**Computer Networks Lab**

**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 – CRC 12, CRC 16 and CRC CCIP.

6. Write a Program to implement Sliding window protocol for Go-back -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

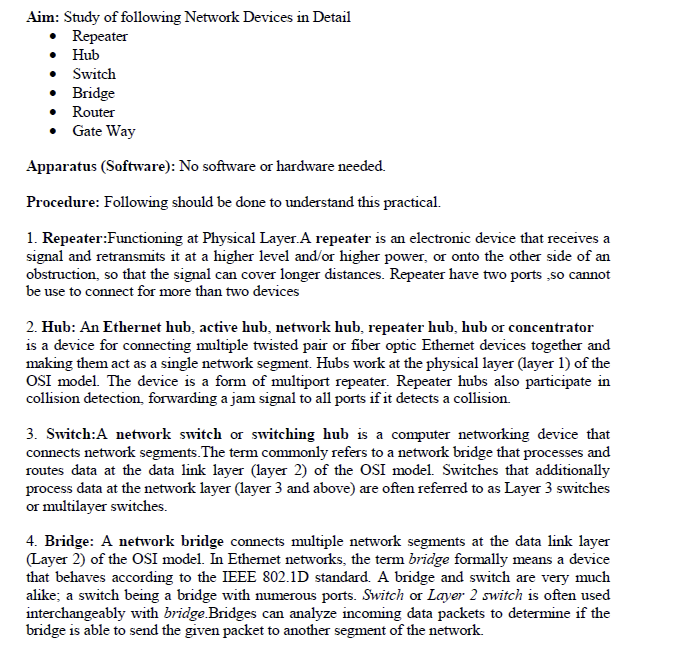
ii. Simulate to Find the Number of Packets Dropped

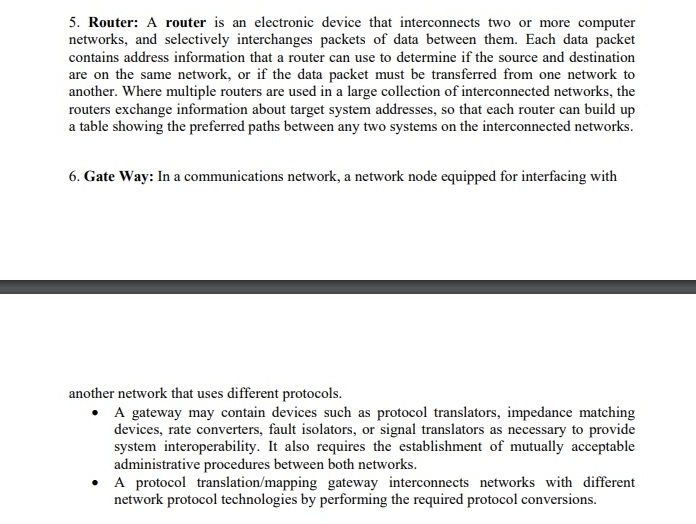
iii. Simulate to Find the Number of Packets Dropped by TCP/UDP

iv. Simulate to Find the Number of Packets Dropped due to Congestion

v. Simulate to Compare Data Rate& Throughput.

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.**

**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];

int i, j, p = 0, q = 0;

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);

getch();

}

**ii) bit stuffing**

#include<stdio.h>

#include<string.h>

int main() {

int a[20], b[30], i, j, k, count, n;

printf("Enter frame size in bits:");

scanf("%d", & n);

printf("Enter the frame in the form of 0 and 1 :");

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

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

i = 0;

count = 1;

j = 0;

while (i < n) {

if (a[i] == 1) {

b[j] = a[i];

for (k = i + 1; a[k] == 1 && k < n && count < 5; k++) {

j++;

b[j] = a[k];

count++;

if (count == 5) {

j++;

b[j] = 0;

}

i = k;

}

} else {

b[j] = a[i];

}

i++;

j++;

}

printf("After Bit Stuffing :");

printf("0 1 1 1 1 1 1 0 ");

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

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

printf("0 1 1 1 1 1 1 0");

return 0;

}

**3. Write a Program to implement data link layer farming method checksum.**

#include<stdio.h>

#include<string.h>

int main()

{

char a[20],b[20];

char sum[20],complement[20];

int i,length;

printf("Enter first binary string\n");

scanf("%s",&a);

printf("Enter second binary string\n");

scanf("%s",&b);

if(strlen(a)==strlen(b)){

length = strlen(a);

char carry='0';

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;

}

printf("\nSum=%c%s",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';

printf("\nChecksum=%c%s",carry,complement);

}

else {

printf("\nWrong input strings");

}

}

**Sample input & output:**

**Enter first binary string**

**101100**

**Enter second binary string**

**111000**

**Sum=1100100**

**Checksum=0011011**

**4. Write a program for Hamming Code generation for error detection and correction.**

// C program for the above approach

#include <math.h>

#include <stdio.h>

// Store input bits

int input[32];

// Store hamming code

int code[32];

int ham\_calc(int, int);

void solve(int input[], int);

// Function to calculate bit for

// ith position

int ham\_calc(int position, int c\_l)

{

int count = 0, i, j;

i = position - 1;

// Traverse to store Hamming Code

while (i < c\_l) {

for (j = i; j < i + position; j++) {

// If current boit is 1

if (code[j] == 1)

count++;

}

// Update i

i = i + 2 \* position;

}

if (count % 2 == 0)

return 0;

else

return 1;

}

// Function to calculate hamming code

void solve(int input[], int n)

{

int i, p\_n = 0, c\_l, j, k;

i = 0;

// Find msg bits having set bit

// at x'th position of number

while (n > (int)pow(2, i) - (i + 1)) {

p\_n++;

i++;

}

c\_l = p\_n + n;

j = k = 0;

// Traverse the msgBits

for (i = 0; i < c\_l; i++) {

// Update the code

if (i == ((int)pow(2, k) - 1)) {

code[i] = 0;

k++;

}

// Update the code[i] to the

// input character at index j

else {

code[i] = input[j];

j++;

}

}

// Traverse and update the

// hamming code

for (i = 0; i < p\_n; i++) {

// Find current position

int position = (int)pow(2, i);

// Find value at current position

int value = ham\_calc(position, c\_l);

// Update the code

code[position - 1] = value;

}

// Print the Hamming Code

printf("\nThe generated Code Word is: ");

for (i = 0; i < c\_l; i++) {

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

}

}

// Driver Code

void main()

{

// Given input message Bit

int N,i;

printf("enter number of bits:");

scanf("%d",&N);

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

{

printf("enter message bit %d :",i);

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

}

// Function Call

solve(input, N);

}

**Sample input & output:**

**enter number of bits:7**

**enter message bit 0 :1**

**enter message bit 1 :0**

**enter message bit 2 :1**

**enter message bit 3 :1**

**enter message bit 4 :0**

**enter message bit 5 :0**

**enter message bit 6 :1**

**The generated Code Word is: 10100111001**

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

#include<stdio.h>

#include<math.h>

#include<string.h>

main()

{

int i,j,k,m,n,cl;

char a[10],b[100],c[100];

//clrscr();

printf("\n ENTER POLYNANOMIAL:");

scanf("%s",a);

printf("\n ENTER THE FRAME:");

scanf("%s",b);

m=strlen(a);

n=strlen(b);

for(i=0;i<m;i++) /\* To eliminat first zeros in

polynomial \*/

{

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

{

m=m-i;

break;

}

}

for(k=0;k<m;k++) /\* To Adjust the polynomial

\*/ a[k]=a[k+i];

cl=m+n-1;

for(i=0;i<n;i++) /\* To copy the original frame to c[]\*/

c[i]=b[i];

for(i=n;i<cl;i++) /\* To add n-1 zeros at the end of frame \*/

c[i]='0';

c[i]='\0'; /\*To make it as a string \*/

for(i=0;i<n;i++) /\* To set polynomial remainder at end of c[]\*/

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

{

for(j=i,k=0;k<m;k++,j++)

if(a[k]==c[j])

c[j]='0';

else c[j]='1';

}

for(i=0;i<n;i++) /\* To copy original data in c[]\*/ c[i]=b[i];

printf("\n THE MESSAGE IS: %s",c);

getch();

}

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

#include<stdio.h>

int main()

{

int windowsize,sent=0,ack,i,c;

printf("enter window size\n");

scanf("%d",&windowsize);

while(1)

{

for( i = 0; sent < windowsize; i++)

{

printf("Frame %d has been transmitted.\n",sent);

sent++;

if(sent == windowsize)

break;

}

printf("\nPlease enter your choice : \n 1-for lost data or ack or delay ack \n 2-for exit\n chioce:");

scanf("%d",&c);

if(c == 1)

{

printf("\n enter the frame number:");

scanf("%d",&ack);

for( i = ack; i < windowsize; i++)

{

printf("Frame %d has been re transmitted.\n",i);

if(i == windowsize)

break;

}

}

else

return 0;

}

return 0;

}

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**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#define NOF\_PACKETS 10

int rand1(int a)

{

int rn = (rand() % 10) % a;

return rn == 0 ? 1 : rn;

}

int main()

{

int packet\_sz[NOF\_PACKETS], i, clk, b\_size, o\_rate, p\_sz\_rm=0, p\_sz, p\_time, op;

for(i = 0; i<NOF\_PACKETS; ++i)

packet\_sz[i] = rand1(6) \* 10;

for(i = 0; i<NOF\_PACKETS; ++i)

printf("\npacket[%d]:%d bytes\t", i, packet\_sz[i]);

printf("\nEnter the Output rate:");

scanf("%d", &o\_rate);

printf("Enter the Bucket Size:");

scanf("%d", &b\_size);

for(i = 0; i<NOF\_PACKETS; ++i)

{

if( (packet\_sz[i] + p\_sz\_rm) > b\_size)

if(packet\_sz[i] > b\_size)/\*compare the packet siz with bucket size\*/

printf("\n\nIncoming packet size (%dbytes) is Greater than bucket capacity (%dbytes)-PACKET REJECTED", packet\_sz[i], b\_size);

else

printf("\n\nBucket capacity exceeded-PACKETS REJECTED!!");

else

{

p\_sz\_rm += packet\_sz[i];

printf("\n\nIncoming Packet size: %d", packet\_sz[i]);

printf("\nBytes remaining to Transmit: %d", p\_sz\_rm);

p\_time = rand1(4) \* 10;

printf("\nTime left for transmission: %d units", p\_time);

for(clk = 10; clk <= p\_time; clk += 10)

{

sleep(1);

if(p\_sz\_rm)

{

if(p\_sz\_rm <= o\_rate)/\*packet size remaining comparing with output rate\*/

op = p\_sz\_rm, p\_sz\_rm = 0;

else

op = o\_rate, p\_sz\_rm -= o\_rate;

printf("\nPacket of size %d Transmitted", op);

printf("----Bytes Remaining to Transmit: %d", p\_sz\_rm);

}

else

{

printf("\nTime left for transmission: %d units", p\_time-clk);

printf("\nNo packets to transmit!!");

}

}

}

}

}

**10. Write a Program to implement Dijkstra‘s algorithm to compute the Shortest path through a graph.**

#include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()

{

int G[MAX][MAX],i,j,n,u;

printf("Enter no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

printf("\nEnter the starting node:");

scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

//pred[] stores the predecessor of each node

//count gives the number of nodes seen so far

//create the cost matrix

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

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

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

//initialize pred[],distance[] and visited[]

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

{

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1)

{

mindistance=INFINITY;

//nextnode gives the node at minimum distance

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

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i];

nextnode=i;

}

//check if a better path exists through nextnode

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

//print the path and distance of each node

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

if(i!=startnode)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

j=pred[j];

printf("<-%d",j);

}while(j!=startnode);

}

}

**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).**

/\*

Distance Vector Routing in this program is implemented using Bellman Ford Algorithm:-

\*/

#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("\nEnter the number of nodes : ");

scanf("%d",&nodes);//Enter the nodes

printf("\nEnter 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];//initialise the distance equal to cost matrix

rt[i].from[j]=j;

}

}

do

{

count=0;

for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the cost matrix

//and add the distance from k to node j

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

for(k=0;k<nodes;k++)

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{//We calculate the minimum distance

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("\n\n For router %d\n",i+1);

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

{

printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);

}

}

printf("\n\n");

getch();

}

**12. Write a Program to implement Broadcast tree by taking subnet of hosts.**

#include<stdio.h>

#include<conio.h>

int a[20][20],q[20],visited[20],n,i,j,f=0,r=-1;

void bfs(int v) {

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

if(a[v][i] && !visited[i])

q[++r]=i;

if(f<=r) {

visited[q[f]]=1;

bfs(q[f++]);

}

}

void main() {

int v;

//clrscr();

printf("\n Enter the number of vertices:");

scanf("%d",&n);

for (i=1;i<=n;i++) {

q[i]=0;

visited[i]=0;

}

printf("\n Enter graph data in matrix form:\n");

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

for (j=1;j<=n;j++)

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

printf("\n Enter the starting vertex:");

scanf("%d",&v);

bfs(v);

printf("\n The node which are reachable are:\n");

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

if(visited[i])

printf("%d\t",i); else

printf("\n Bfs is not possible");

getch();

}

**13. Wireshark**

**i. Packet Capture Using Wire shark**

**ii. Starting Wire shark**

**iii. Viewing Captured Traffic**

**iv. Analysis and Statistics & Filters.**

Wireshark is a network protocol analyzer, or an application that captures packets from a network connection, such as from your computer to your home office or the internet. Packet is the name given to a discrete unit of data in a typical Ethernet network.

Wireshark is the most often-used packet sniffer in the world. Like any other packet sniffer, Wireshark does three things:

1. **Packet Capture:** Wireshark listens to a network connection in real time and then grabs entire streams of traffic – quite possibly tens of thousands of packets at a time.
2. **Filtering:** Wireshark is capable of slicing and dicing all of this random live data using filters. By applying a filter, you can obtain just the information you need to see.
3. **Visualization:** Wireshark, like any good packet sniffer, allows you to dive right into the very middle of a network packet. It also allows you to visualize entire conversations and network streams.

**i. Packet Capture Using Wire shark**

Once you’ve installed Wireshark, you can start grabbing network traffic. But remember: To capture any packets, you need to have proper permissions on your computer to put Wireshark into promiscuous mode.

* + In a Windows system, this usually means you have administrator access.
  + In a Linux system, it usually means that you have root access.

As long as you have the right permissions, you have several options to actually start the capture. Perhaps the best is to select Capture >> Options from the main window. This will bring up the Capture Interfaces window, as shown below in Figure 4.

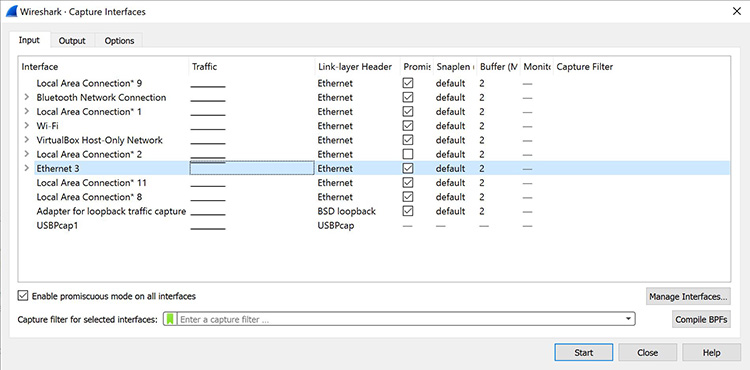


Figure 4: The Capture Interfaces dialog in Wireshark

This window will list all available interfaces. In this case, Wireshark provides several to choose from.

For this example, we’ll select the Ethernet 3 interface, which is the most active interface. Wireshark visualizes the traffic by showing a moving line, which represents the packets on the network.

**ii. Starting Wire shark**

You can download Wireshark for free at [www.wireshark.org](https://www.wireshark.org/). It’s also freely available, as an open source application under the [GNU General Public License](https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html) version 2.

How to Install Wireshark on Windows

If you’re a Windows operating system user, download the version appropriate for your particular version. If you use Windows 10, for example, you’d grab the 64-bit Windows installer and follow the wizard to install. To install, you’ll need administrator permissions.

**iii. Viewing Captured Traffic**

Once the network interface is selected, you simply click the Start button to begin your capture. As the capture begins, it’s possible to view the packets that appear on the screen, as shown in Figure 5, below.

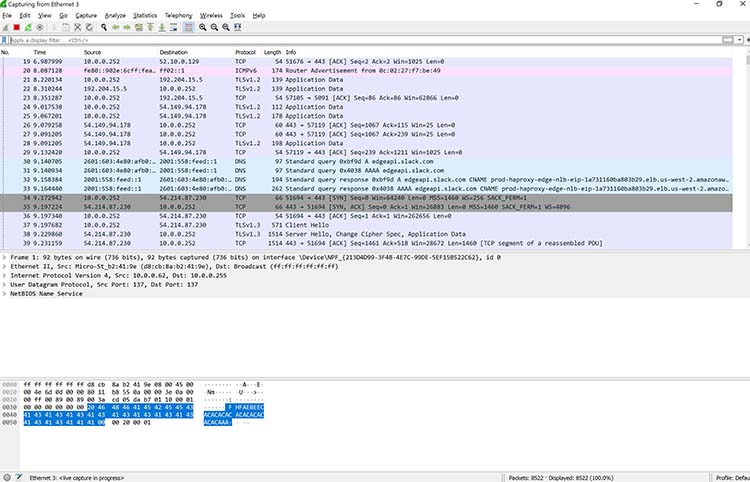


Figure 5: Wireshark capturing packets

Once you have captured all the packets that you want, simply click the red, square button at the top. Now you have a static packet capture to investigate.

**iv. Analysis and Statistics & Filters.**

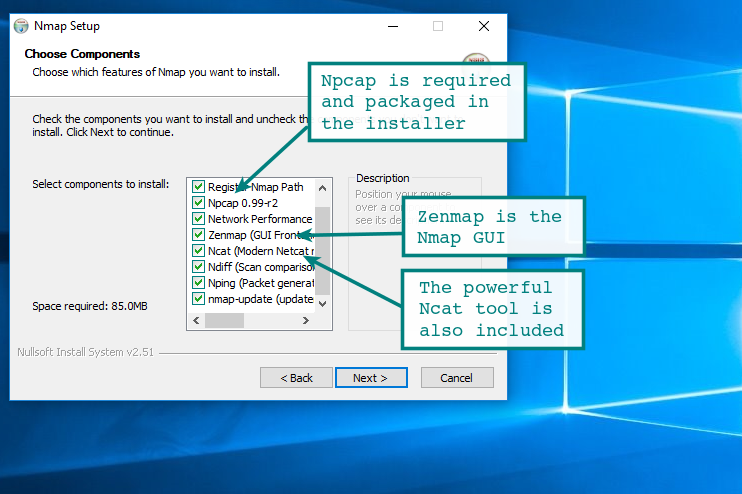
. It’s possible to view the input/output (I/O) statistics of an entire packet capture.

In Wireshark, just go to Statistics >> I/O Graph, and you’ll see a graph similar to the one shown in Figure 8.

**14. How to run Nmap scan**

**Installing Nmap for Windows**

To install the Windows version of Nmap [download](https://nmap.org/download.html) the executable installer and click through the wizard. It is your standard Next | Next | Next | finish... all done. By default, the Nmap installation directory will be added to the system path. With [Nmap](https://nmap.org/) in your system path, you can run nmap or ncat from any command window.

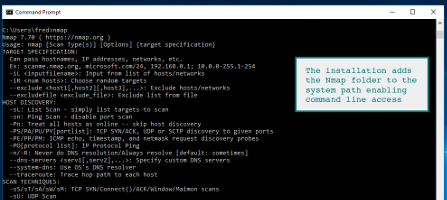


It will run on all the more modern versions of Windows including Windows 7, 2008 and Windows 10. If you are running something older such as 2K or earlier you may run into problems, but if you are still on those platforms you already have problems...

If you install from the zip file, there are a few additional configuration items to be aware of and apply. These are all documented on the [nmap](https://nmap.org/book/inst-windows.html) installation page for Windows.

**Nmap on the Windows Command Line**

During a default installation of the Nmap Windows package, the installation path will be added to the system path. Simply fire up a command prompt and launch nmap. If you installed from the standalone zip file, you need to add the installation folder to the system path manually through system properties.

[](https://hackertarget.com/wp-content/uploads/2018/05/nmap-windows-command-line.png)

As you can see the familiar Nmap command options appear after running the command. Access to the [Nmap NSE scripts](https://hackertarget.com/7-nmap-nse-scripts-recon/) is available as are all the standard options.

**15. Operating System Detection using Nmap**

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.

Type :  nmap -O 192.168.0.9

**16. Do the following using NS2 Simulator**

i. NS2 Simulator-Introduction

**Network simulation** (NS) is one of the types of simulation, which is used to simulate the networks such as in MANETs, VANETs, etc. It provides simulation for routing and multicast protocols for both wired and wireless networks. NS is licensed for use under version 2 of the GNU (General Public License) and is popularly known as **NS2**. It is an object-oriented, discrete event-driven simulator written in C++ and Otcl/Tcl.   
NS-2 can be used to implement network protocols such as TCP and UDP, traffic source behavior such as FTP, Telnet, Web, CBR, and VBR, router queues management mechanism such as Drop Tail, RED, and CBQ, routing algorithms, and many more. In ns2, C++ is used for detailed protocol implementation and Otcl is used for the setup. The compiled C++ objects are made available to the Otcl interpreter and in this way, the ready-made C++ objects can be controlled from the OTcl level.

ii. Simulate to Find the Number of Packets Dropped

* Define the trace files, and place monitors at places in the topology to collect information about packets flows. NS2 supports two primary monitoring capabilities: traces and monitors. The traces enable recording of packets whenever an event such as packet drop or arrival occurs in a queue or a link. The monitors provide a means for collecting quantities, such as number of packet drops or number of arrived packets in the queue. The monitor can be used to collect these quantities for all packets or just for a specified flow (a flow monitor).

# open the nam trace file  
set nam\_trace\_fd [open tcp\_tahoe.nam w]  
$ns namtrace-all $nam\_trace\_fd  
set trace\_fd [open tcp\_tahoe.tr w]

#Define a 'finish' procedure  
proc finish {} {  
        global ns nam\_trace\_fd trace\_fd

        # close the nam trace file  
        $ns flush-trace  
        close $nam\_trace\_fd

        # execute nam on the trace file  
        exit 0  
}

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.