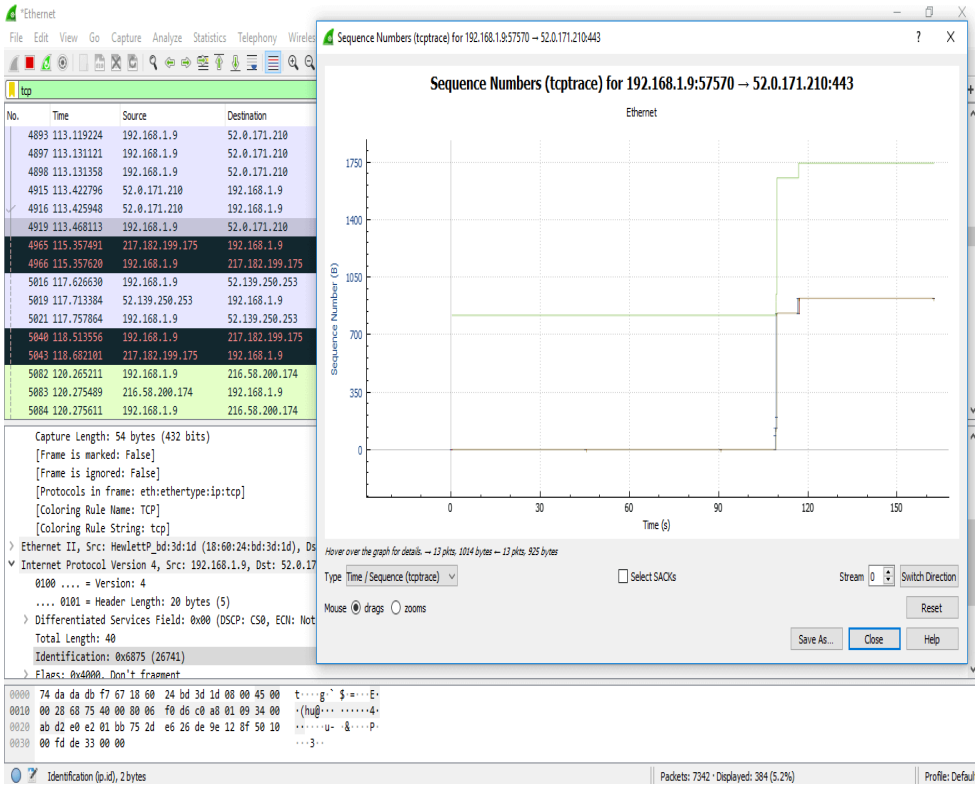
Another category of the graph comes under the option '**TCP Stream graphs**.'

It gives the visualization of the TCP sequence number with time.

Below are the steps to understand the **TCP Stream graphs**:

- Open the Wireshark. Click on the interface to watch the network traffic.
- Apply the filter as 'tcp.'

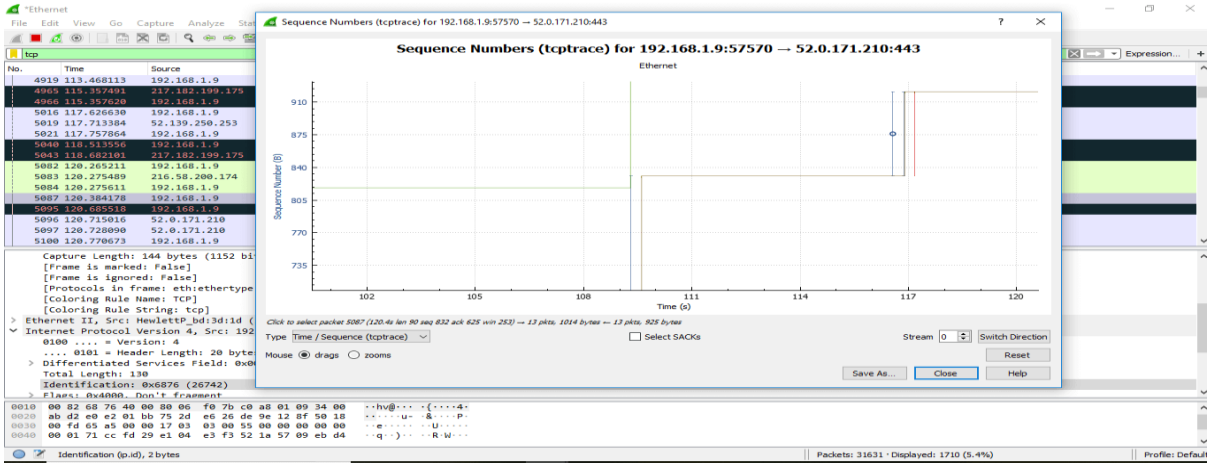
- Click on the option 'Statistics 'on the menu bar and select '**TCP Stream graphs**' and select 'Time sequence (tcptrace). You can also choose other options in the 'TCP Stream graphs' category depending on your requirements. Now the screen will look as:



Now, as you zoom on the graph, you will notice the points in detail. The lines shown are the packets. The length along the Y-axis shows how big the packet is. You can also see the green line going up and then comes at the same level. This means that the data has been ACK (Acknowledged). Here going up means that more data is being sent.

The data is being sent and then ACK, this is the proper use of the TCP. The flat line here signifies that nothing is happening.

The green line above is called '**received window**.' The gap between the received window and the packet, defines how much space is in the received buffer.



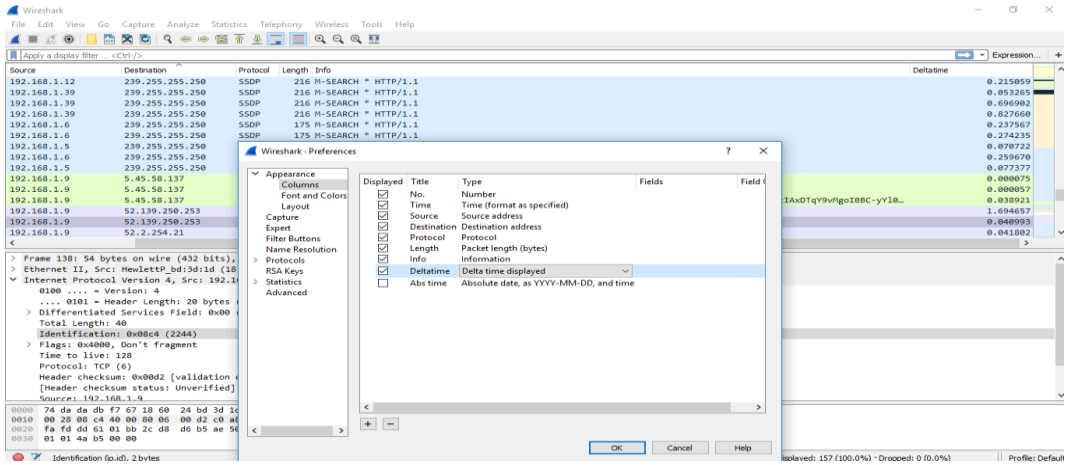
FACTS ABOUT WIRESHARK/ IMPORTANT STEPS/ MOST USED

Below are the facts or points implemented in real life:

Adding a delta column: To add any column, below are the steps:

- On any of the column menu, right-click and choose 'Column Preferences' and then select 'Column.'
- Click on the '+' sign, and add the column by name like delta-time and under the 'Type' category, select the delta time or delta time displayed.

The screen will then look as:

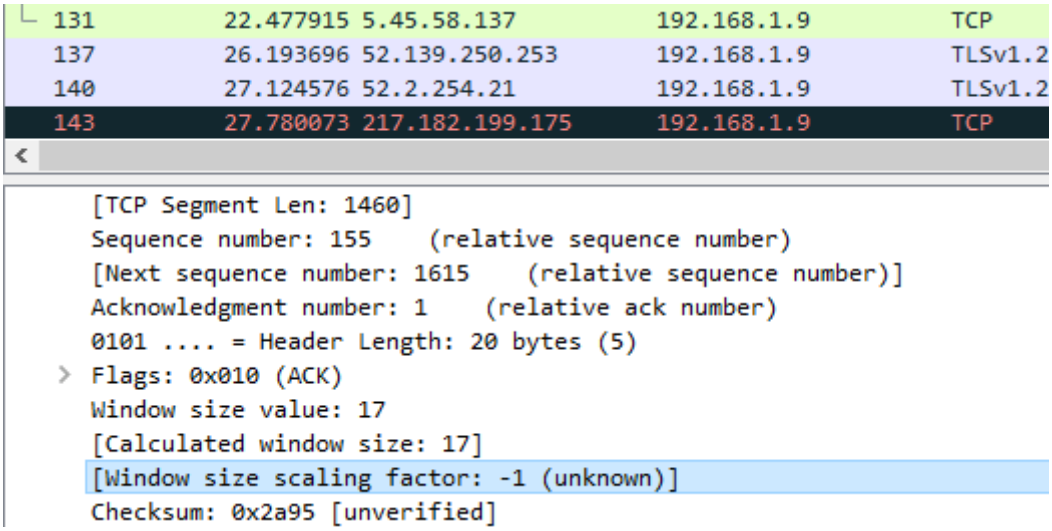


Below the captured packets, the data you see in the **square brackets** is the information that is not available in the packet itself. It is something that Wireshark displays for your benefit. If you want to add anything from this screen to the column area, you can right-click and select 'Apply as column.' That option will be added to the capture screen.

The most important is:

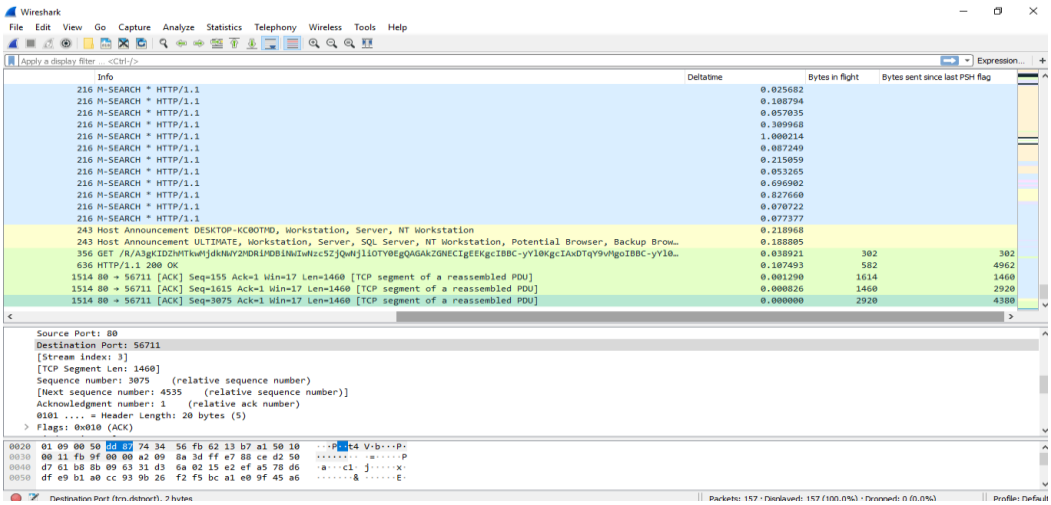
3 Way-Handshake

- When you are capturing your data, analyze the problem, you will get the three-way handshake.
- It contains good options like the TCP options.
- From this, you can determine the shift time and figure out if you have captured packets on the client-side or the server-side. There is a little delay between SYN and SYN- ACK packet at server-side while there is a more delay between the SYN and SYN-ACK at the client-side. There is a delay at the server-side only between the SYN-ACK and ACK. The SYN has to reach to the client. After the three-way handshake, the data has to reach the server.
- ou can also notice the difference in the TCP options between the SYN and SYN-ACK packets. The window scaling factor is also essential, as shown below:

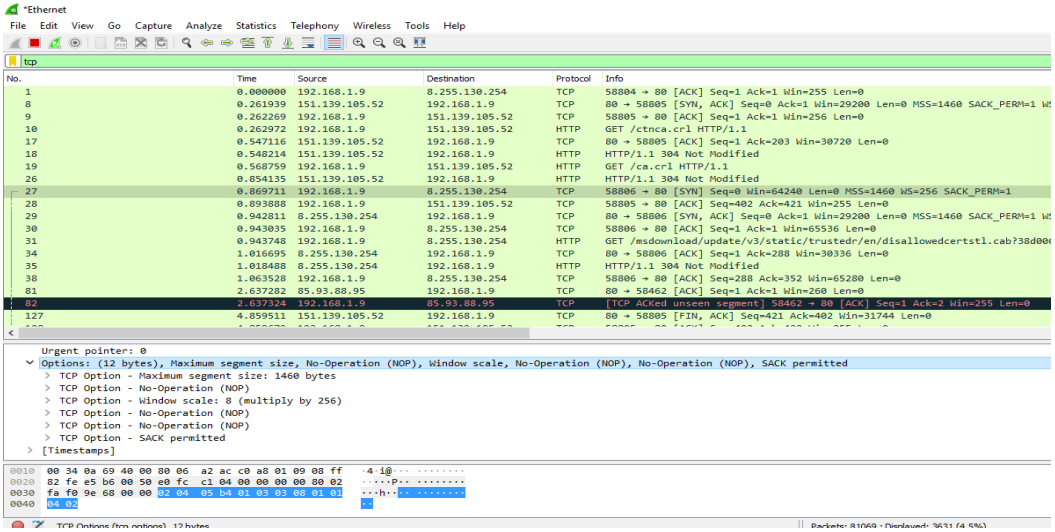


Without three-way handshake, you cannot view the window scaling factor.

- One sequence number means 1 byte of data. It also has an importance of the TCP Stream Graphs which is already explained above.
- Under the TCP options, capture window, you can see the information about the 'PSH byte' and 'Bytes in flight.' Right-click on that and choose 'Apply as Column.' You can see both the columns and data according to it. The image for this is shown below:



- In TCP Header, three-way handshake MSS (Maximum Header Size) means that the maximum amount of data it can receive of TCP payload. The image is shown below:



- MSS 1460 implies that this is per packet amount of data. This size varies from packet to packet. Something like a router, firewall, etc. will do MSS clamping because it knows what is going forward. It checks the value greater than 8000 bytes and brings down it to an appropriate level so that it can go across without fragmentation or being dropped.

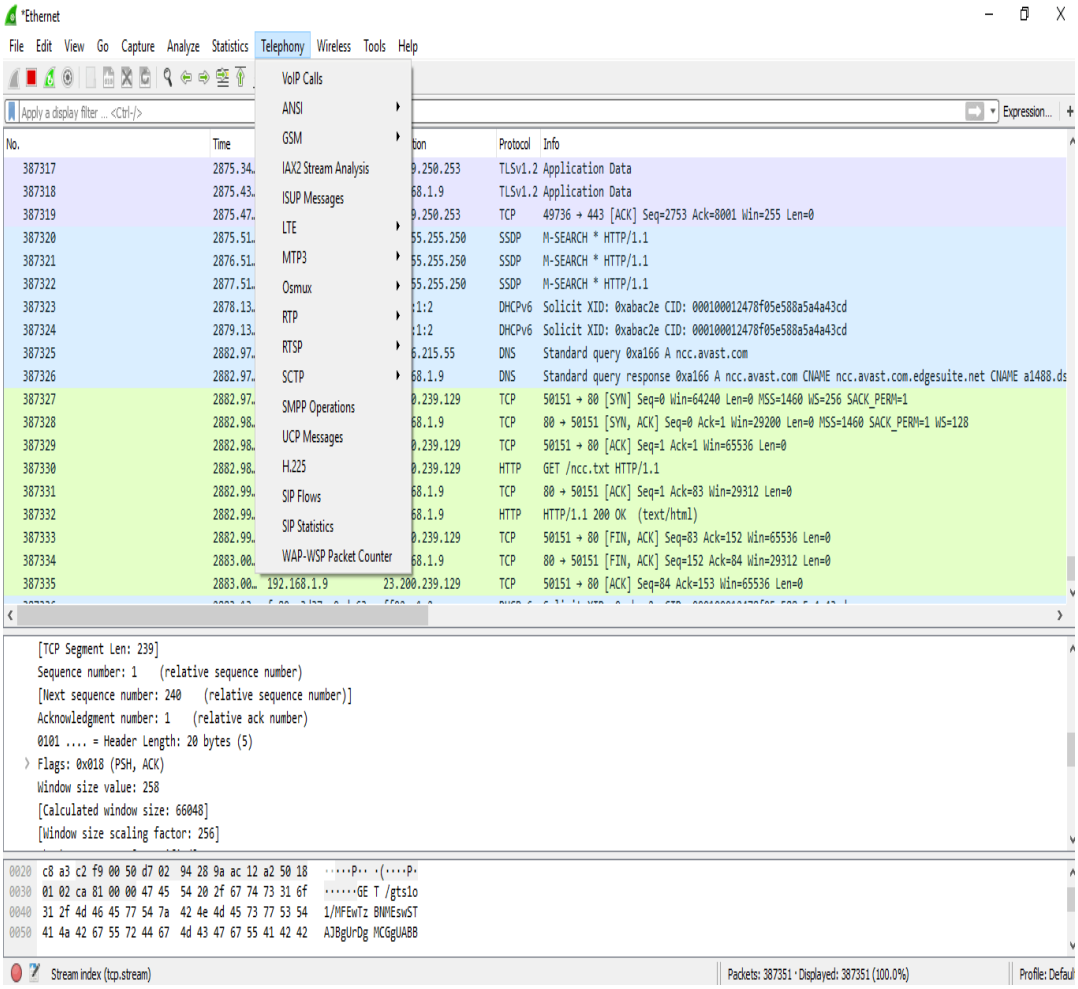
Some Facts about Wireshark:

- We do recommend not to disable the default settings of the TCP and Wireshark unless you know what you?re doing.
- If there are the blank page and slow loading, then it is unusable.
- It is good to capture packets from both ends.
- Lean on your provider when you have the data.
- It is a LIVE CAPTURE software used widely.
- It can also capture packets from a set of captured one's.
- There are many protocols dissectors.

- The list of commonly used Endpoints or IP endpoints is: Bluetooth (MAC 48-bit addresses), Ethernet, fiber channel, USB, UDP, FDDI, IPv4, IPv6, JXTA, NCP, TCP, etc.
- Name resolutions are used to convert numerical values into the human-readable format. There are two ways- network services resolution and resolve from Wireshark configuration files. It is only possible when capturing is not in progress. It can be resolved after the packet is added to the list. To rebuild the list with correct resolved names you can use **View-> Reload**.
- In ARP, Wireshark asks the OS to convert the Ethernet address to the IP address.
- Since it is a live capture process, so it is important to set the correct time and zone on your computer.

TELEPHONY

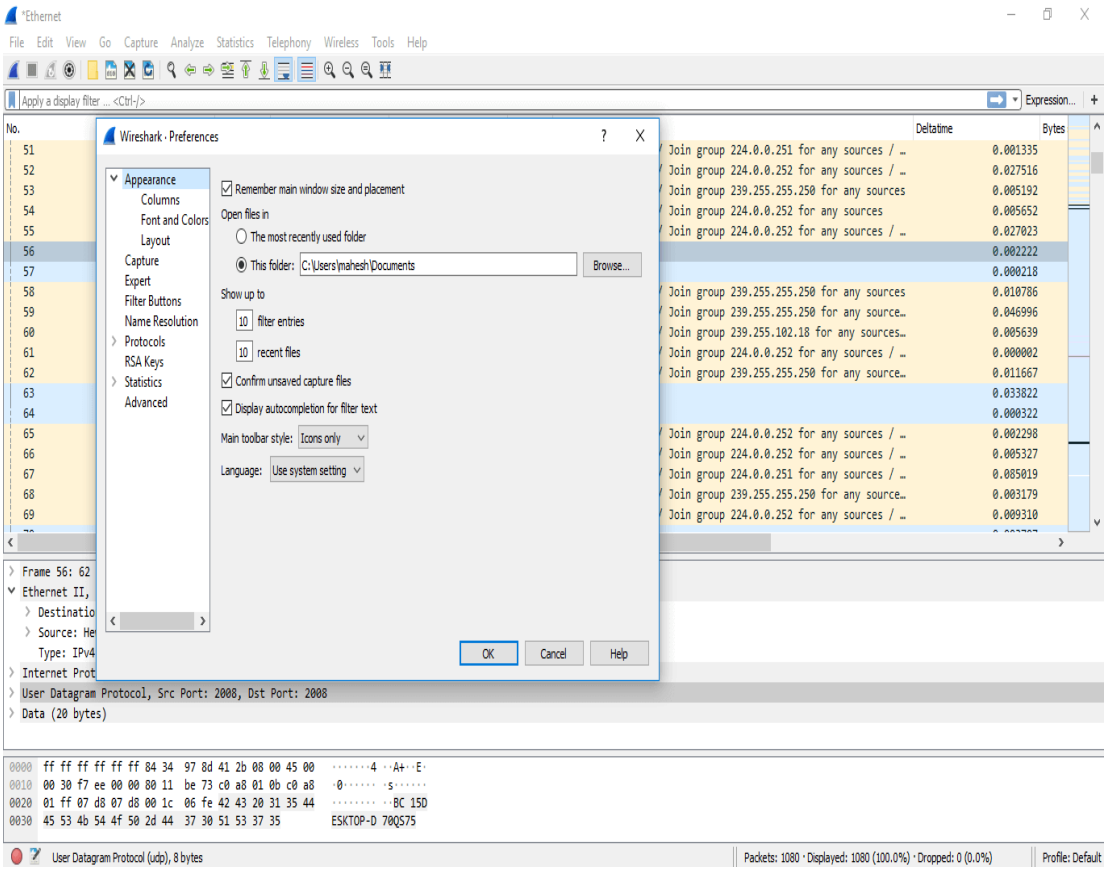
The Telephony is the option on the menu bar. The image is shown below:



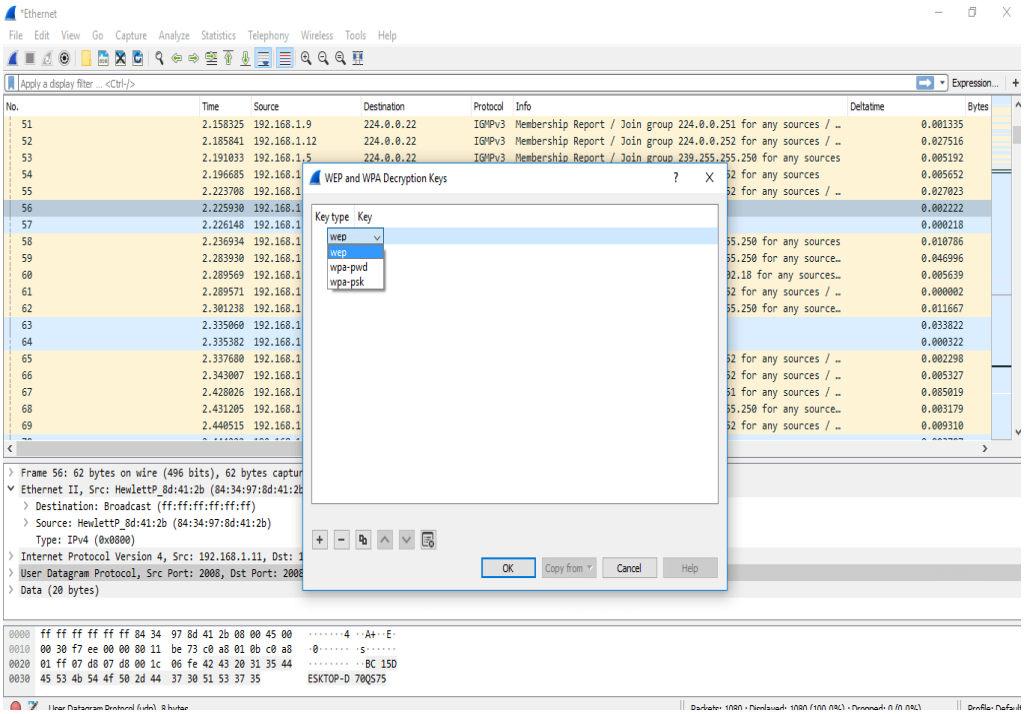
WIRESHARK DECRYPTION

The decryption process is used for the data to be in a readable format. Below are the steps for the decryption process.

- Open the Wireshark and then select the particular interface as explained above.
- Go to the 'Edit' option and select the 'Preferences' option.
- A dialogue will appear as shown below:



- Select the 'Protocol' option in the left column.
- From the drop-down list, select the 'IEEE 802.11' option. Check the box of decryption and click on the Edit option under it.
- A box will appear. Click on the option shown below:


*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Info Deltatime Bytes
51 2.158325 192.168.1.9 224.0.0.22 IGMPv3 Membership Report / Join group 224.0.0.251 for any sources / ... 0.001335
52 2.185841 192.168.1.12 224.0.0.22 IGMPv3 Membership Report / Join group 224.0.0.252 for any sources / ... 0.027516
53 2.191033 192.168.1.5 224.0.0.22 IGMPv3 Membership Report / Join group 239.255.255.250 for any sources 0.005192
54 2.196685 192.168.1 for any sources 0.005652
55 2.223708 192.168.1 for any sources / ... 0.027023
56 2.225930 192.168.1 0.002222
57 2.226148 192.168.1 0.000218
58 2.236934 192.168.1 55.250 for any sources 0.010786
59 2.283930 192.168.1 55.250 for any source... 0.046996
60 2.289569 192.168.1 92.18 for any sources... 0.005639
61 2.289571 192.168.1 for any sources / ... 0.000002
62 2.301238 192.168.1 55.250 for any source... 0.011667
63 2.335060 192.168.1 0.033822
64 2.335382 192.168.1 0.000322
65 2.337680 192.168.1 for any sources / ... 0.002298
66 2.343007 192.168.1 for any sources / ... 0.005327
67 2.428026 192.168.1 for any sources / ... 0.085019
68 2.431205 192.168.1 55.250 for any source... 0.003179
69 2.440515 192.168.1 for any sources / ... 0.009310
Frame 56: 62 bytes on wire (496 bits), 62 bytes captur
Ethernet II, Src: HewlettP_8d:41:2b (84:34:97:8d:41:2b
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Source: HewlettP_8d:41:2b (84:34:97:8d:41:2b)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.1.11, Dst: 1
User Datagram Protocol, Src Port: 2008, Dst Port: 2008
Data (20 bytes)
0000 ff ff ff ff ff ff 84 34 97 8d 41 2b 08 00 45 00 .....4 ..A+..E.
0010 00 30 f7 ee 00 00 80 11 be 73 c0 a8 01 0b c0 a8 .0..... .s.....
0020 01 ff 07 d8 07 d8 00 1c 06 fe 42 43 20 31 35 44 ....BC 15D
0030 45 53 4b 54 4f 50 2d 44 37 30 51 53 37 35 ESKTOP-D 70QS75
WEP and WPA Decryption Keys
Key type Key
wep
wep
wpa-pwd
wpa-psk
OK
Copy from
Cancel
Help

EXPERIMENT-14

AIM: How to run Nmap scan

While Nmap was once a Unix-only tool, a Windows version was released in 2000 and has since become the second most popular Nmap platform (behind Linux). Because of this popularity and the fact that many Windows users do not have a compiler, binary executables are distributed for each major Nmap release. We support Nmap on Windows 7 and newer, as well as Windows Server 2008 and newer. We also maintain a guide for users who must run Nmap on earlier Windows releases. While it has improved dramatically, the Windows port is not quite as efficient as on Unix. Here are the known limitations:

- Nmap only supports ethernet interfaces (including most 802.11 wireless cards and many VPN clients) for raw packet scans. Unless you use the -sT -Pn options, RAS connections (such as PPP dialups) and certain VPN clients are not supported. This support was dropped when Microsoft removed raw TCP/IP socket support in Windows XP SP2. Now Nmap must send lower-level ethernet frames instead.
- **.reg**. To make the changes by hand, add these three Registry DWORD values to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters:

MaxUserPort: Set a large value such as 65534 (0x0000fffe). See MS KB 196271.
TCPTimedWaitDelay: Set the minimum value (0x0000001e). See MS KB 149532.
StrictTimeWaitSeqCheck

Set to 1 so TCPTimedWaitDelay is checked.

Note
I would like to thank Ryan Permeh of eEye, Andy Lutomirski, and Jens Vogt for their hard work on the Nmap Windows port. For many years, Nmap was a Unix-only tool, and it would likely still be that way if not for their efforts.

Windows users have three choices for installing Nmap, all of which are available from the download page at https://nmap.org/download.html.

Windows Self-installer

Every Nmap release includes a Windows self-installer named nmap-<version>-setup.exe (where <version> is the version number of the specific release). Most Nmap users choose this option since it is so easy. Another advantage of the self-installer is that it provides the option to install the Zenmap GUI and other tools. Simply run the installer file and let it walk you through panels for choosing an install path and installing Npcap. The installer was created with the open-source Nullsoft Scriptable Install System. After it completes, read the section called "Executing Nmap on Windows" for instructions on executing Nmap on the command-line or through Zenmap.

Command-line Zip Binaries

Note

Most users prefer installing Nmap with the self-installer discussed previously.

Every stable Nmap release comes with Windows command-line binaries and associated files in a Zip archive. No graphical interface is included, so you need to run nmap.exe from a DOS/command window. Or you can download and install a superior command shell such as those included with the free Cygwin system available from https://www.cygwin.com. Here are the step-by-step instructions for installing and executing the Nmap .zip binaries.

Installing the Nmap zip binaries

1. Download the .zip binaries from https://nmap.org/download.html.
2. Extract the zip file into the directory you want Nmap to reside in. An example would be C:\Program Files. A directory called nmap-<version> should be created, which includes the Nmap executable and data files.
3. For improved performance, apply the Nmap Registry changes discussed previously.
4. Nmap requires the free Npcap packet capture library. We include a recent Npcap installer which is available in the zip file as npcap-<version>.exe, where <version> is the Npcap version rather than the Nmap version. Alternatively, you can obtain and install the latest version from https://npcap.com.

5. Due to the way Nmap is compiled, it requires the Microsoft Visual C++ Redistributable Package of runtime components. Many systems already have this installed from other packages, but you should run VC_redist.x86.exe from the zip file just in case you need it. Pass the /q option to run these installers in quiet (non interactive) mode.
6. Instructions for executing your compiled Nmap are given in the section called "Executing Nmap on Windows".

Compile from Source Code

Most Windows users prefer to use the Nmap binary self-installer, but compilation from source code is an option, particularly if you plan to help with Nmap development. Compilation requires Microsoft Visual C++ 2019, which is part of their commercial Visual Studio suite. Any of the Visual Studio 2019 editions should work, including the free Visual Studio 2019 Community.

Some of Nmap's dependencies on Windows are inconvenient to build. For this reason, precompiled binaries of the dependencies are stored in Subversion, in the directory /nmap-mswin32-aux. When building from source, whether from a source code release or from Subversion, check out /nmap-mswin32-aux as described below.

Executing Nmap on Windows

Nmap releases now include the Zenmap graphical user interface for Nmap. If you used the Nmap installer and left the Zenmap field checked, there should be a new Zenmap entry on your desktop and Start Menu. Click this to get started. Zenmap is fully documented in Chapter 12, Zenmap GUI Users' Guide. While many users love Zenmap, others prefer the

traditional command-line approach to executing Nmap. Here are detailed instructions for users who are unfamiliar with command-line interfaces:

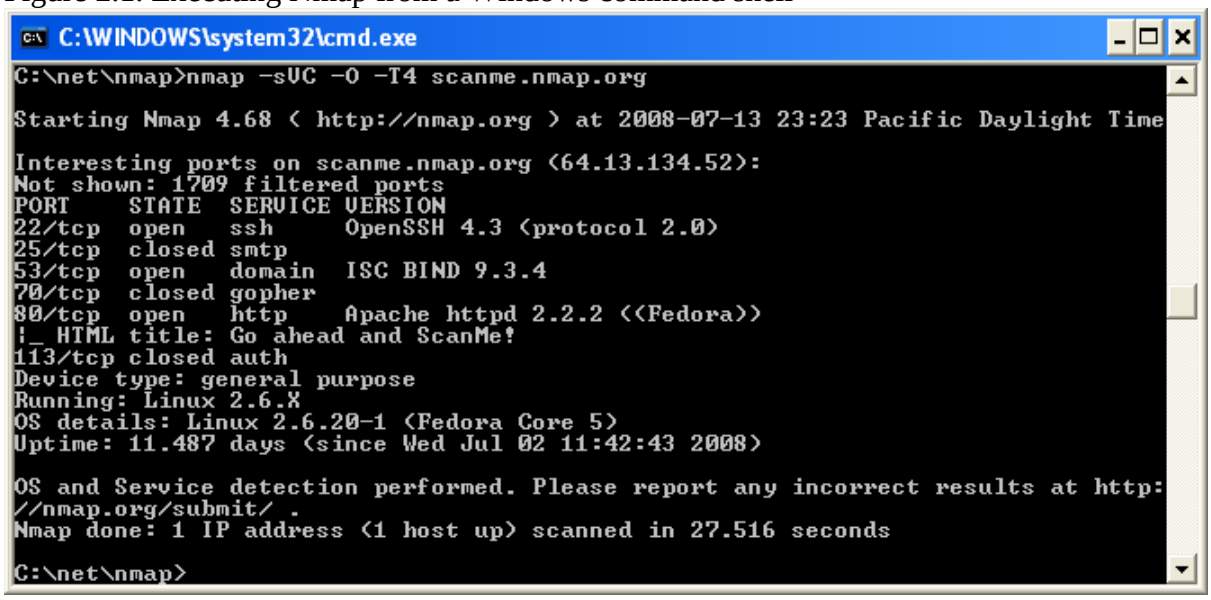
1. Make sure the user you are logged in as has administrative privileges on the computer (user should be a member of the administrators group).
2. Open a command/DOS Window. Though it can be found in the program menu tree, the simplest approach is to choose "Start" -> "Run" and type **cmd<enter>**. Opening a Cygwin window (if you installed it) by clicking on the Cygwin icon on the desktop works too, although the necessary commands differ slightly from those shown here.
3. Change to the directory you installed Nmap into. You can skip this step if Nmap is already in your command path (the Zenmap isntaller adds it there by default). Otherwise, type the following commands.

```
c:
cd "\Program Files (x86)\Nmap"
```

   On Windows releases prior to Windows 7, specify \Program Files\Nmap instead. The directory will also be different if you chose to install Nmap in a non-default location.

6. Execute **nmap.exe**. Figure 2.1 is a screen shot showing a simple example.

Figure 2.1. Executing Nmap from a Windows command shell



If you execute Nmap frequently, you can add the Nmap directory (c:\Program Files (x86)\Nmap by default) to your command execution path:

1. Open the System Properties window to the Advanced tab by running SystemPropertiesAdvanced.exe.
2. Click the "Environment Variables" button.
3. Choose Path from the System variables section, then hit edit.
4. Add a semi-colon and then your Nmap directory (e.g. c:\Program Files (x86)\Nmap) to the end of the value.
5. Open a new command prompt and you should be able to execute a command such as **nmap scanme.nmap.org** from any directory.

Nmap on Windows: Installation and Usage Guide for Windows Users

Available for Windows, Linux, macOS, and a range of other operating systems, Nmap is widely used to perform network scans, conduct security auditing, and find vulnerabilities in networks.

As the project's official page explains, at the most basic level, Nmap allows you to quickly map the ports on your network, and do so without being detected.

This functionality is accessed through a well-structured set of Nmap commands which will be familiar to anyone who has worked with command-line network tools before. Commands can also be built into Nmap scripts to extend your capabilities even further.

Installing Nmap on Windows, and for that matter using it on Windows, is fairly straightforward. We'll show you how to download Nmap, and how to install it. We'll then take you through the most common use cases for Nmap, before showing you the official GUI alternative called ZeNmap.

¶Installing Nmap on Windows
Installing Nmap on Windows is straightforward:

1. The first step is to go to the official download page and download the latest stable version of Nmap.

   NOTE: There are typically a number of different versions of Nmap available–the latest stable version, in addition to early-release betas that will offer extra features at the cost of some stability. Download the version you feel most comfortable with, which for most beginners will be the latest stable version.

   Next, navigate to the location where the file is downloaded. If your Windows installation is fairly standard, this will be in your "Downloads" folder. You will see a file there called "Nmap-X.XX-setup", or similar. If you can't find the file, do a quick search for it.

2. This file is an EXE–an executable. In order to use it, you will have to run it with administrator privileges. To do that, right-click the file and then click "run as administrator."

3. The installer will now run. A window will appear that will ask you to accept the end-user agreement. Click "I Agree" to do so.

4. Next, the installer will ask you which components of Nmap you'd like to install. All of the components will be selected and installed by default. Unless you are experienced with the program and don't need some of these components, go ahead and accept the proposed installation.

5. Then the installer will ask you where you want to install Nmap. It will default to C:\Program Files (x86)\Nmap, but you can change this if you would like to. The important thing is that you know where Nmap is installed, because (as we'll see shortly) you'll need that information in order to call it from the command line.

6. Click "Install", and Nmap will start to install. This should be a pretty quick process, even on old hardware–Nmap is a small program, despite being so useful!

7. You'll then get confirmation that Nmap is inIf all went smoothly, you should now have a working version of Nmap on your computer.

   Depending on your level of experience, however, you might be a little confused at this point. By default, Nmap is a command-line tool, and as such it doesn't have an icon that appears in your programs menu.

   If you are familiar with using the command line (or want to give it a go), you can proceed to the next section. If you want a graphical program (a GUI) for Nmap, take a look at the section on ZeNmap below–this program will provide a more familiar interface for novice users.

   ¶
   Available for Windows, Linux, macOS, and a range of other operating systems, Nmap is widely used to perform network scans, conduct security auditing, and find vulnerabilities in networks.
   As the project's official page explains, at the most basic level, Nmap allows you to quickly map the ports on your network, and do so without being detected.

   This functionality is accessed through a well-structured set of Nmap commands which will be familiar to anyone who has worked with command-line network tools before. Commands can also be built into Nmap scripts to extend your capabilities even further.

   Installing Nmap on Windows, and for that matter using it on Windows, is fairly straightforward. We'll show you how to download Nmap, and how to install it. We'll then take you through the most common use cases for Nmap, before showing you the official GUI alternative called ZeNmap.

   ¶Installing Nmap on Windows
   Installing Nmap on Windows is straightforward:

1. The first step is to go to the official download page and download the latest stable version of Nmap.

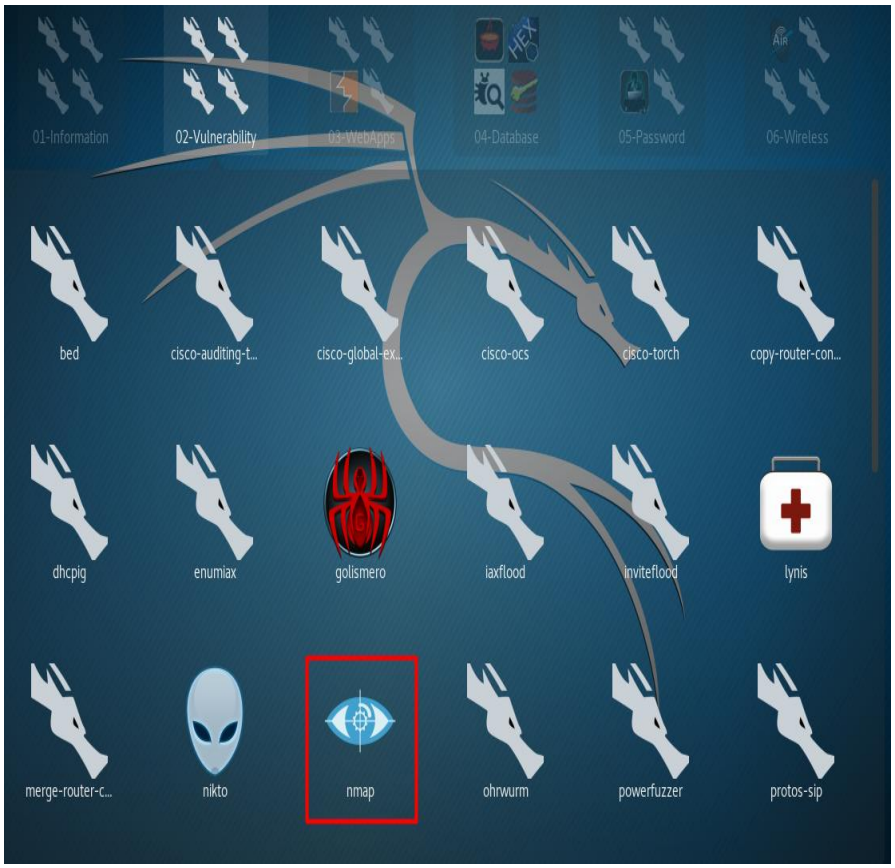
   NOTE: There are typically a number of different versions of Nmap available–the latest stable version, in addition to early-release betas that will offer extra features at the cost of some stability. Download the version you feel most comfortable with, which for most beginners will be the latest stable version.

2. Next, navigate to the location where the file is downloaded. If your Windows installation is fairly standard, this will be in your "Downloads" folder. You will see a file there called "Nmap-X.XX-setup", or similar. If you can't find the file, do a quick search for it.
3. This file is an EXE–an executable. In order to use it, you will have to run it with administrator privileges. To do that, right-click the file and then click "run as administrator."
4. The installer will now run. A window will appear that will ask you to accept the end-user agreement. Click "I Agree" to do so.
5. Next, the installer will ask you which components of Nmap you'd like to install. All of the components will be selected and installed by default. Unless you are experienced with the program and don't need some of these components, go ahead and accept the proposed installation.
6. Then the installer will ask you where you want to install Nmap. It will default to C:\Program Files (x86)\Nmap, but you can change this if you would like to. The important thing is that you know where Nmap is installed, because (as we'll see shortly) you'll need that information in order to call it from the command line.

7. Click "Install", and Nmap will start to install. This should be a pretty quick process, even on old hardware–Nmap is a small program, despite being so useful!
8. You'll then get confirmation that Nmap is installed.

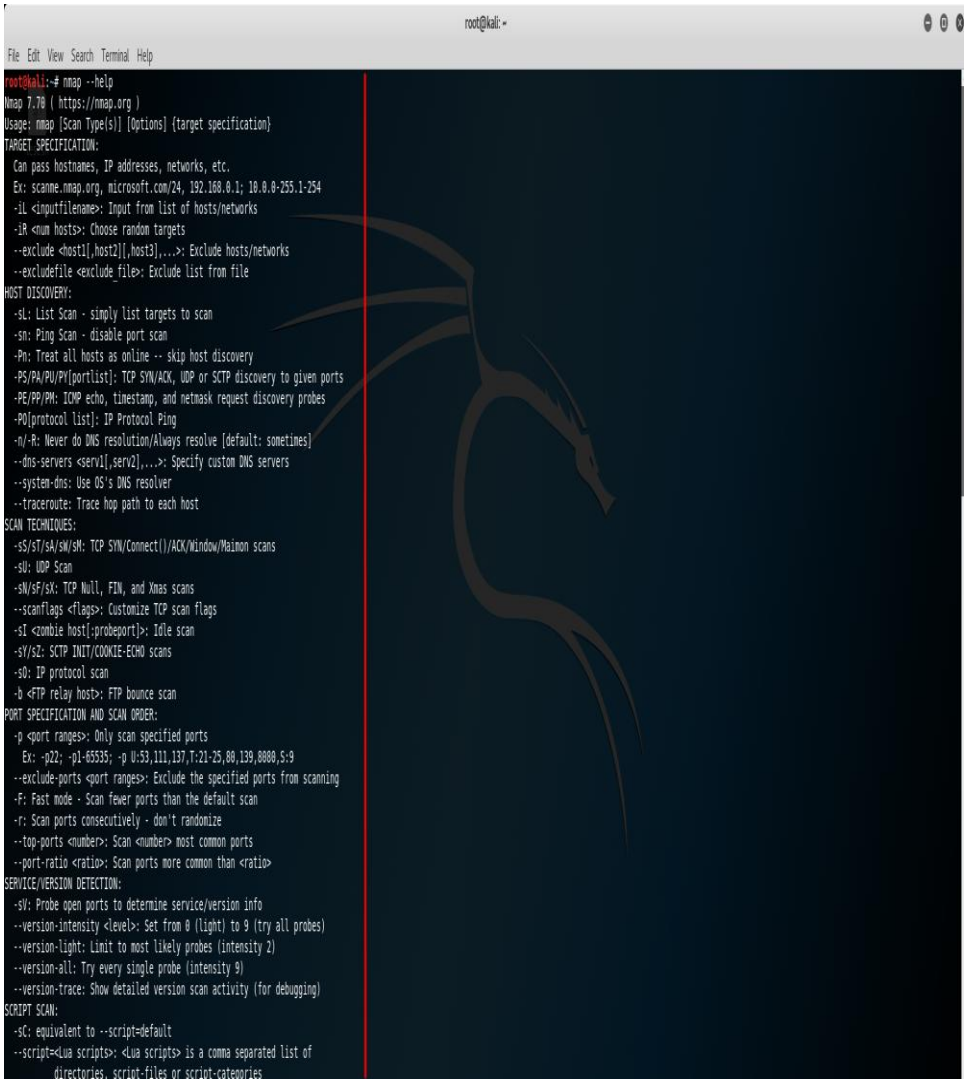
How to launch Nmap?

In Windows hosts you can simply install nmap and run it from the desktop icon using administrator privileges . In linux hosts there are 2 ways of doing it, in case of kali linux and parrot os you can find the icon and click to start and later give it root privileges by entering your password .



The other way is you can simply run

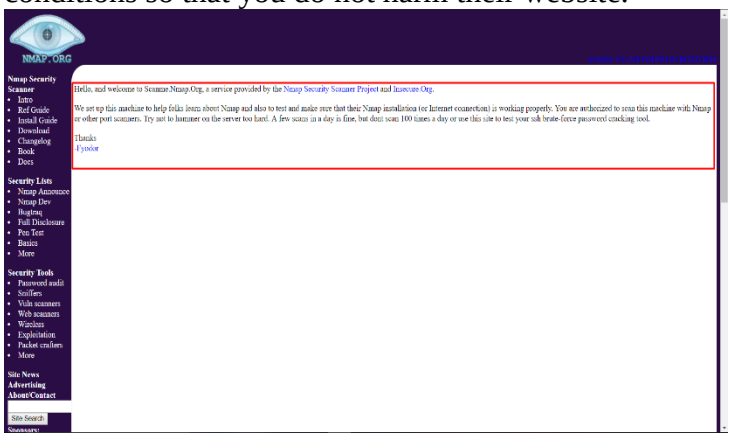
nmap --help



You can use it as a manual for using commands, just scroll down and head towards examples.

How to do simple scans and be legal?

As already mentioned, scanning networks and websites using nmap can be illegal, you may need written permissions to so. So, to do scans that are legal you can use scanme.org, they offer you to perform scans on their website without any issues, but please read their conditions so that you do not harm their website.



Now lets see a simple example to do a scan. To do so simply use nslookup command following the website url or address. If you do not know the IP address of the website and using the command.

nslookup scanme.nmap.org

will give you its address. Now when you get the address you can use the same for scanning the network by

nslookup "address"

the address should be written as IP address which you found on the previous scan and without quotes.

```
root@kali:~# nslookup scanme.nmap.org
Server:         192.168.29.2
Address:        192.168.29.2#53

Non-authoritative answer:
Name:   scanme.nmap.org
Address: 45.33.32.156
Name:   scanme.nmap.org
Address: 2600:3c01::f03c:91ff:fe18:bb2f

root@kali:~# nslookup 45.33.32.156
156.32.33.45.in-addr.arpa       name = scanme.nmap.org.

Authoritative answers can be found from:

root@kali:~#
```

This is how you can do a simple network scan. Now you can also save your scans in a text file for simplicity by using the command

nslookup 45.33.32.156 >> result.txt

```
root@kali:~# nslookup 45.33.32.156 >> result.txt
root@kali:~#
```

Please note that nmap is a very noisy scanning utility and you need to be anonymous and legal in some cases to do so. Please ensure that you use it for legal and educational purposes

EXPERIMENT-15

AIM: Operating System Detection using Nmap

Service and OS detection

Nmap is one of the most popular tools used for the enumeration of a targeted host. Nmap can use scans that provide the OS, version, and service detection for individual or multiple devices. Detection scans are critical to the enumeration process when conducting penetration testing of a network. It is important to know where vulnerable machines are located on the network so they can be fixed or replaced before they are attacked. Many attackers will use these scans to figure out what payloads would be most effective on a victim's device. The OS scan works by using the TCP/IP stack fingerprinting method. The services scan works by using the Nmap-service-probes database to enumerate details of services running on a targeted host.

Detect OS and services

This is the command to scan and search for the OS (and the OS version) on a host. This command will provide valuable information for the enumeration phase of your network security assessment (if you only want to detect the operating system, type nmap -O 192.168.0.9):

nmap -A 192.168.0.9
sniffers-12.jpg

```
root@EthicalHaks:~# nmap -A 192.168.0.9

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:49 PDT
Nmap scan report for 192.168.0.9
Host is up (0.000058s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE VERSION
111/tcp open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100024  1          46044/udp  status
|_  100024  1          54793/tcp  status
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.8 - 4.4
Network Distance: 0 hops

OS and Service detection performed. Please report any incorrect results at https
://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.71 seconds
```

Standard service detection

This is the command to scan for running service. Nmap contains a database of about 2,200 well-known services and associated ports. Examples of these services are HTTP (port 80), SMTP (port 25), DNS (port 53), and SSH (port 22):

nmap -sV 192.168.0.9

sniffers-13.jpg

```
root@EthicalHaks:~# nmap -sV 192.168.0.9

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 21:50 PDT
Nmap scan report for 192.168.0.9
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE VERSION
111/tcp open  rpcbind 2-4 (RPC #100000)

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.78 seconds
```

More aggressive service detection

This is the command for an aggressive scan. Usually, experienced hackers will not use this command because it is noisy and leaves a large footprint on the network. Most black hat hackers prefer to run as silently as possible:

nmap -sV --version-intensity 5 192.168.0.9

sniffers-14.jpg

```
root@EthicalHaks:~# nmap -sV 192.168.0.9 --version intensity 5

Nmap version 7.12 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.2.4 openssl-1.0.2g libpcre-8.38 nmap-libpcap-1.7.3 nmap-
libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

Lighter banner-grabbing detection

This is the command for a light scan. A hacker will often use a light scan such as this to remain undetected. This scan is far less noisy than an aggressive scan. Running silently and staying undetected gives the hacker a major advantage while conducting enumeration of targeted hosts:

nmap -sV --version-intensity 0 192.168.0.9

sniffers-15.jpg

```
root@EthicalHaks:~# nmap -sV 192.168.0.9 --version intensity 0

Nmap version 7.12 ( https://nmap.org )
Platform: x86_64-pc-linux-gnu
Compiled with: liblua-5.2.4 openssl-1.0.2g libpcre-8.38 nmap-libpcap-1.7.3 nmap-
libdnet-1.12 ipv6
Compiled without:
Available nsock engines: epoll poll select
```

Service and OS detection depend on different techniques to determine the operating system or service running on a certain port. A more aggressive service detection is useful if there are services running on unexpected ports, although the lighter version of the service will be much faster and leave less of a footprint. The lighter scan does not attempt to detect the service; it simply grabs the banner of the open service to determine what is running.

Nmap output formats

Save default output to file

This command saves the output of a scan. With Nmap, you can save the scan output in different formats:

nmap -oN outputfile.txt 192.168.0.12

sniffers-16.jpg

```
root@EthicalHaks:~# nmap -oN outputfile.txt 192.168.0.12

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 22:00 PDT
Nmap scan report for 192.168.0.12
Host is up (0.0000020s latency).
Not shown: 999 closed ports
PORT    STATE SERVICE
111/tcp open  rpcbind

Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

Save in all formats

This command allows you to save in all formats. The default format can also be saved to a file using a file redirect command, or > file. Using the -oN option allows the results to be saved, but also allows them to be viewed in the terminal as the scan is being conducted:

nmap -oA outputfile 192.168.0.12

sniffers-17.jpg

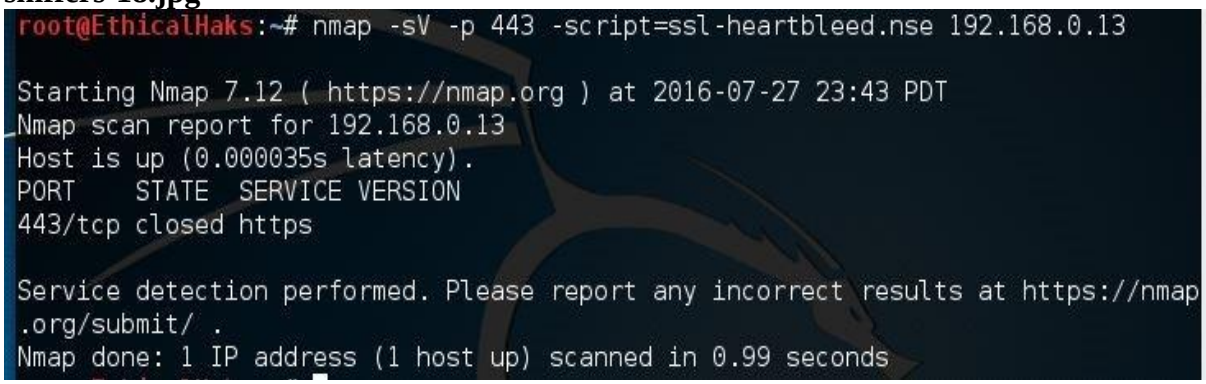
```
root@EthicalHaks:~# nmap -oA outputfile.txt 192.168.0.12

Starting Nmap 7.12 ( https://nmap.org ) at 2016-07-23 22:02 PDT
Nmap scan report for 192.168.0.12
```

Scan using a specific NSE script

This command will search for a potential heartbleed attack. A Heartbleed attack exploits a vulnerability that is found in older, unpatched versions of OpenSSL:

nmap -sV -p 443 -script=ssl-heartbleed.nse 192.168.1.1

sniffers-18.jpg



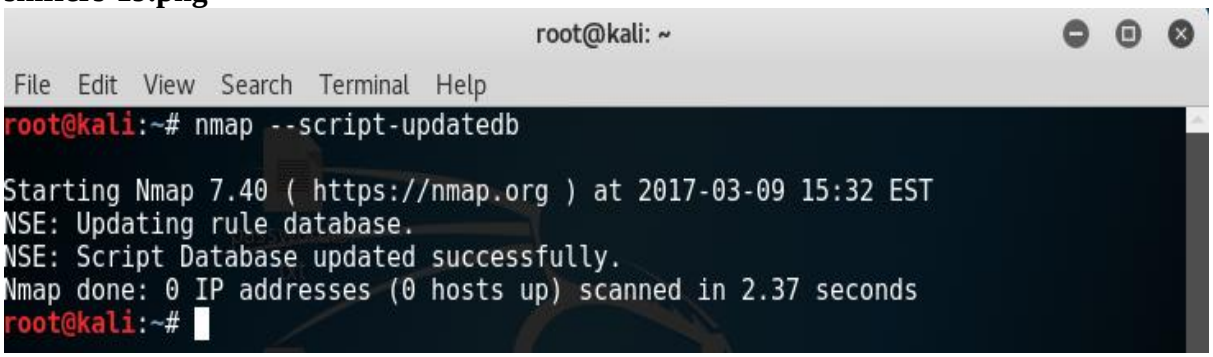
Scan with a set of scripts

This command is useful when searching for multiple types of attack. Using multiple scripts will save time and allow for better efficiency while monitoring the network. You can also use the following command to scan for heartbleed attacks:

nmap -sV -p 443 --script=ssl-heartbleed 192.168.0.13/24

It is important to keep an updated database of current scripts. To update the Nmap script database, type the command nmap - -script-updatedb . The following screenshot demonstrates the screen you will see when you run this command:

sniffers-19.png



Currently, Nmap has 471 NSE scripts installed. The scripts allow you to perform a wide range of network security testing and discovery functions. If you are serious about your network security, then you should take the time to get familiar with some of the Nmap scripts.

The option --script-help=$scriptname will show help for the individual scripts. To get a list of installed scripts, use the command locate nse | grep script.

EXPERIMENT-16

AIM: Do the following using NS2 Simulator

i. **NS2 Simulator-Introduction**
ii. **Simulate to Find the Number of Packets Dropped**
iii. **Simulate to Find the Number of Packets Dropped by TCP/UDP**
iv. **Simulate to Find the Number of Packets Dropped due to Congestion**
v. **Simulate to Compare Data Rate& Throughput.**

NS2 Simulator-Introduction

NS2 Simulators is one of our prime services started for the beginners, who wish to learn NS-2 completely. It is also a widespread network simulator used by majority of students and scholars today. Our Tutorial for Beginners provides basic information about NS2, which can be utilized by students to get an idea about NS-2. Right start is the key for right success. This is the reason; we focus on guiding the beginners as they are like mud which can be molded in any ways.

NS-2 BASICS:

It is also an open source simulator widely used for Educational and research purpose. Two major things one must know about it, is also the language used in it and it's supported Networks. It is also basically implemented in OTCL [Object oriented extension of TCL]. Two major languages, it also supports are C++ and TCL [Tool command language]. It is also used to simulate both wired and wireless Networks [Mobile Ad hoc Networks, Wireless sensor Networks, Wireless body area network etc]. Simulation in NS2 is based on the following constraints i.e. Node creation and also configuration, Node color description, creation of duplex links and also orientations, labeling of Nodes, establishing queuing methods, TCP and also UDP connection and data transmission.

Now, let's discuss most basic aspect of NS2 i.e. overall commands involved in Node creation, links, agents and applications etc.

NODE COMMANDS:

To create a wired node:

[simulator – instance] node

Set ns[new simulator] set n1[$ns node]

To change the shape of the Node:

- Use of shape procedure of node class.
- Available shapes are box, circle and also hexagon
- Default shape[Circle]
- Syntax:

[node – instance] shape < circle| hexagon | box >

To reset all the agents of the Node:

- Use of Rest command to rest all the agents attached to a node

[node – instance] reset

To fetch id of Node:

[node- instance] id

To attach an agent to a node on specific port:

[node – instance] attach – agent [agent – instance] optional :< Port no

To attach label to the node:
[node – instance] label [label]

LINK COMMAND IN NS2:

- In NS2, nodes are connect in two ways[Simplex and Duplex]
- Simplex Connection-One way communication
- Duplex connection-Allow two way communication

It requires

- Bandwidth[specified in Mbps(Mb)]
- Delay[Specified in milli seconds]
- Type of Queue[DropTail, FQ, CBQ, RED, DRR, FQ etc]
- Syntax:

$ns link type [simplex – link/duplex-link] [node – instance 1] [node – instance 2] bandwidth delay Q-Type

AGENTS IN NS2

- Every node transport mechanism in NS2 need to also defined to send data which is done using agents·
- Foe ex. FTP application uses TCP protocol as also TCP agent·
- Command to attach agent to node:

$ [simulator – instance] attach – agent [node – instance] [agent – instance]

- To fetch the port number also to which the agents is attached:

$ [agent – instance] port

- Connect one agent to another agent:

$ [simulator – instance] connect [agent – instance] [agent – instance]

APPLICATION IN NS2:

- Two types of application[traffic generator and also simulated application]
- Traffic agent:
  - -Pareto traffic generator
  - -Exponential traffic generator
  - -CBR and also trace
- Simulated Application [FTP and Telnet]

Overall Syntax:

$set cbr [new Application / Traffic / CBR]

$ set exp [new Application / Traffic/ Exponential]

$set per [new Application / Traffic / Pareto]

$ set tr [new Application / Traffic / Trace]

$set ftp [new Application / FTP]

$ set tel [new Application / Telnet]

We also have discuss most basic aspect of NS2 Simulator Tutorial for beginners. This will give you an idea about the major components use in NS-2.