IDEAL INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUK, Kakinada)

VIDYUT NAGAR, KAKINADA - 533003. (A.P.)

DEPARTMENT OF CSE - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

COMPUTER NETWORKS LAB MANUAL

For III B. Tech II Semester

Subject Code : R203242
Regulation : R20/JNTUK
Academic Year : 2023-2024

IDEAL INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

COMPUTER NETWORKS LAB MANUAL

VISION AND MISSION OF THE INSTITUTION

VISION

To emerge as a premier institution for technical education in the country through academic excellence and to be recognized as a center for excellence in research & development, catering to the needs of our country.

MISSION

1. To establish a robust institution that consistently maintains state-of-the-art infrastructure facilities.
2. To create a cohesive, world-class team of faculty dedicated to innovative and experimental teaching–learning processes.
3. To enhance industry interaction and facilitate the delivery of technical education and research tailored to meet the specific needs of the country.

VISION AND MISSION OF THE DEPARTMENT

VISION

To build a strong teaching-learning base with a flair for innovation and research that responds to the dynamic needs of the software industry and the society with good ethical practices.

MISSION

- To provide strong foundation both in theory and applications of Computer Science & Engineering, to solve real-world problems.
- To empower students with state-of-art knowledge and up to date technological skills, making them globally competent.
- Inculcating professional behavior, strong ethical values, innovative research capabilities and leadership abilities.

IDEAL INSTITUTE OF TECHNOLOGY

DEPARTMENT OF CSE - ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

COMPUTER NETWORKS LAB MANUAL

Program Specific Outcomes	
PSO1	An ability to formulate and analyze a problem, and define the computing requirements to its solution using basic principles of mathematics science and computer engineering and ability to design, implement, and evaluate a computer based system, process, component, or software to meet the desired needs.
PSO2	An ability to design and conduct research based experiments, perform analysis and interpretation of data and provide valid conclusions and ability to use current techniques, skills, and tools necessary for computing practice, and understanding and commitment towards the professional and ethical responsibilities of an engineer.

Course Outcomes	
CO1	Know how reliable data communication is achieved through data link layer.
CO2	Suggest appropriate routing algorithm for the network.
CO3	Provide internet connection to the system and its installation.
CO4	Work on various network management tools.

Course Objectives
This Course will enable students to
• Learn basic concepts of computer networking
• Acquire practical notions of protocols with the emphasis on TCP/IP
• Gain a practical approach to Ethernet/Internet networking
• Understand the layered architecture and how do some important protocols work

INDEX

S.NO.	NAME OF THE EXPERIMENT
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 – CRC 12, CRC 16 and CRC CCIP.
6	Write a Program to implement Sliding window protocol for Goback N.
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 Broadcast tree by taking subnet of hosts.
13	Wireshark i. Packet Capture Using Wire shark ii. Starting Wire shark 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 v. Simulate to Compare Data Rate& Throughput.

Experiment-1

Study of Network devices in detail and connect the computers in Local Area Network
a) Aim: Study of Network Devices in Detail

The following are the various Network Devices used in Computer Networks:

- Repeater
- Hub
- Switch
- Bridge
- Router
- Gate Way
- Brouter
- NIC

1. Repeater:
Functions at Physical Layer. A repeater is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted to extend the length to which the signal can be transmitted over the same network. Repeaters not only amplify the signal but also regenerate it Repeater has two ports, so cannot be used to connect more than two devices.

2. Hub:
An Ethernet hub, (active hub, passive hub, intelligent hub, network hub, repeater hub, hub) or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, the collision domain of all hosts connected through Hub remains one. Also, they do not have the intelligence to find out the best path for data packets which leads to inefficiencies and wastage.

3. Bridge:
A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of the source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device. A network bridge connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term bridge formally means a device that behaves according to the IEEE 802.1D standard. Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.

4. Switch:
A network switch or switching hub is a computer networking device that connects network segments. A switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. The switch can perform error checking before forwarding data, which makes it very efficient as it does not forward packets that have errors and forward good packets selectively to the correct

port only. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.

5. Router:
A router is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

6. Gate Way:
In a communications network, a network node equipped for interfacing with another network that uses different protocols is called a gateway.

• A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.

• A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.

7. Brouter:
It is also known as the bridging router is a device that combines features of both bridge and router. It can work either at the data link layer or a network layer. Working as a router, it is capable of routing packets across networks and working as the bridge, it is capable of filtering local area network traffic.

8. NIC:
NIC or network interface card is a network adapter that is used to connect the computer to the network. It is installed in the computer to establish a LAN. It has a unique id that is written on the chip, and it has a connector to connect the cable to it. The cable acts as an interface between the computer and the router or modem. NIC card is a layer 2 device which means that it works on both the physical and data link layers of the network model.

Different Network Devices:

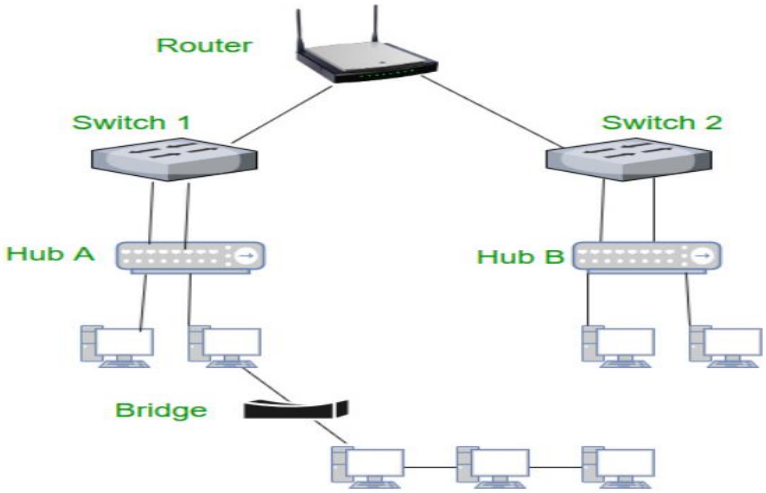


Diagram shows you how to prepare Cross wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Diagram shows you how to prepare straight through wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

b) Aim: Connect the computers in Local Area Network.

Procedure: On the host computer

On the host computer, follow these steps to share the Internet connection:

1. Log on to the host computer as Administrator or as Owner.
2. Click Start, and then click Control Panel.

3. Click Network and Internet Connections.
4. Click Network Connections.
5. Right-click the connection that you use to connect to the Internet. For example, if you connect to the Internet by using a modem, right-click the connection that you want under Dial-up / other networks available.
6. Click Properties.
7. Click the Advanced tab.
8. Under Internet Connection Sharing, select - Allow other network users to connect through this computer's Internet connection check box.
9. If you are sharing a dial-up Internet connection, select - Establish a dial-up connection whenever a computer on my network attempts to access the Internet check box if you want to permit your computer to automatically connect to the Internet.
10. Click OK. You'll receive the following message:

> When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address 192.168.0. 1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP addresses, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection Sharing?

11. Click Yes. The connection to the Internet is shared to other computers on the local area network (LAN).

The network adapter that is connected to the LAN is configured with a static IP address of 192.168.0. 1 and a subnet mask of 255.255.255.0

On the client computer

To connect to the Internet by using the shared connection, you must confirm the LAN adapter IP configuration, and then configure the client computer. To confirm the LAN adapter IP configuration, follow these steps:

1. Log on to the client computer as Administrator or as Owner.
2. Click Start, and then click Control Panel.
3. Click Network and Internet Connections.
4. Click Network Connections.
5. Right-click Local Area Connection and then click Properties.
6. Click the General tab, click Internet Protocol (TCP/IP) in the connection uses the following items list, and then click Properties.
7. In the Internet Protocol (TCP/IP) Properties dialog box, click Obtain an IP address automatically (if it is not already selected), and then click OK.

Note: You can also assign a unique static IP address in the range of 192.168.0.2 to 254. For example, you can assign the following static IP address, subnet mask, and default gateway:

- IP Address 192.168.31.202
- Subnet mask 255.255.255.0
- Default gateway 192.168.31.1

8. In the Local Area Connection Properties dialog box, click OK.
9. Quit Control Panel.

2)AIM: Write a program to implement the data link layer framing methods such as
i) Character stuffing ii)Bit stuffing

DESCRIPTION: Character Stuffing−A byte is stuffed in the message to differentiate from the delimiter. This is also called character-oriented framing or byte stuffing. Bit Stuffing−A pattern of bits of arbitrary length is stuffed in the message to differentiate from the delimiter. This is also called bit -oriented framing.

i)SOURCE CODE:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char byts[500],sd,ed;
   printf("Enter characters to be stuffed: ");
   scanf("%s", &byts);getchar();
   printf("Enter Starting Delimiter: ");
   scanf("%c", &sd);getchar();
   printf("Enter Ending Delimiter: ");
   scanf("%c", &ed);
   printf("%c ",sd);
   for(int i=0;i<strlen(byts);i++)
        {
        if(byts[i]==sd || byts[i]==ed)
                printf("%c%c",byts[i],byts[i]);
        else
                printf("%c",byts[i]);
        }
        printf(" %c",ed);
        return 0;
}
```

OUTPUT:

```
Enter characters to be stuffed: goodday
Enter Starting Delimiter: y
Enter Ending Delimiter: o
y gooooddayy o
```

ii)SOURCE CODE:

```
#include<stdio.h>
#include<string.h>
void main()
{
        char bits[500];
        int count=0,i;
        printf("Enter the bits to stuffed : ");
        scanf("%s",&bits);
        printf("01111110 ");
        for(i=0;i<strlen(bits);i++)
        {
                if(bits[i]=='1')
                        count++;
                else
                        count=0;
                printf("%c",bits[i]);
                if(count==5)
                {
```

```
                        printf("0");
                        count=0;
                }
        }
        printf(" 01111110");
}
```

OUTPUT:

```
Enter the bits to stuffed : 10111111001
01111110 101111101001 01111110
```

3)AIM: Write a program to implement data link layer framing method checksum
DESCRIPTION: In checksum error detection scheme, the data is divided into k segments each of m bits. In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum. The checksum segment is sent along with the data segments. At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented. If the result is zero, the received data is accepted; otherwise discarded.
SOURCE CODE:

```
#include <stdio.h>
void printCheckSum(int* b1,int* b2,int n)
{
	int temp[n],carry=0,i,t;
	for(i=n-1;i>=0;i--)
	{
		t=b1[i]+b2[i]+carry;
		temp[i]=t%2;
		carry=t/2;
	}
	if(carry)
	{
		for(i=n-1;i>=0;i--)
		{
			t=temp[i]+carry;
			temp[i]=t%2;
			carry=t/2;
		}
	}
	//print ans
	printf("1's Sum : ");
	for(i=0;i<n;i++)
	printf("%d",temp[i]);
	printf("\n");
	printf("Checksum: ");
	for(i=0;i<n;i++)
	printf("%d",!temp[i]);
}
int main()
{
	int i,n;
	char dup_bits1[100],dup_bits2[100];
	int bits1[100],bits2[100];
	printf("Enter length of bits:");
	scanf("%d",&n);
	printf("Enter Subnet 1:");
	scanf("%s",&dup_bits1);
	for(i=0;i<n;i++)
	bits1[i]=dup_bits1[i]-'0';
	printf("Enter Subnet 2:");
	scanf("%s",&dup_bits2);
	for(i=0;i<n;i++)
	bits2[i]=dup_bits2[i]-'0';
	printf("Subnet 1: %s\n",dup_bits1);
	printf("Subnet 2: %s\n",dup_bits2);
	printCheckSum(bits1,bits2,n);
```

```
    return 0;
}
```

OUTPUT:

```
Enter length of bits:8
Enter Subnet 1:11100101
Enter Subnet 2:10101111
Subnet 1: 11100101
Subnet 2: 10101111
1's Sum : 10010101
Checksum: 01101010
```

4) AIM: Write a program for Hamming code generation for error detection and correction
DESCRIPTION: In Computer Networks, Hamming code is used for the set of error-correction codes which may occur when the data is moved from the sender to the receiver. The hamming method corrects the error by finding the state at which the error has occurred. Redundant bits are extra binary bits that are generated and added to the information-carrying bits of data transfer to ensure that no bits were lost during the data transfer. The redundancy bits are placed at certain calculated positions to eliminate the errors and the distance between the two redundancy bits is called "Hamming Distance".
SOURCE CODE:
```
#include <stdio.h>
#include <string.h>
#define is_even_parity 1
int binaryTodecimal(char num[])
{
   int i,decimal,mul=0;
   for(decimal=0,i=strlen(num)-1;i>=0;--i,++mul)
   {
      decimal = decimal + (num[i] - 48) * (1 << mul);
   }
        return decimal;
}
char Helper(char lst[])
{
        int count=0,i;
        for(i=0;i<=strlen(lst);i++)
                if(lst[i] == '1')
                        count++;
   if(count%2 != 0)
   {
        if(is_even_parity)
         return '1';
      else
         return '0';
        }
   else
        {
        if(is_even_parity)
         return '0';
      else
         return '1';
        }
}
void GetHammingCode(char d[])
{
        char p1,p2,p4,p8;
        char arr1[10] = {d[0],d[2],d[3],d[5],d[6]};
        p1 = Helper(arr1);
        char arr2[10] = {d[0],d[1],d[3],d[4],d[6]};
   p2 = Helper(arr2);
   char arr3[10] = {d[3],d[4],d[5]};
   p4 = Helper(arr3);
   char arr4[10] = {d[0],d[1],d[2]};
   p8 = Helper(arr4);
   char Hcode[20] = {d[0],d[1],d[2],p8,d[3],d[4],d[5],p4,d[6],p2,p1};
   printf("Hamming Code For %s is %s\n",d,Hcode);
```

```
}
void IsCorrect(char l[11])
{
        char p1,p2,p4,p8;
        int error,i;
        char arr1[10] = {l[0],l[2],l[4],l[6],l[8],l[10]};
        p1 = Helper(arr1);
        char arr2[10] = {l[0],l[1],l[4],l[5],l[8],l[9]};
   p2 = Helper(arr2);
   char arr3[10] = {l[4],l[5],l[6],l[7]};
   p4 = Helper(arr3);
   char arr4[10] = {l[0],l[1],l[2],l[3]};
   p8 = Helper(arr4);
   char arr5[10] = {p8,p4,p2,p1};
   error = binaryTodecimal(arr5);
   if(error == 0)
      printf("Entered Hamming Code is Correct");
   else
   {
      if(l[11-error] == '1')
                    l[11-error] = '0';
              else
                    l[11-error] = '1';
      printf("Entered Hamming Code is Wrong\n");
      printf("Corrected Hamming Code is ");
      for(i=0;i<11;i++)
        printf("%c",l[i]);
   }
}
int main()
{
        char data[10];
        printf("Enter Data :");
        scanf("%s",&data);
        GetHammingCode(data);
        char arr[11] = "10101101110";
        IsCorrect(arr);
}
```

OUTPUT:

```
Enter Data :110010
Hamming Code For 110010 is 11000101
Entered Hamming Code is Wrong
Corrected Hamming Code is 10101001110
```

5)AIM: Write a program to implement on a data set of characters the three CRC polynomials CRC 12,CRC 16 and CRC CCIP

DESCRIPTION:CRC method can detect a single burst of length n, since only one bit per column will be changed, a burst of length n+1 will pass undetected, if the first bit is inverted, the last bit is inverted and all other bits are correct. If the block is badly garbled by a long burst or by multiple shorter burst, the probability that any of the n columns will have the correct parity that is 0.5. so the probability of a bad block being expected when it should not be 2 power(-n). This scheme sometimes is known as Cyclic Redundancy Code.

SOURCE CODE:

```
#include<stdio.h>
#include<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("-----MENU-----\n");
		printf("1.Crc12\n2.Crc16\n3.Crc ccip\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("Enter data:");
		scanf("%s",t);
		printf("Generating polynomial:%s\n",g);
		a=strlen(t);
		for(e=a;e<a+N-1;e++)
```

```
                    t[e]='0';
            printf("Modified data is:%s\n",t);
            crc();
            printf("Checksum is:%s\n",cs);
            for(e=a;e<a+N-1;e++)
                    t[e]=cs[e-a];
            printf("Final codeword is : %s\n",t);
            printf("Test error detection 0(yes) 1(no)?:");
            scanf("%d",&e);
            if(e==0)
            {
                    do{
                    printf("Enter 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("Erroneous 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("No error detected\n\n");
    }while(flag!=1);
}
```

OUTPUT:

```
-----MENU-----
1.Crc12
2.Crc16
3.Crc ccip
4.Exit

Enter your option :1
Enter data:110010101
Generating polynomial:1100000001111
Modified data is:110010101000000000000
Checksum is:011110000111
Final codeword is : 110010101011110000111
Test error detection 0(yes) 1(no)?:1
No error detected

-----MENU-----
1.Crc12
2.Crc16
3.Crc ccip
4.Exit

Enter your option :2
Enter data:111010101
Generating polynomial:11000000000000101
Modified data is:1110101010000000000000000
Checksum is:0000010011111110
Final codeword is : 1110101010000010011111110
Test error detection 0(yes) 1(no)?:1
No error detected
```

```
-----MENU-----
1.Crc12
2.Crc16
3.Crc ccip
4.Exit

Enter your option :3
Enter data:10101110
Generating polynomial:10001000000100001
Modified data is:101011100000000000000000
Checksum is:01010100001001000
Final codeword is : 1010111001010100001001000
Test error detection 0(yes) 1(no)?:1
No error detected

-----MENU-----
1.Crc12
2.Crc16
3.Crc ccip
4.Exit

Enter your option :4
```

6) AIM: Write a program to implement Sliding Window protocol for Goback N

DESCRIPTION: Go – Back – N protocol provides for sending multiple frames before receiving the acknowledgment for the first frame. The frames are sequentially numbered and a finite number of frames. The maximum number of frames that can be sent depends upon the size of the sending window. If the acknowledgment of a frame is not received within an agreed upon time period, all frames starting from that frame are retransmitted.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
	int windowsize,sent=0,ack,i;
	printf("Enter window size :");
	scanf("%d",&windowsize);
	while(1)
	{
		for(i=0;i<windowsize;i++)
		{
			printf("Frame %d has been transmitted.\n",sent);
			sent++;
			if(sent == windowsize)
				break;
		}
		printf("\nPlease enter the last Acknowledgement received :");
		scanf("%d",&ack);
		if(ack == windowsize)
			break;
		else
			sent = ack;
	}
	return 0;
}
```

OUTPUT:

```
Enter window size :8
Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received :4
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received :7
Frame 7 has been transmitted.

Please enter the last Acknowledgement received :8
```

7) AIM: Write a program to implement Sliding Window Protocol for Selective repeat
DESCRIPTION: Selective Repeat protocol provides for sending multiple frames depending upon the availability of frames in the sending window, even if it does not receive acknowledgement for any frame in the interim. The maximum number of frames that can be sent depends upon the size of the sending window. Here, only the erroneous or lost frames are retransmitted, while the good frames are received and buffered. It uses two windows of equal size: a sending window that stores the frames to be sent and a receiving window that stores the frames receive by the receiver. The size is half the maximum sequence number of the frame.
SO**URCE CODE:**
```
#include <stdio.h>
#include<stdlib.h>
#define no_of_packets 15
#define window_size 5
int main()
{
	int p,i=1,a,w,nac,ack;
	a=no_of_packets;
	w=window_size;
	printf("Transmitting begins...! No of Packets : %d\n",a);
	while(i<=a)
	{
		printf("Sending Packets from %d to %d\n",i,w+i-1);
		for(p=i;p<w+i;p++)
			printf("Transmitting Packet %d\n",p);
		nac = i+rand()%w;
		if(nac==i)
		{
			printf("Ack : %d\n",w+i);
			i=i+w;
			continue;
		}
		printf("NACK : %d\n",nac);
		printf("Sending Packet : %d\n",nac);
		printf("Ack : %d\n",nac+1);
		printf("Ack : %d\n",i+w);
		i = i + w;
	}
	return 0;
}
```

OUTPUT:

```
Transmitting begins...! No of Packets : 15
Sending Packets from 1 to 5
Transmitting Packet 1
Transmitting Packet 2
Transmitting Packet 3
Transmitting Packet 4
Transmitting Packet 5
NACK : 2
Sending Packet : 2
Ack : 3
Ack : 6
Sending Packets from 6 to 10
Transmitting Packet 6
Transmitting Packet 7
Transmitting Packet 8
Transmitting Packet 9
Transmitting Packet 10
NACK : 8
Sending Packet : 8
Ack : 9
Ack : 11
Sending Packets from 11 to 15
Transmitting Packet 11
Transmitting Packet 12
Transmitting Packet 13
Transmitting Packet 14
Transmitting Packet 15
NACK : 15
Sending Packet : 15
Ack : 16
Ack : 16
```

8) AIM: Write a program to implement Stop and Wait Protocol

DESCRIPTION: Here stop and wait means, whatever the data that sender wants to send, he sends the data to the receiver. After sending the data, he stops and waits until he receives the acknowledgment from the receiver. It is a data-link layer protocol which is used for transmitting the data over the noiseless channels. It provides unidirectional data transmission which means that either sending or receiving of data will take place at a time. It provides flow-control mechanism but does not provide any error control mechanism. The idea behind the usage of this frame is that when the sender sends the frame then he waits for the acknowledgment before sending the next frame.

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
	int i,n,r,a;
	n=5;
	printf("The No of Packets are : %d\n",n);
	for(i=1;i<=n;i++)
	{
		printf("The Packet Sent is %d\n",i);
		r=rand()%2;
		if(r==1)
		{
			a = rand()%2;
			if(a==1)
			{
				printf("Ack number : %d\n",i+1);
			}
			else
			{
				printf("NACK number : %d\n",i+1);
				i--;
			}
		}
		else
		{
			printf("Time Out..! Resend Packet\n");
			i--;
		}
	}
}
```

OUTPUT:

```
The No of Packets are : 5
The Packet Sent is 1
Ack number : 2
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
NACK number : 3
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
Time Out..! Resend Packet
The Packet Sent is 2
Ack number : 3
The Packet Sent is 3
Ack number : 4
The Packet Sent is 4
Ack number : 5
The Packet Sent is 5
NACK number : 6
The Packet Sent is 5
NACK number : 6
The Packet Sent is 5
NACK number : 6
The Packet Sent is 5
Time Out..! Resend Packet
The Packet Sent is 5
NACK number : 6
The Packet Sent is 5
Time Out..! Resend Packet
The Packet Sent is 5
NACK number : 6
The Packet Sent is 5
Time Out..! Resend Packet
The Packet Sent is 5
Ack number : 6
```

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

DESCRIPTION: A state occurring in network layer when the message traffic is so heavy that it slows down network response time is called congestion. In leaky bucket algorithm, when host wants to send packet, packet is thrown into the bucket. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate. Bursty traffic is converted to a uniform traffic by the leaky bucket. In practice the bucket is a finite queue that outputs at a finite rate.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
	int no_of_queries, storage, output_pkt_size;
	int input_pkt_size, bucket_size, size_left;
	storage = 0;
	no_of_queries = 4;
	bucket_size = 10;
	input_pkt_size = 4;
	output_pkt_size = 1;
	for(int i=0; i<no_of_queries;i++)
	{
		size_left = bucket_size - storage;
		if(input_pkt_size <= size_left)
		{
			storage += input_pkt_size;
			printf("Buffer size: %d Out of bucket size: %d\n", storage, bucket_size);
		}
		else
		{
			printf("Packet loss: %d\n", (input_pkt_size-(size_left)));
			storage=bucket_size;
			printf("Buffer size: %d out of bucket size: %d\n", storage, bucket_size);

		}
		storage -= output_pkt_size;
	}
	return 0;
}
```

OUTPUT:

```
Buffer size: 4 Out of bucket size: 10
Buffer size: 7 Out of bucket size: 10
Buffer size: 10 Out of bucket size: 10
Packet loss: 3
Buffer size: 10 out of bucket size: 10
```

10) AIM: Write a program to implement Dijkstra's algorithm to compute the Shortest path through a graph

DESCRIPTION: Dijkstra's Algorithm basically starts at the node that you choose (the source node) and it analyzes the graph to find the shortest path between that node and all the other nodes in the graph. The algorithm keeps track of the currently known shortest distance from each node to the source node and it updates these values if it finds a shorter path. Once the algorithm has found the shortest path between the source node and another node, that node is marked as "visited" and added to the path.
The process continues until all the nodes in the graph have been added to the path.

SOURCE CODE:

```
#include<limits.h>
#include<stdio.h>
#include<stdbool.h>
// Number of vertices in the graph
#define V 9
int minDistance(int dist[], bool sptSet[])
{
	int min=INT_MAX,min_index;
	for(int v=0;v<V;v++)
		if(sptSet[v]==false && dist[v]<=min)
			min=dist[v],min_index=v;
	return min_index;
}
void printSolution(int dist[])
{
	printf("Vertex \t\t Distance from Source\n");
	for(int i=0;i<V;i++)
		printf("%d \t\t %d\n",i,dist[i]);
}
void dijkstra(int graph[V][V],int src)
{
	int dist[V];
	bool sptSet[V];
	for(int i=0;i<V;i++)
		dist[i]=INT_MAX,sptSet[i]=false;
	dist[src]=0;
	for(int count=0;count<V-1;count++)
	{
		int u=minDistance(dist,sptSet);
		sptSet[u]=true;
		for(int v=0;v<V;v++)
			if(!sptSet[v] && graph[u][v] && dist[u]!=INT_MAX
				&& dist[u]+graph[u][v]<dist[v])
				dist[v]=dist[u]+graph[u][v];
	}
	printSolution(dist);
}
int main()
{
	int graph[V][V]={{0,4,0,0,0,0,0,8,0},{4,0,8,0,0,0,0,11,0},
					{0,8,0,7,0,4,0,0,2},{0,0,7,0,9,14,0,0,0},
					{0,0,0,9,0,10,0,0,0},{0,0,4,14,10,0,2,0,0},
					{0,0,0,0,0,2,0,1,6},{8,11,0,0,0,0,1,0,7},
					{0,0,2,0,0,0,6,7,0}};
	dijkstra(graph,0);
```

```
        return 0;
}
```

OUTPUT:

```
Vertex           Distance from Source
0                0
1                4
2                12
3                19
4                21
5                11
6                9
7                8
8                14
```

11)AIM: 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).

DESCRIPTION: In DVR, each router maintains a routing table. It contains only one entry for each router. It contains two parts − a preferred outgoing line to use for that destination and an estimate of time (delay). Tables are updated by exchanging the information with the neighbor's nodes. Each router knows the delay in reaching its neighbors (Ex − send echo request).Routers periodically exchange routing tables with each of their neighbors. It compares the delay in its local table with the delay in the neighbor's table and the cost of reaching that neighbor. If the path via the neighbor has a lower cost, then the router updates its local table to forward packets to the neighbor.

SOURCE CODE:

```
#include<stdio.h>
struct node
{
   unsigned dist[20];
   unsigned from[20];
}rt[10];
int main()
{
   int costmat[20][20];
   int nodes,i,j,k,count=0;
   printf("Enter the number of nodes :");
   scanf("%d",&nodes);
   printf("Enter the cost matrix :\n");
   for(i=0;i<nodes;i++)
   {
      for(j=0;j<nodes;j++)
      {
         scanf("%d",&costmat[i][j]);
         costmat[i][i]=0;
         rt[i].dist[j]=costmat[i][j];
         rt[i].from[j]=j;
      }
   }
   do
   {
      count=0;
      for(i=0;i<nodes;i++)
      for(j=0;j<nodes;j++)
      for(k=0;k<nodes;k++)
      if(rt[i].dist[j]>costmat[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<nodes;i++)
   {
      printf("\nFor router %d :\n",i+1);
      for(j=0;j<nodes;j++)
                {
         printf("   Node %d via %d Distance %d\n",j+1,rt[i].from[j]+1,rt[i].dist[j]);
      }
   }
```

```
        return 0;
}
```

OUTPUT:

```
Enter the number of nodes :3
Enter the cost matrix :
0 2 7
2 0 1
7 1 0

For router 1 :
   Node 1 via 1 Distance 0
   Node 2 via 2 Distance 2
   Node 3 via 2 Distance 3

For router 2 :
   Node 1 via 1 Distance 2
   Node 2 via 2 Distance 0
   Node 3 via 3 Distance 1

For router 3 :
   Node 1 via 2 Distance 3
   Node 2 via 2 Distance 1
   Node 3 via 3 Distance 0
```

12)AIM: Write a program to implement Broadcast tree by taking subnet of hosts.
DESCRIPTION: This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers the algorithm just finds the broadcast between them on the graph
SOURCE CODE:
```
#include<stdio.h>
struct ed
{
	int v1,v2,w;
}
edj[20],temp;
int main()
{
	int i,j,n=0,s,d,par[20],s1,d1;
	printf("Enter no of edges :");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the node1,node2,Weight :");
		scanf("%d %d %d",&edj[i].v1,&edj[i].v2,&edj[i].w);
		par[i]=0;
	}
	for(i=0;i<n;i++)
	{
		for(j=0;j<=i;j++)
			if(edj[j].w>edj[i].w)
			{
				temp=edj[i];
				edj[i]=edj[j];
				edj[j]=temp;
			}
	}
	printf("\nENTERED VALUES\n");
	for(i=0;i<n;i++)
	printf("%d  %d  %d\n",edj[i].v1,edj[i].v2,edj[i].w);
	printf("\nBROADCAST TREE FOR THE GIVEN GRAPH\n");
	for(i=0;i<n;i++)
	{
		s=edj[i].v1;
		d=edj[i].v2;
		s1=s;d1=d;
		while(par[s1]>0)
			s1=par[s1];
		while(par[d1]>0)
			d1=par[d1];
		if(s1!=d1)
		{
			par[d]=s;
			printf("%d  %d  %d\n",s,d,edj[i].w);
		}
	}
}
```

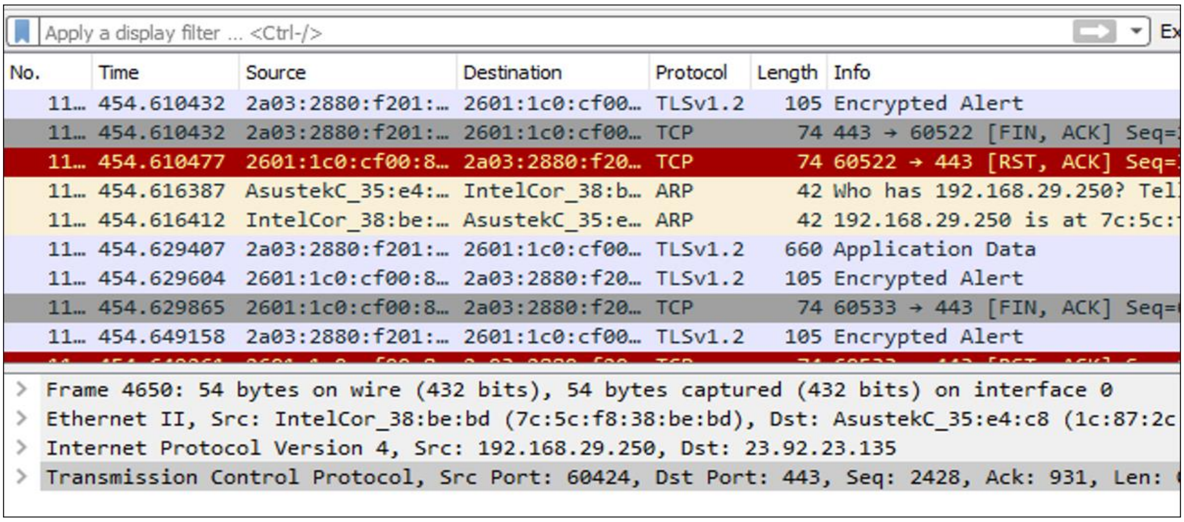
OUTPUT:

```
Enter no of edges :4
Enter the node1,node2,Weight :4 6 2
Enter the node1,node2,Weight :3 5 7
Enter the node1,node2,Weight :2 4 1
Enter the node1,node2,Weight :6 3 5

ENTERED VALUES
2  4  1
4  6  2
6  3  5
3  5  7

BROADCAST TREE FOR THE GIVEN GRAPH
2  4  1
4  6  2
6  3  5
3  5  7
```

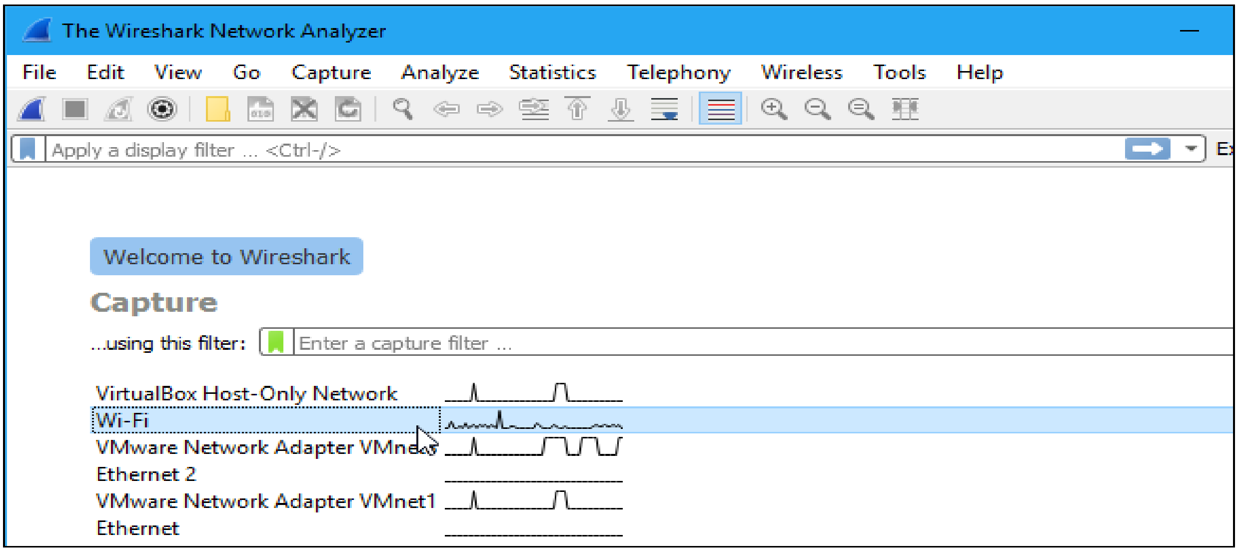
13. Use Wireshark to Capture, Filter and Inspect Packets



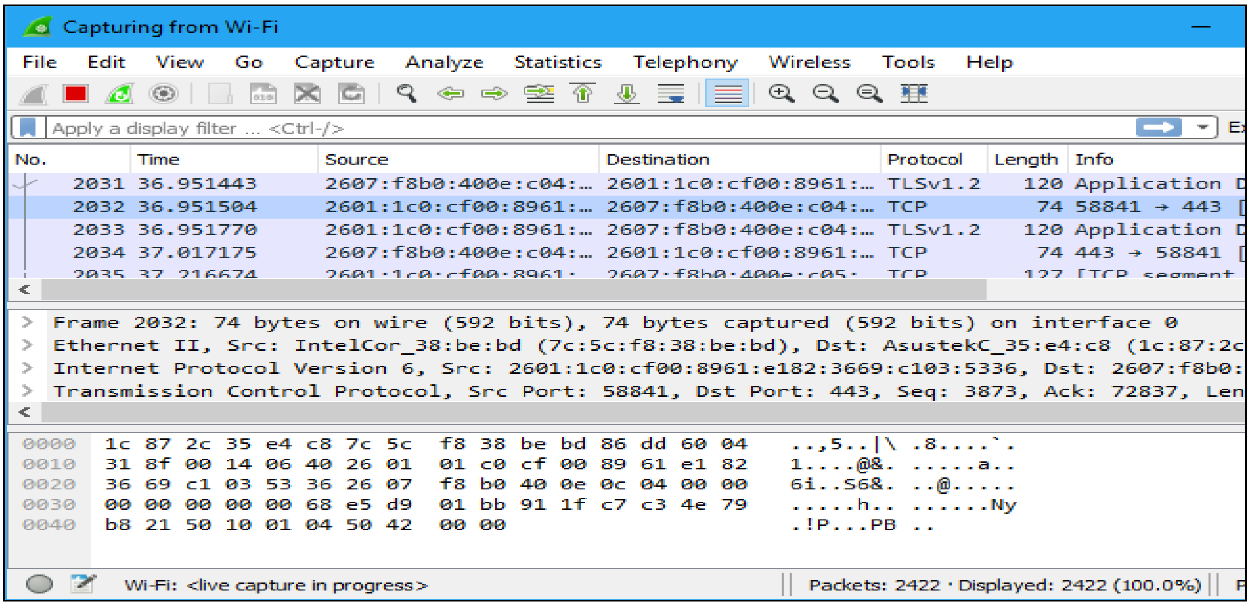
Wireshark, a network analysis tool formerly known as Ethereal, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets.

Capturing Packets

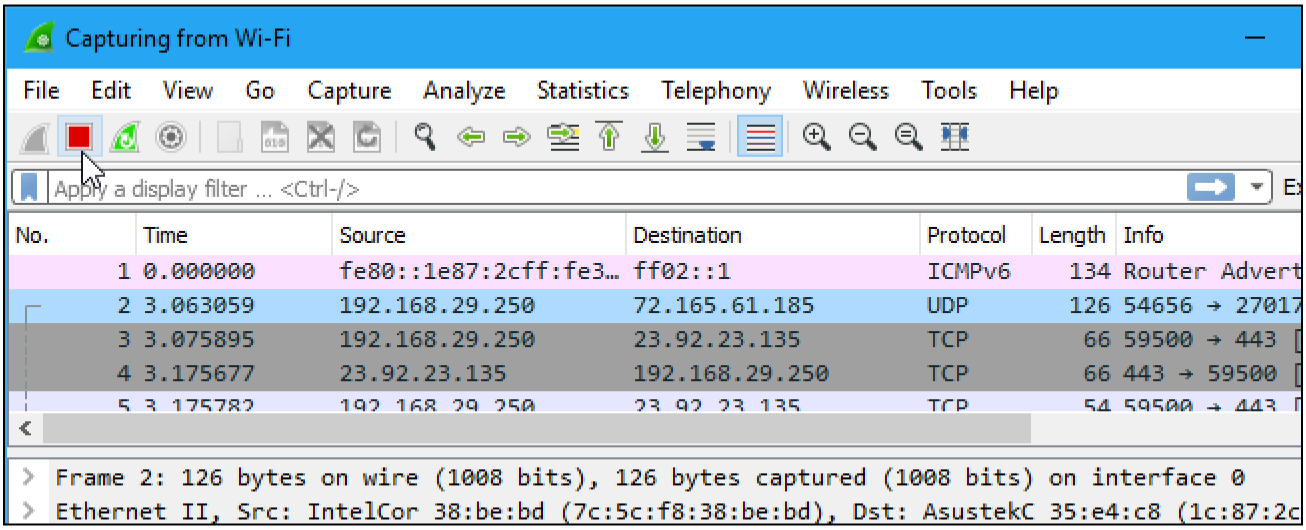
After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.



ADVERTISEMENT
As soon as you click the interface's name, you'll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.



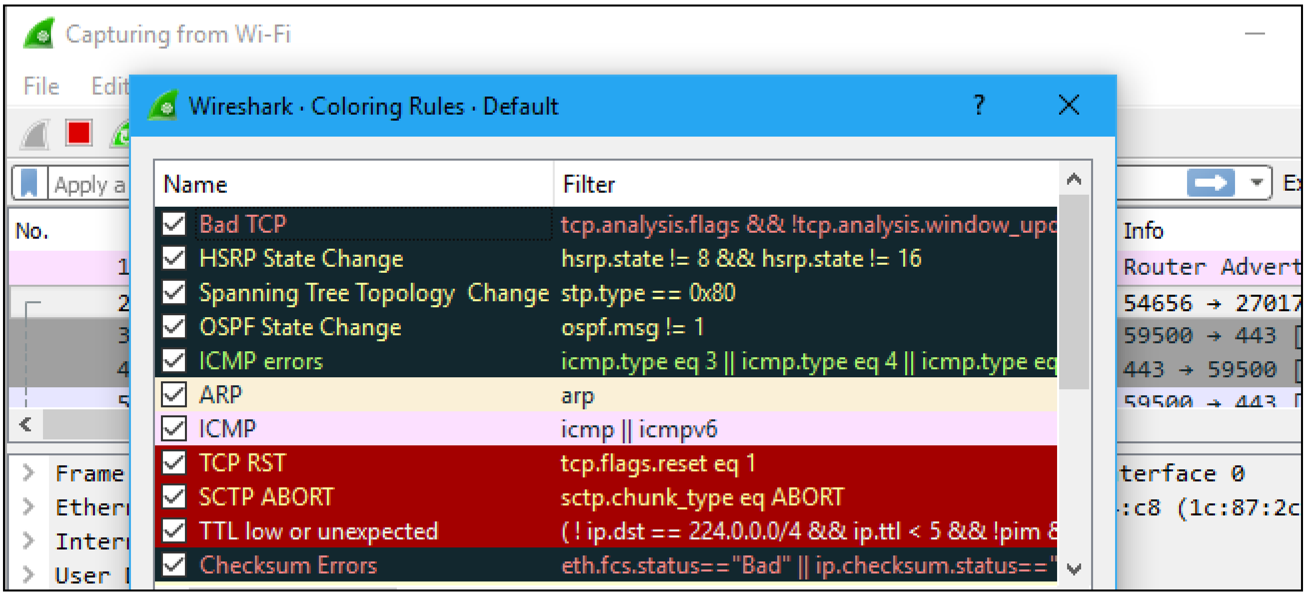
Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.



Color Coding

Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order.

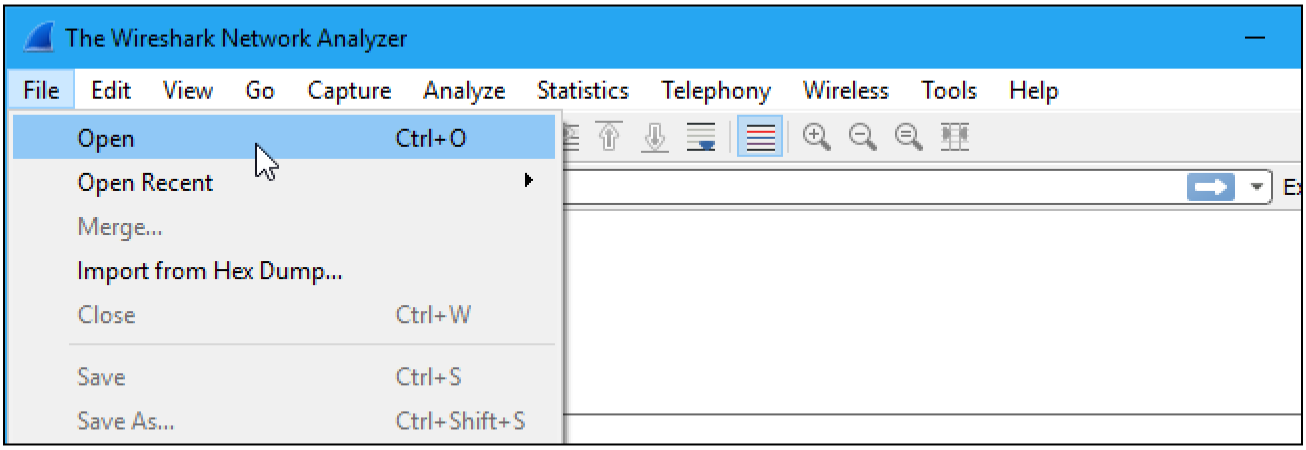
To view exactly what the color codes mean, click View > Coloring Rules. You can also customize and modify the coloring rules from here, if you like.



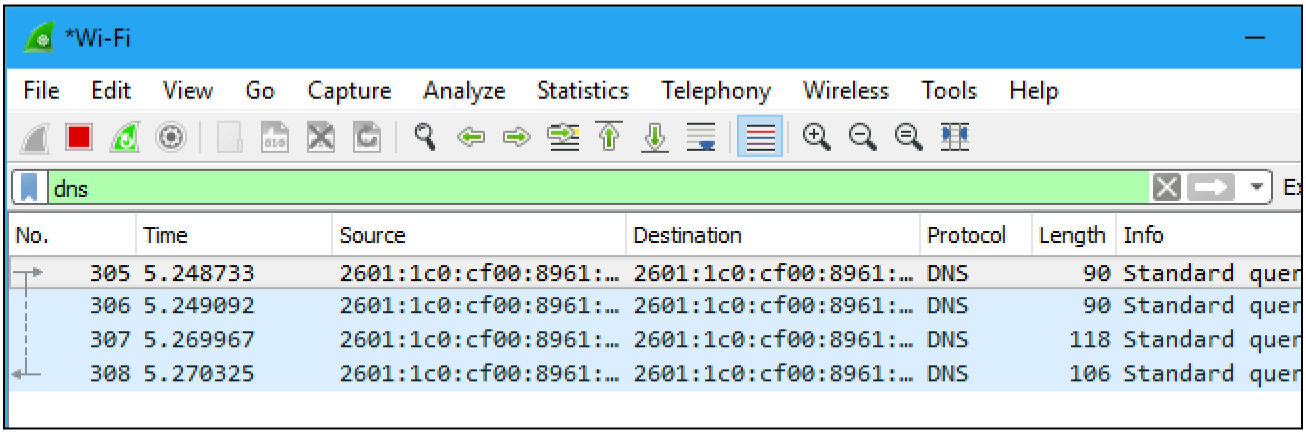
Sample Captures

If there's nothing interesting on your own network to inspect, Wireshark's wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one.

also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.

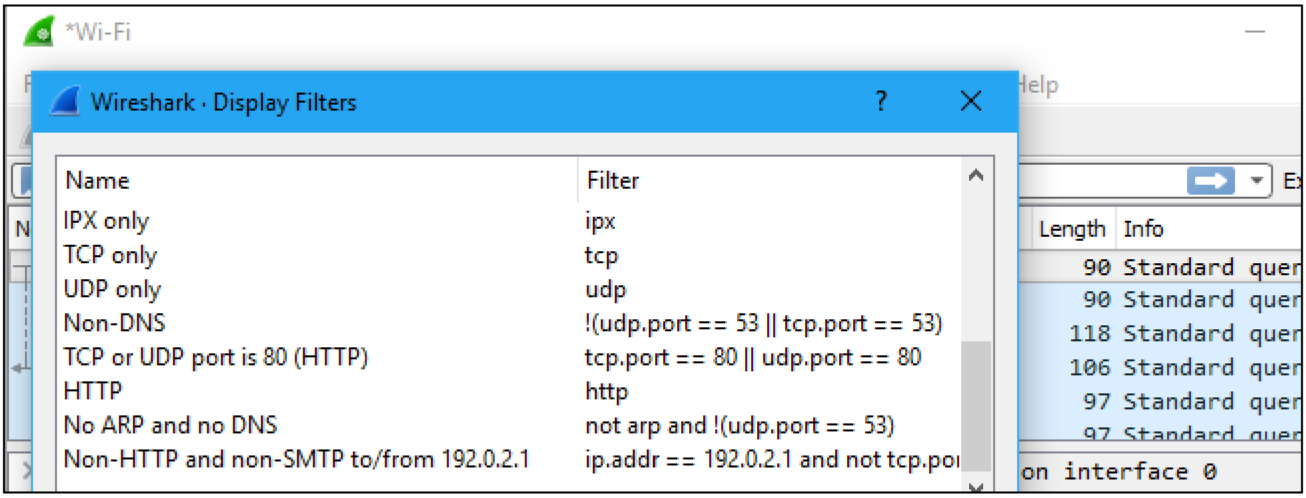


Filtering Packets

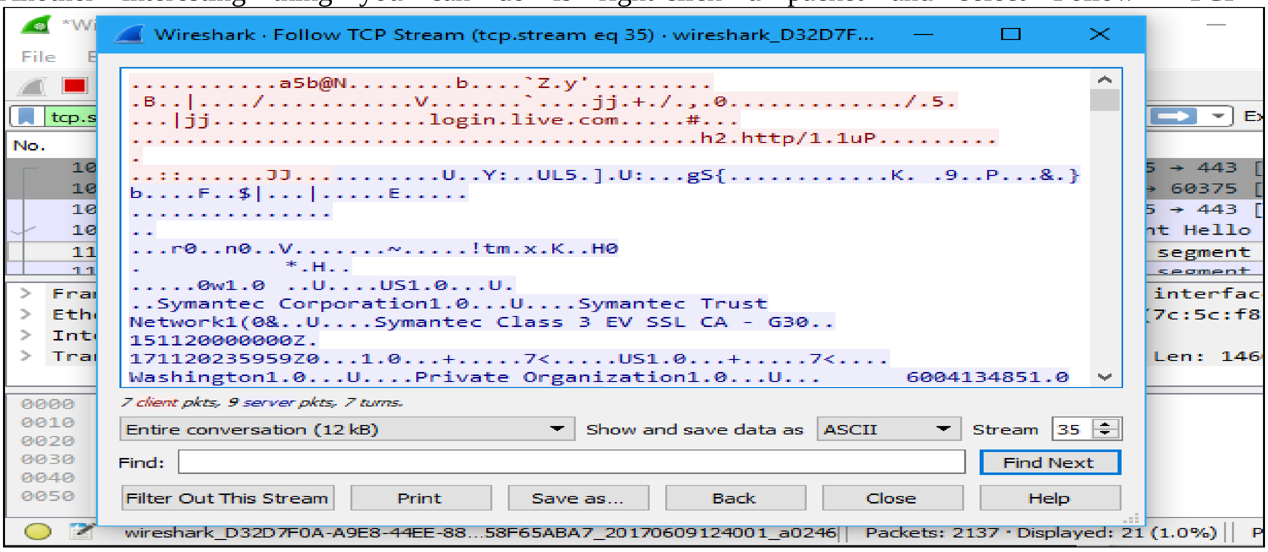


You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future.

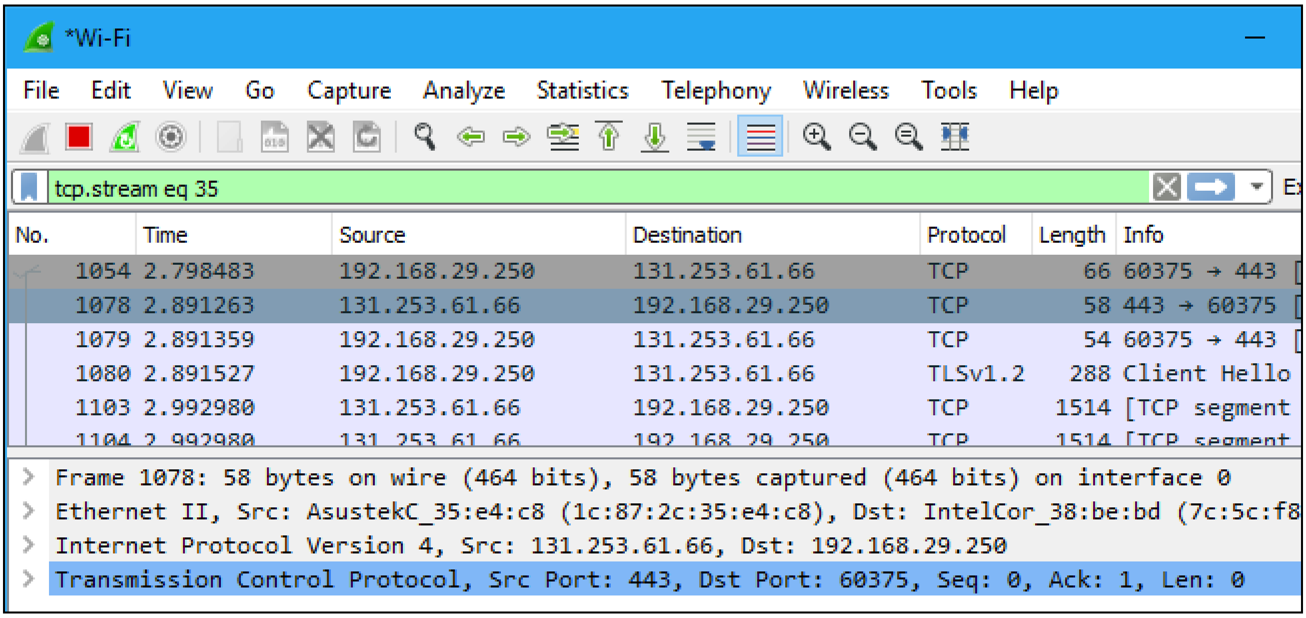
For more information on Wireshark's display filtering language, read the Building display filter expressions page in the official Wireshark documentation.



Another interesting thing you can do is right-click a packet and select Follow > TCP Stream.

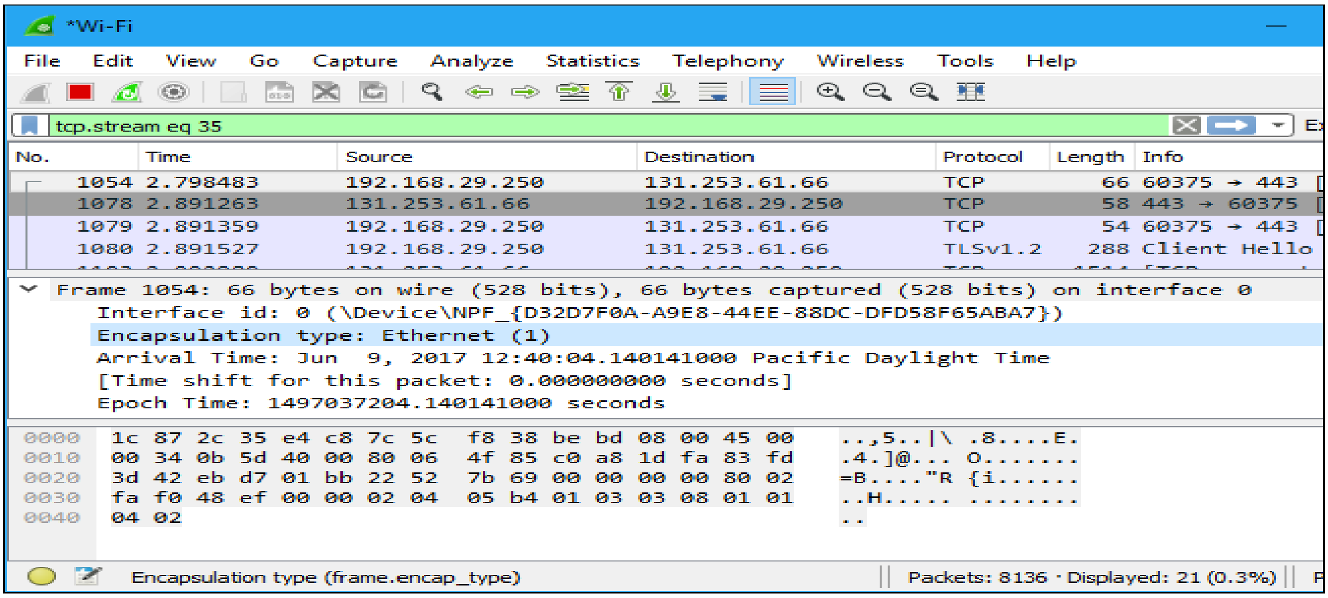


Close the window and you'll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

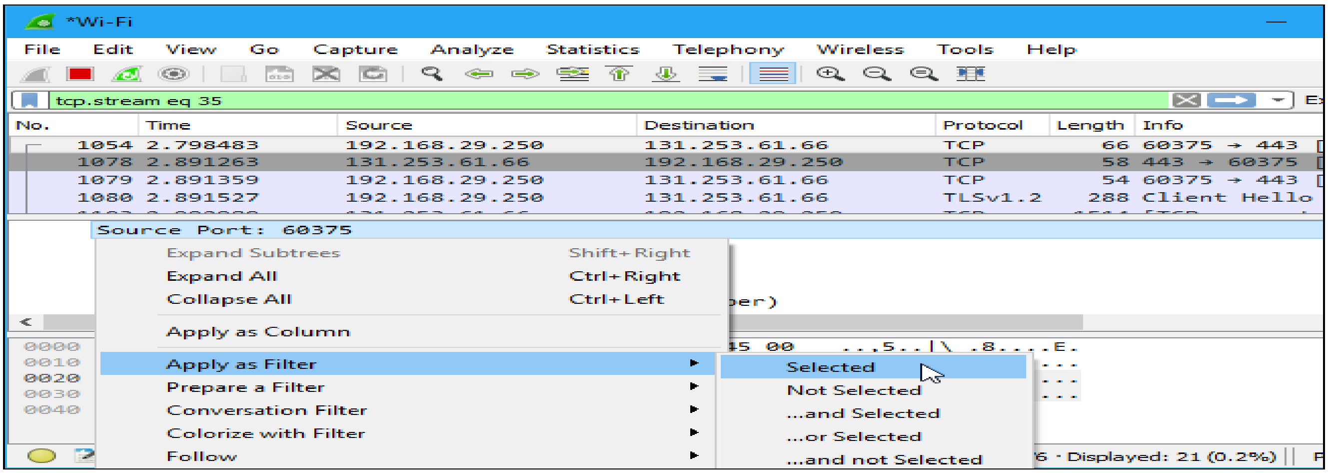


Inspecting Packets

Click a packet to select it and you can dig down to view its details.



You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.

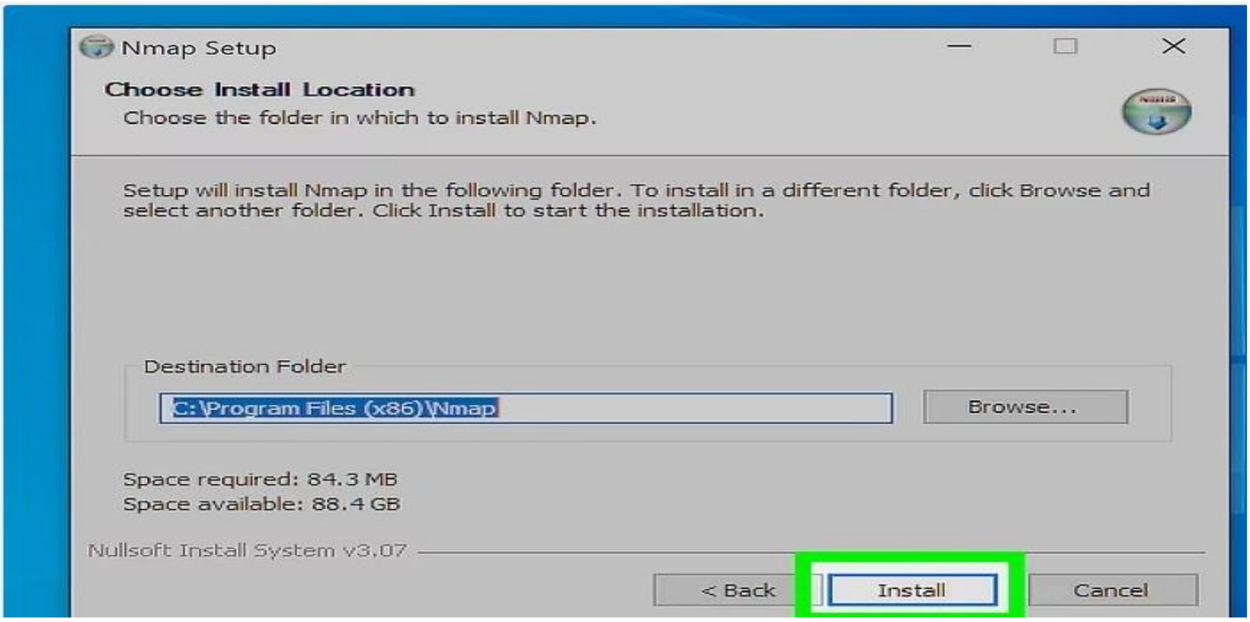


Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

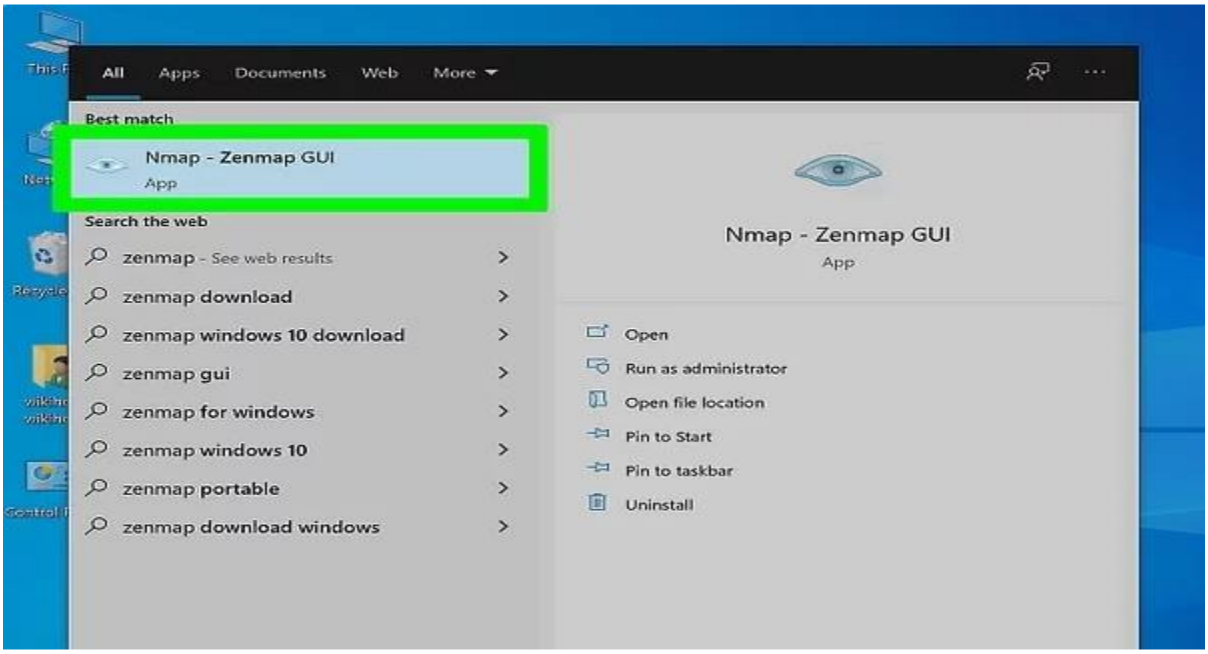
14. How to Run a Simple Nmap Scan

Step 1: Download the Nmap installer. This can be found for free from the developer's website. It is highly recommended that you download directly from the developer to avoid any potential viruses or fake files. Downloading the Nmap installer includes Zenmap, the graphical interface for Nmap which makes it easy for newcomers to perform scans without having to learn command lines.

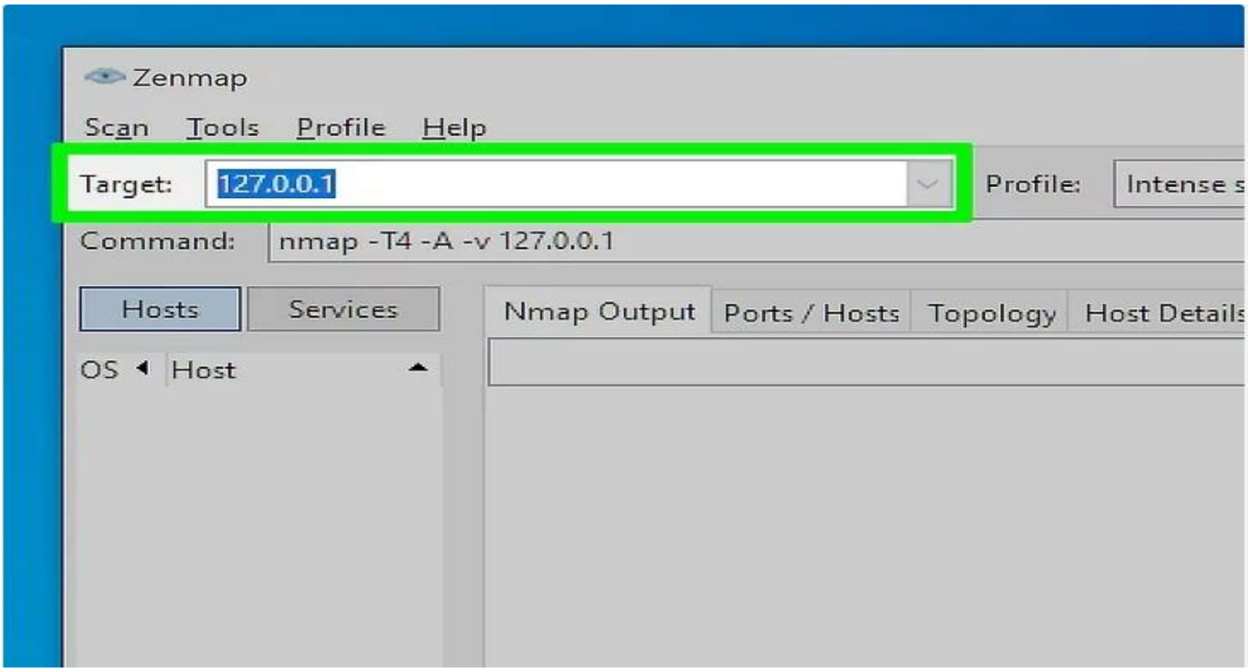
- The Zenmap program is available for Windows, Linux, and Mac OS X. You can find the installation files for all operating systems on the Nmap website.



Step 2: Install Nmap. Run the installer once it is finished downloading. You will be asked which components you would like to install. In order to get the full benefit of Nmap, keep all of these checked. Nmap will not install any adware or spyware.

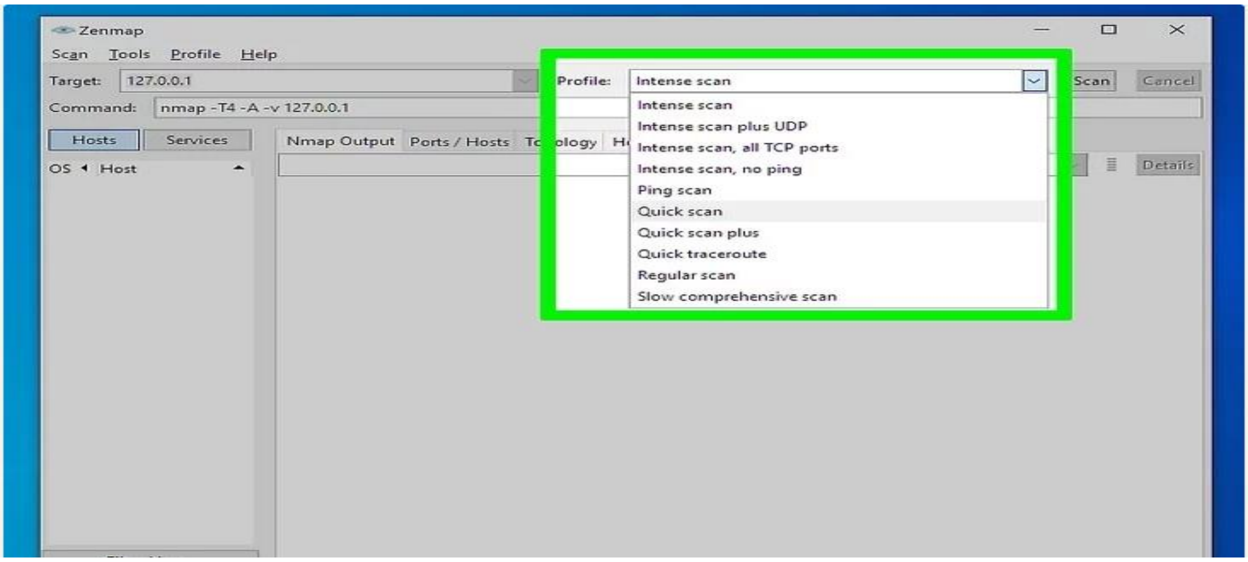


Step 3: Run the "Nmap – Zenmap" GUI program. If you left your settings at default during installation, you should be able to see an icon for it on your desktop. If not, look in your Start menu. Opening Zenmap will start the program.



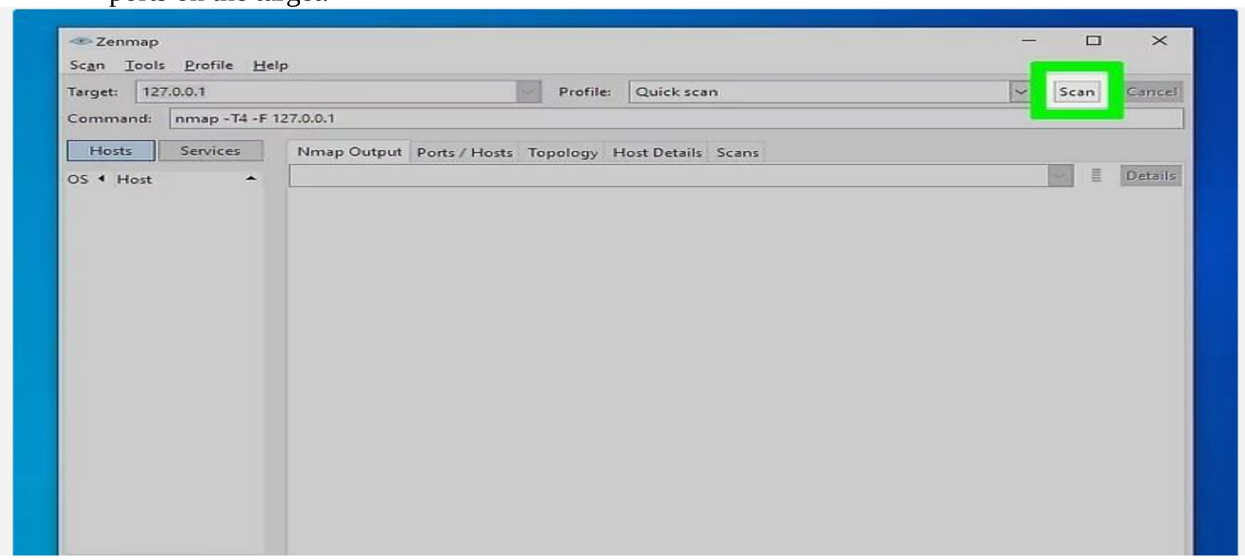
Step 4: Enter in the target for your scan. The Zenmap program makes scanning a fairly simple process. The first step to running a scan is choosing your target. You can enter a domain (example.com), an IP address (127.0.0.1), a network (192.168.1.0/24), or a combination of those.

Depending on the intensity and target of your scan, running an Nmap scan may be against the terms of your internet service provider, and may land you in hot water. Always check your local laws and your ISP contract before performing Nmap scans on targets other than your own network.

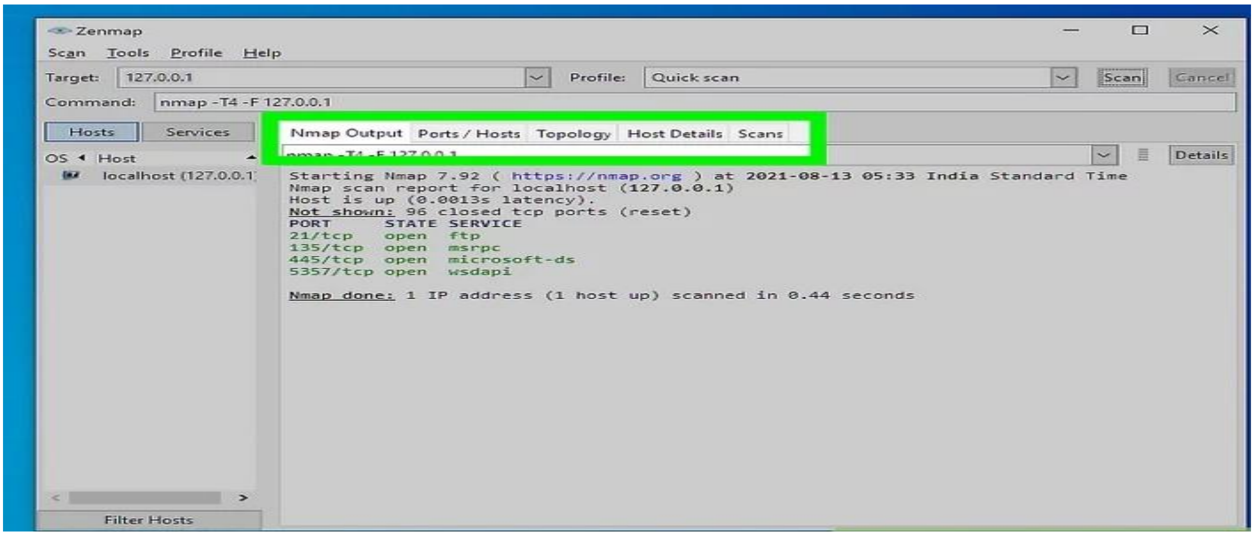


Step 5: Choose your Profile. Profiles are preset groupings of modifiers that change what is scanned. The profiles allow you to quickly select different types of scans without having to type in the modifiers on the command line. Choose the profile that best fits your needs:[1]

- **Intense scan -** A comprehensive scan. Contains Operating System (OS) detection, version detection, script scanning, traceroute, and has aggressive scan timing. This is considered an intrusive scan.
- **Ping scan -** This scan simply detects if the targets are online, it does not scan any ports.
- **Quick scan -** This is quicker than a regular scan due to aggressive timing and only scanning select ports.
- **Regular scan -** This is the standard Nmap scan without any modifiers. It will return ping and return open ports on the target.



Step 6: **Click Scan to start scanning.** The active results of the scan will be displayed in the Nmap Output tab. The time the scan takes will depend on the scan profile you chose, the physical distance to the target, and the target's network configuration.



Step 7: **Read your results.** Once the scan is finished, you'll see the message "Nmap done" at the bottom of the Nmap Output tab. You can now check your results, depending on the type of scan you performed. All of the results will be listed in the main Nmap Output tab, but you can use the other tabs to get a better look at specific data.[2]

- **Ports/Hosts -** This tab will show the results of your port scan, including the services for those ports.
- **Topology -** This shows the traceroute for the scan you performed. You can see how many hops your data goes through to reach the target.
- **Host Details -** This shows a summary of your target learned through scans, such as the number of ports, IP addresses, hostnames, operating systems, and more.
- **Scans -** This tab stores the commands of your previously-run scans. This allows you to quickly re-scan with a specific set of parameters.

15. Operating System Detecting using Nmap

Service and OS detection

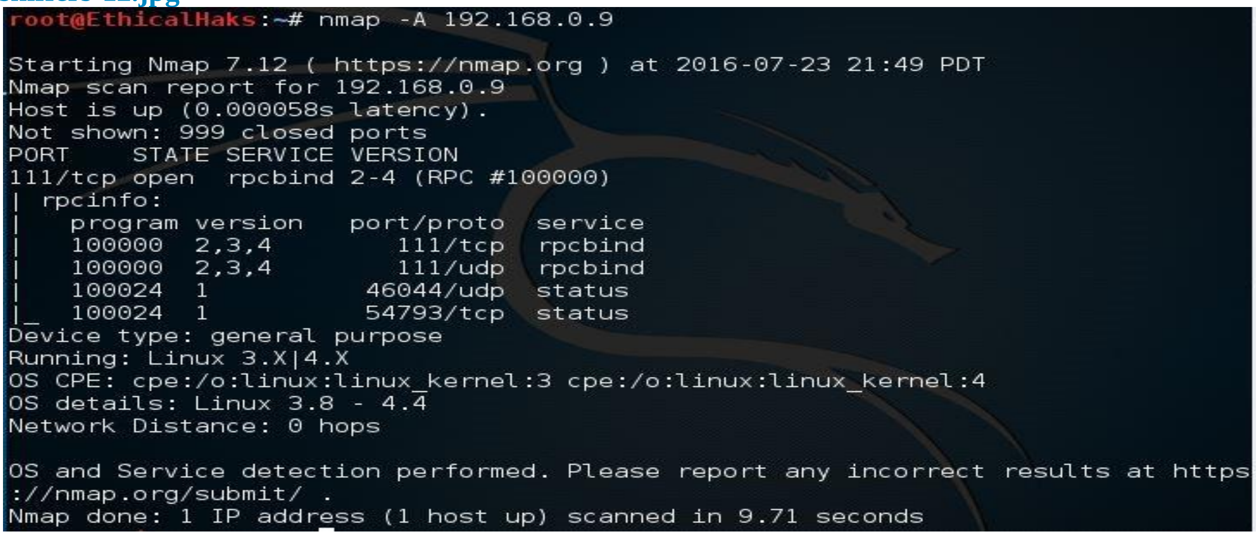
Nmap is one of the most popular tools used for the enumeration of a targeted host. Nmap can use scans that provide the OS, version, and service detection for individual or multiple devices. Detection scans are critical to the enumeration process when conducting penetration testing of a network. It is 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

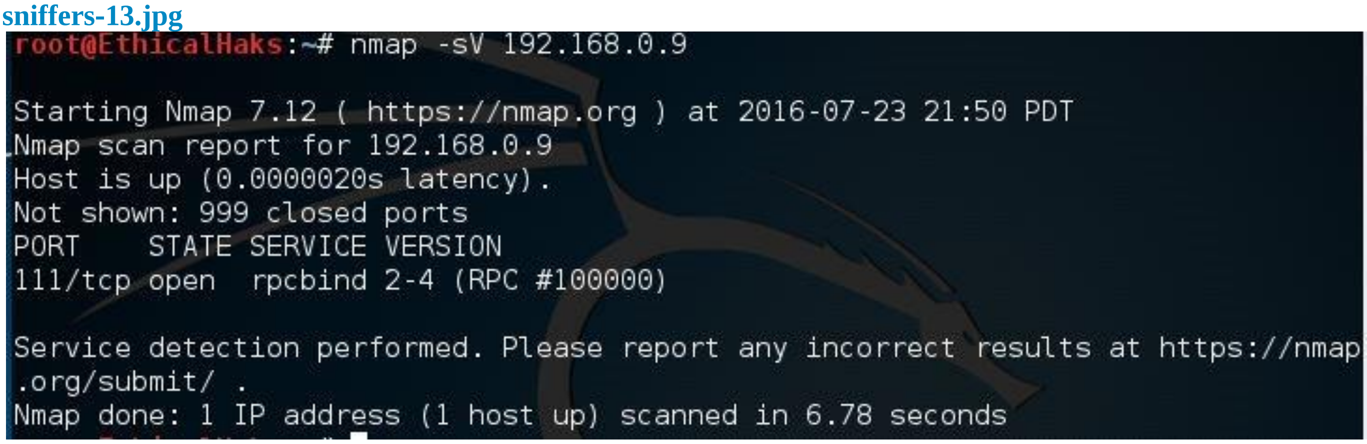


Standard service detection

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



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

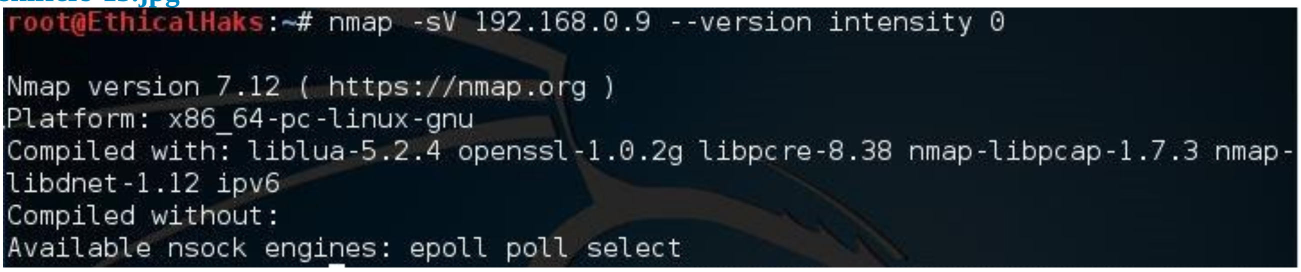
sniffers-14.jpg

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



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.

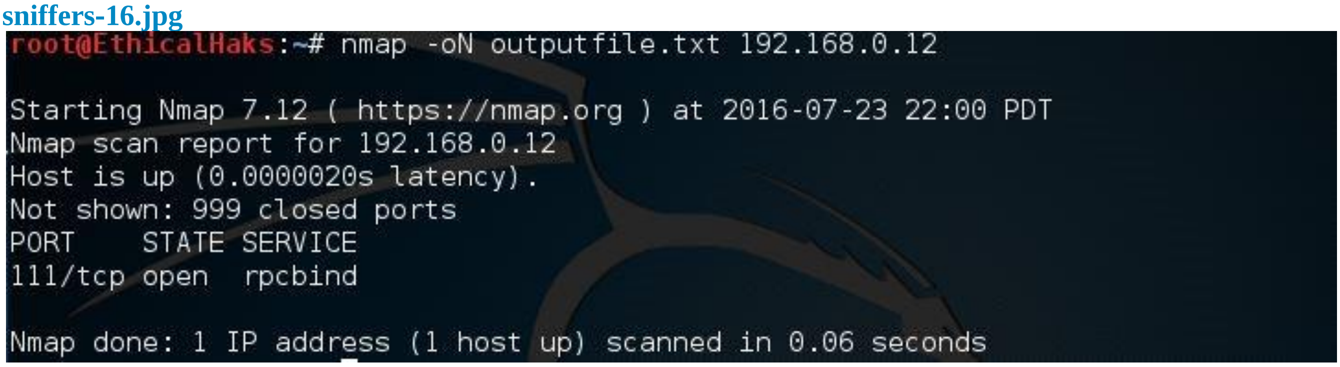
Nmap output formats

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

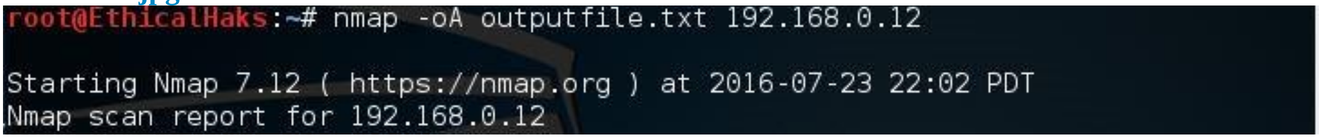


Save in all formats

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

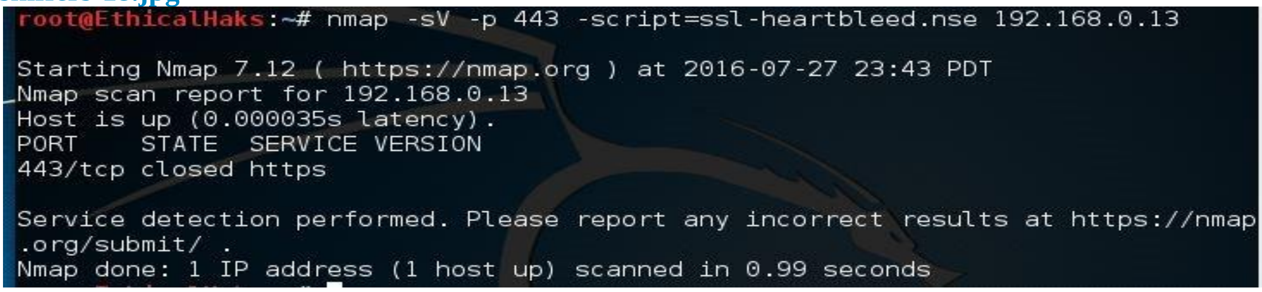


Scan using a specific NSE script

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



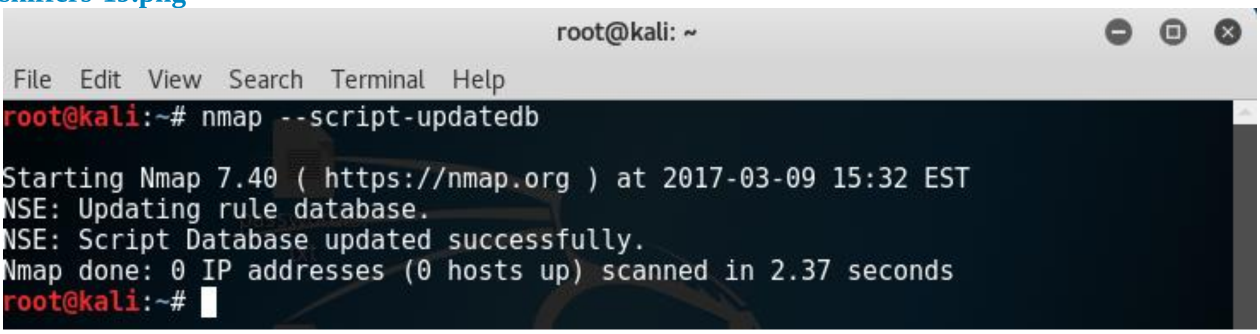
Scan with a set of scripts

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

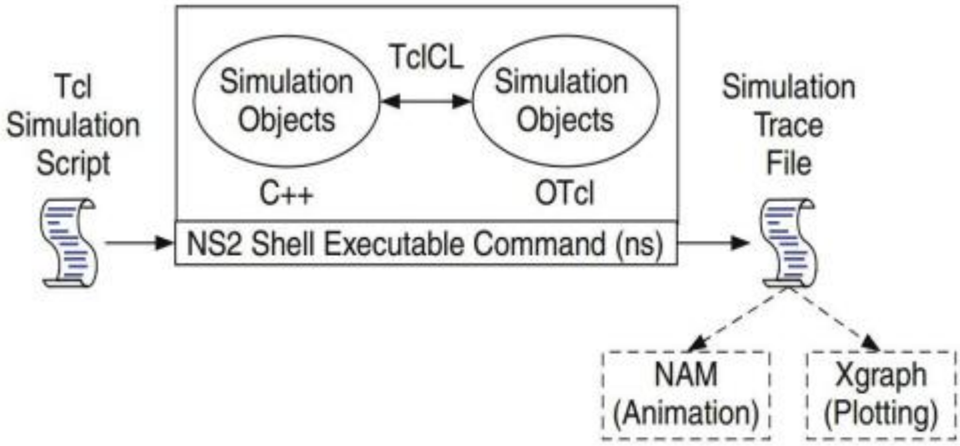
It is important to keep an updated database of current scripts. To update the Nmap script database, type the command nmap - -script-updatedb . The following screenshot demonstrates the screen you will see when you run this command:

sniffers-19.png



16. Do the following using NS2 Simulator
i. NS2 Simulator- Introduction
AIM: To study about NS2 simulator in detail.
THEORY: Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity

of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project . Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers and developers in the community are constantly working to keep NS2 strong and versatile.
BASIC ARCHITECTURE:



NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.
In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend).

Tcl scripting

Tcl is a general purpose scripting language. [Interpreter]
• Tcl runs on most of the platforms such as Unix, Windows, and Mac.
• The strength of Tcl is its simplicity.
• It is not necessary to declare a data type for variable prior to the usage.

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

puts stdout{Hello, World!} Hello, World!

Variables Command Substitution set
a 5 set len [string length foobar]
set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components

Create the event scheduler
Open new files & turn on the tracing
Create the nodes
Setup the links

Configure the traffic type (e.g., TCP, UDP, etc) Set the time of traffic generation (e.g., CBR, FTP) Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.
2. Definition of network nodes, links, queues and topology.
3. Definition of agents and of applications.
4. The nam visualization tool.
5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script. This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using ―open command:

#Open the Trace file

```
set tracefile1 [open out.tr w]
$ns trace-all $tracefile1
```

#Open the NAM trace file

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
```

The above creates a dta trace file called out.tr and a nam visualization trace file called out.nam. Within the tcl script, these files are not called explicitly by their names, but instead by pointers that aredeclared above and called ―tracefile1 and ―namfile respectively. Remark that they begins with a # symbol. The second line open the file ―out.tr to be used for writing, declared with the letter ―w. The third line uses a simulator method called trace-all that have as parameter the name of the file where thetraces will go.

Define a "finish" procedure
Proc finish { } {
global ns tracefile1 namfile
$ns lush-trace
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
}

Definition of a network of links and nodes

The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definitionof a link is:

$ns duplex-link $n0 $n2 10Mb 10ms DropTail

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagationdelay and a capacity of 10Mb per sec for each direction.
To define a directional link instead of a bi-directional one, we should replace ―duplex-link by ―simplex-link.
In ns, an output queue of a node is implemented as a part of each link whose input is that node. Weshould also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20

$ns queue-limit $n0 $n2 20

FTP over TCP

TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by thedestination to know whether packets are well received.
There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type ofagent appears in the first line:

set tcp [new Agent/TCP]

The command **$ns attach-agent $n0 $tcp** defines the source node of the tcp connection.
The command **set sink [new Agent /TCPSink]** Defines the behavior of the destination node of TCPand assigns to it a pointer called sink.

#Setup a UDP

connection set

udp [new

Agent/UDP]

$ns attach-agent
$n1 $udp set null
[new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid_2

#setup a CBR over UDP connection
The below shows the definition of a CBR application using a UDP agent
The command **$ns attach-agent $n4 $sink** defines the destination node. The command **$ns connect $tcp $sink** finally makes the TCP connection between the source and destination nodes.

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set packetsize_ 100

$cbr set rate_ 0.01Mb

$cbr set random_ false

TCP has many parameters with initial fixed defaults values that can be changed if mentioned explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be changed toanother value, say 552bytes, using the command **$tcp set packetSize_ 552**.