Computer Networks

LAB MANUAL

(Approvedby AICTE-NewDelhi, Accreditedby NAACwith'A' &
AffiliatedtoJNTU,Hyderabad)RecognisedUnderSection2(f)&12(B)oftheUGCact,1956
Dundigal,Quthbullapur(M),Hyderabad-500043

ProgrammeEducationalObjectives(PEO's)

ComputerScienceandEngineering

PEO1:	Establish a successful professional career in industry, government or
PEO2:	academia.Gainmultidisciplinaryknowledgeprovidingasustainablecompetitiveedgein higherstudiesorResearch.
PEO3:	Promotedesign, analyze,andexhibitofproducts,through strong communication, leadershipandethicalskills,tosucceedanentrepreneurial.

II.PROGRAMMEOUTCOMES(PO's)

PO1	**Engineeringknowledge:** Applytheknowledgeofmathematics,science, engineeringfundamentals,andanengineeringspecializationtothesolutionofcomplexengineeringproblems.
PO2	**Problemanalysis:**Identify,formulate,reviewresearchliterature,andanalyzecomplexengineeringproblemsreachingsubstantiatedconclusionsusingfirstprinciplesofmathematics,naturalsciences,andengineeringsciences.
PO3	**Design/developmentofsolutions:**Designsolutionsforcomplexengineeringproblems and design system components or processes that meet the specified needswithappropriateconsiderationforthepublichealthandsafety,andthecultural, societal,andenvironmentalconsiderations.
PO4	**Conduct investigationsofcomplex problems:**Useresearch-based knowledgeandresearchmethodsincludingdesignofexperiments,analysisandinterpretationofdata,andsynthesisoftheinformationtoprovidevalidconclusions.
PO5	**Modern tool usage :**Create,select, and apply appropriate techniques, resources,and modern engineering and IT tools including prediction and modeling to complexengineeringactivitieswithanunderstandingofthelimitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledgetoassesssocietal,health,safety,legalandculturalissuesandtheconsequentresponsibilitiesrelevanttotheprofessionalengineeringpractice.

PO7	**Environment andsustainability:**Understandtheimpactoftheprofessionalengineeringsolutionsinsocietalandenvironmentalcontexts,anddemonstratetheknowledgeandneedforsustainabledevelopment.
PO8	**Ethics:**Applyethicalprinciplesandcommittoprofessionalethicsandresponsibilities andnormsoftheengineeringpractice.
PO9	**Individualandteamwork:**Functioneffectivelyasanindividual,andasamemberorleaderindiverseteams,andinmultidisciplinarysettings.
PO10	**Communication:** Communicate effectively on complex engineering activities withthe engineering community and with society at large, such as, being able to comprehendandwriteeffectivereportsanddesigndocumentation,makeeffectivepresentations,andgiveandreceiveclearinstructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding oftheengineeringandmanagementprinciplesandapplythesetoone'sownwork,asamemberandleaderinateam,tomanageprojectsandinmultidisciplinaryenvironments.
PO12	**Life-longlearning:** Recognizetheneedfor,andhavethepreparationandabilitytoengageinindependentandlife-longlearninginthebroadestcontextoftechnological change.

MARRI LAXMAN REDDY
Institute of Technology & Management
(Autonomous)
(Approvedby AICTE-NewDelhi, Accreditedby NAACwith'A' &
AffiliatedtoJNTU,Hyderabad)RecognisedUnderSection2(f)&12(B)oftheUGCact,1956
Dundigal,Quthbullapur(M),Hyderabad-500043

COURSESTRUCTURE,OBJECTIVES

COURSESTRUCTURE

Computer Networks Labwillhaveacontinuousevaluation,duringFirstsemesterofSecond yearfor30internalmarksand70externalexaminationmarks.

Outofthe30 marksforinternalevaluation,day-to-dayworkinthelaboratorywillbeevaluatedfor15marksandinternalpracticalexaminationshallbeevaluatedfor15marksconductedbythe concernedfaculty.

The end semester examination will be conducted with an external examiner and internalexaminer.TheexternalexaminerwillbeappointedbythePrincipal.

COURSEOBJECTIVES

1. To understand the working principle of various communication protocols.
2. To understand the network simulator environment and visualize a network topology and observe its performance.
3. To analyze the traffic flow and the contents of protocol frames.

MARRI LAXMAN REDDY
Institute of Technology & Management
(Autonomous)
(Approvedby AICTE-NewDelhi, Accreditedby NAACwith'A' &
AffiliatedtoJNTU,Hyderabad)RecognisedUnderSection2(f)&12(B)oftheUGCact,1956
Dundigal,Quthbullapur(M),Hyderabad-500043

CourseOutcomes(CO's)

CO1: Implement data link layer farming methods.

CO2: Analyze error detection and error correction codes

CO3: Implement and analyze routing and congestion issues in network design.

CO4: Implement Encoding and Decoding techniques used in presentation layer

CO5: Work with different network tools.

CourseOutcomes(CO's)–ProgramOutcomes(PO's)Mapping

CO's / P0's	PO1	PO2	PO 3	PO 4	PO5	PO 6	PO7	PO 8	PO9	PO1 0	PO1 1	PO1 2	PSO 1	PSO 2
CO1	√													
CO2	√	√											√	
CO3	√	√	√				√			√				
CO4	√	√	√	√						√			√	
CO5	√	√	√	√			√			√			√	

Simple-1 **Moderate-2** **High-3**

MARRI LAXMAN REDDY
Institute of Technology & Management
(Autonomous)
(Approvedby AICTE-NewDelhi, Accreditedby NAACwith'A' &
AffiliatedtoJNTU,Hyderabad)RecognisedUnderSection2(f)&12(B)oftheUGCact,1956
Dundigal,Quthbullapur(M),Hyderabad-500043

INDEX

12	Operating System Detection using Nmap			
13	Do the following using NS2Simulator i. NS2Simulator-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. vi. Simulate to Plot Congestion for Different Source/Destination vii. Simulate to Determine the Performance with respect to Transmission of Packets			
ADD ON PROGRAMS				
1	Implement RSA			
2	Implement Rail Fence Cipher program			
3	Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.			
4	Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.			

Experiment :1

1. **Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.**

1 a. NAME OF THE EXPERIMENT: Character Stuffing.

OBJECTIVE: Implement the data link layer framing methods.

RESOURCE: Turbo C

PROGRAM LOGIC:

The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called character stuffing.

ALOGRITM:

Begin
Step 1: Initialize I and j as 0
Step 2: Declare n and pos as integer and a[20],b[50],ch as character
Step 3: read the string a
Step 4: find the length of the string n, i.e n-strlen(a)
Step 5: read the position, pos
Step 6: if pos > n then
Step 7: print invalid position and read again the position, pos
Step 8: endif
Step 9: read the character, ch
Step 10: Initialize the array b, b[0…5] as ’d’, ’l’, ’e’, ’s’, ’t’,’x’ respectively
Step 11: j=6;
Step 12: Repeat step[(13to22) until i<n
Step 13: if i==pos-1 then
Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’ respectively
Step 15: increment j by 7, i.e j=j+7
Step 16: endif
Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then
Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively
Step 19: increment j by 3, i.e j=j+3
Step 20: endif
Step 21: b[j]=a[i]
Step 22: increment I and j;
Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’,’e’,‘t’, ‘x’,‘\0’ respectively
Step 24: print frame after stuffing
Step 25: print b
End

1. **b. NAME OF THE EXPERIMENT**: **Bit Stuffing**.

OBJECTIVE: Implement the data link layer framing method.

RESOURCE: Turbo C

PROGRAM LOGIC:

The new technique allows data frames to contain an arbitrary number if bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data.

ALOGRITM:

Begin
Step 1: Read frame length n
Step 2: Repeat step (3 to 4) until i<n(: Read values into the input frame (0's and1's) i.e.
Step 3: initialize I i=0;
Step 4: read a[i] and increment i
Step 5: Initialize i=0, j=0,count =0
Step 6: repeat step (7 to 22) until i<n
Step 7: If a[i] == 1 then
Step 8: b[j] = a[i]
Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)
Step 10: Initialize k=i+1;
Step 11: Increment j and b[j]= a[k];
Step 12: Increment count ;
Step 13: if count =5 then
Step 14: increment j,
Step 15: b[j] =0
Step 16: end if
Step 17: i=k;
Step 18: increment k
Step 19: else
Step 20: b[j] = a[i]
Step 21: end if
Step 22: increment I and j
Step 23: print the frame after bit stuffing
Step 24: repeat step (25 to 26) until i< j
Step 25: print b[i]
Step 26: increment i
End

1. c. NAME OF THE EXPERIMENT: Character count

SOURCE CODE:

```
#include<stdio.h>
#include<string.h>
char data[20][20];
int n;
void main()
{
int i,ch,j;
char tmp[20][20];
printf("enter the number of frames:");
scanf("%d",&n);
for(i=0;i<=n;i++)
{
if(i!=0)
{
printf("frame%d:",i);
scanf("%s",&data[i]);
}
}
for(i=0;i<=n;i++)
{
tmp[i][0]=49+strlen(data[i]);
tmp[i][1]='\0';
strcat(tmp[i],data[i]);
}
printf("\n\t\t AT THE SENDER:\n");
printf("data as frames:\n");
for(i=1;i<=n;i++)
{
printf("frame%d:",i);
puts(tmp[i]);
}
printf("data transmitted:");
for(i=1;i<=n;i++)
printf("%s",tmp[i]);
printf("\n\t\t AT THE RECEIVER\n");
printf("the data received:");


for(i=1;i<=n;i++)
{
ch=(int)(tmp[i][0]-49);
for(j=1;j<=ch;j++)
```

```
data[i][j-1]=tmp[i][j];
data[i][j-1]='\0';
}
printf("\n the data after removing count char:");
for(i=1;i<=n;i++)
printf("%s",data[i]);
printf("\n the data in frame form:\n");
for(i=1;i<=n;i++)
{
printf("frame%d:",i);
puts(data[i]);
}
}
```

Viva Questions:

1. What is meant by data communication?
2. What are the components used in data communication?
3. How many data transfer modes are there?
4. Name the different types of communication between devices?
5. What is meant by protocol?
6. What is the use of point to point network?
7. What is multi point network?
8. What is meant by topology?
9. Name the different types of topologies?
10. What is the use of mesh topology?
11. What is the use of star topology?
12. What is the use of bus topology?
13. What is the use of ring topology
14. To link n devices fully connected mesh has how many physical channels are required?
15. How many ports are required in n devices mesh topology?
16. What are the advantages of mesh topology?
17. What are the disadvantages of mesh topology?
18. What are the advantages of star topology?
19. What are the disadvantages of star topology?
20. What are the advantages of bus topology?
21. What are the disadvantages of bus topology?
22. What are the advantages of ring topology?
23. What are the disadvantages of ring topology?
24. In which network bus topology is implemented?
25. What is meant by hybrid network?
26. Name the different categories in network connections?
27. What is the difference between protocol and standard?

28. Name the different categories in standards.
29. Abbreviate OSI and ISO
30. How many layers are there in OSI model
31. Name the different layers in OSI model
32. What is the full form of TCP/IP
33. How many layers are there in the TCP/IP
34. Differentiate between OSI model and TCP/IP
35. What is the use of physical layer/
36. What is Character stuffing?
37. What is the use of character stuffing?
38. ----------are the delimiters for the character stuffing?
39. Expand DLE STX?
40. Expand DLE ETX?
41. What is Stuffing?
42. What is use of Stuffing?
43. With bit stuffing the boundary between two frames can be unambiquously recognize by?
44. . ……..is a analogous to character stuffing?
45. The senders data link layer encounters……..no of 1's consecutively

Experiment: 2

2. **Write a program to compute CRC code for the polynomials CRC-12 and CRC-16**

NAME OF THE EXPERIMENT: Cyclic Redundancy Check.

OBJECTIVE: Implement on a data set of characters the three CRC polynomials – CRC 12,CRC 16
RESOURCE: Turbo C

PROGRAM LOGIC:
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 known as Cyclic Redundancy Code

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

ALGORITHM:

Step 1:A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1.
Step 2:The newly formed data unit 1. A string of n as is appended to the data unit. The length of predetermined divisor is n+ 1. i.e. original data + string of n as are divided by the divisor using binary division and remainder is obtained. This remainder is called CRC.
Step 3:Now, string of n Os appended to data unit is replaced by the CRC remainder (which is also of n bit).
Step 4: The data unit + CRC is then transmitted to receiver.
Step 5: The receiver on receiving it divides data unit + CRC by the same divisor & checks the remainder.
Step 6: If the remainder of division is zero, receiver assumes that there is no error in data and it accepts it.
Step 7: If remainder is non-zero then there is an error in data and receiver rejects it.

Viva Questions:

1. What is CRC?
2. What is the use of the CRC?
3. Name the CRC standards?
4. Define Checksum?
5. Define generator polynomial?

6. What are the techniques use for error correction and detection methods
7. Name the different techniques used for data error checking methods?
8. Name the different techniques used for data correcting methods?
9. What is the use of data link layer?
10. What is meant by framing?
11. How many types of framing methods are there?
12. What is meant by single bit error?
13. What is meant by burst error?
14. What is meant by redundancy bits?
15. What is meant by fixed frame- size?
16. What is meant by variable-size framing?
17. What is meant by byte stuffing?
18. What is the use of hamming code?
19. Who invented hamming code?
20. What is meant by even parity?
21. What is meant by odd parity?
22. How many bits are corrected by using hamming code?
23. What is meant by multiplexing?
24. Name the different types of multiplexing?
25. What is the use of TDM?
26. What is the use of WDM?
27. What is the use of TDM?
28. Which multiplexing combines optical signals?
29. How many types of switched networks are available?
30. Name the different types of switched networks?
31. What is meant by circuit switched network?
32. Name the different types of packet switched networks?
33. What are the three phases used in virtual circuit network?
34. How any types of transmission medias are there?
35. What is the use of transmission media?
36. In which network transmission media used?
37. Differentiate between guided and unguided media?
38. Which are called guided media?
39. Which are called unguided media?
40. How the data can be corrupted?

Experiment: 3

3. **Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism**.

3.a. Sliding window protocol

SOURCE CODE :

```
#include<stdio.h>
int main()
{
int w,i,f,frames[50];
printf("enter window size:");
scanf("%d",&w);
printf("\n Enter number of frames to transmmit:");
scanf("%d",&f);
printf("\n Enter %d frames:",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]);
printf("\n With sliding window protocol the frames will be sent in the following manner(assuming no corruption of frames)\n\n");
printf("After sending %d frames at each stage sender waits for acknowledgement sent by the receiver\n\n",w);
for(i=1;i<=f;i++)
{
if(i%w==0)
{
printf("%d\n",frames[i]);
printf("Acknowledgement of above frames sent is received by sender\n\n");
}
else printf("%d",frames[i]);
}
if(f%w!=0)
printf("\n Acknowledgement of above frames sent is received by sender\n");
return 0;
}
```

3.b. - Sliding window protocol using go-bach N:

SOURCE CODE :

```
#include<stdio.h>
int main()
{
	int windowsize,sent=0,ack,i;
	printf("enter window size\n");
	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.\n");
			scanf("%d",&ack);

			if(ack == windowsize)
				break;
			else
				sent = ack;
	}
return 0;
}
```

Experiment 4

4. Implement Dijsktra's algorithm to compute the shortest path through a network

RESOURCE: Turbo C

Program Logic: Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a SPT (shortest path tree) with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

ALGORITM:

1. In distance vector routing algorithm, the **cost** is considered as the **hop** count (number of networks passed to reach the destination node). So a cost between two neighbouring routers is set to 1.
2. Each router updates its routing table when it receives the information (distance vector) form the neighbouring routers.
3. After updating its routing table, a router must forward its result to its neighbouring router So that they can update their routing table.
4. Each router keeps the three information in its routing table i.e. **destination network**, **cost** & the **next hop**.
5. The router sends the information of each route as a record **R**.

Viva questions:

1. Define Dijkstra's algorithm?
2. What is the use of Dijkstra's algorithm?
3. What is path?
4. What is minimum cost path?
5. How to find shortest path using Dijkstra's algorithm?

Experiment 5

5. Take an example subnet of hosts and obtain a broadcast tree for the subnet.

NAME OF THE EXPERIMENT: Broadcast Tree.
OBJECTIVE: Implement broadcast tree for a given subnet of hosts
RESOURCE: Turbo C

PROGRAM LOGIC:
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.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
int p,q,u,v,n;
int min=99,mincost=0;
int t[50][2],i,j;
int parent[50],edge[50][50];
main()
{
clrscr();
printf("\n Enter the number of nodes");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("%c\t",65+i);
parent[i]=-1;
}
printf("\n");
for(i=0;i<n;i++)
{
printf("%c",65+i);
for(j=0;j<n;j++)
scanf("%d",&edge[i][j]);
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
if(edge[i][j]!=99)
```

```
if(min>edge[i][j])
{
min=edge[i][j];
u=i;
v=j;
}
p=find(u);
q=find(v);
if(p!=q)
{
t[i][0]=u;
t[i][1]=v;
mincost=mincost+edge[u][v];
sunion(p,q);
}
Else
{
t[i][0]=-1;t[i][1]=-1;
}
min=99;
}
printf("Minimum cost is %d\n Minimum spanning tree is\n" ,mincost);
for(i=0;i<n;i++)
if(t[i][0]!=-1 && t[i][1]!=-1)
{
printf("%c %c %d", 65+t[i][0],65+t[i][1],edge[t[i][0]][t[i][1]]);printf("\n");
}
getch();
}
sunion(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)
i=parent[i];
return i;
}
```

Viva questions:

1. What is spanning tree?
2. What is broad cast tree?
3. What are the advantages of broadcast tree?
4. What is flooding?

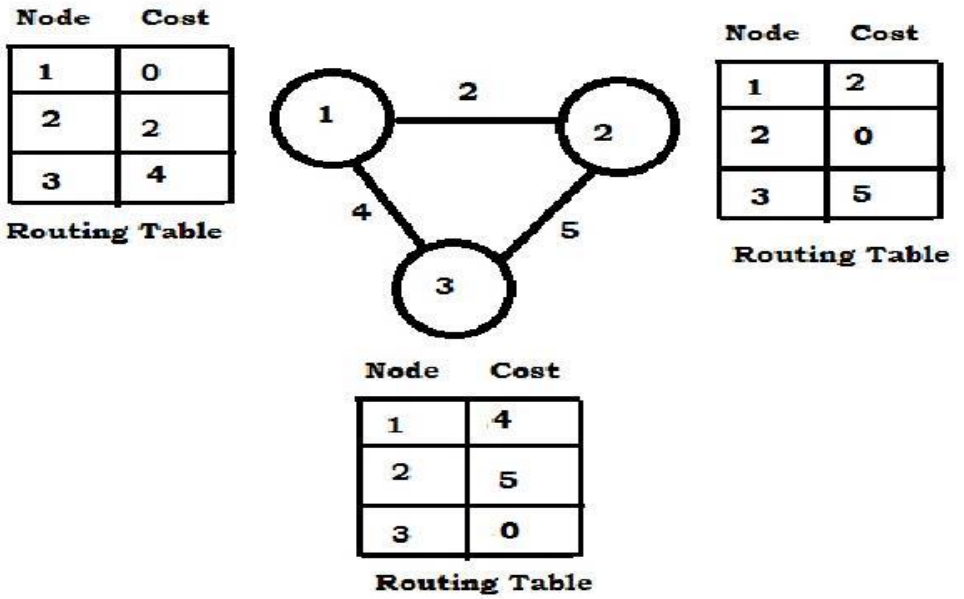
5. What is subnet?

Experiment 6

6. Implement distance vector routing algorithm for obtaining routing tables at each node.

NAME OF HE EXPERIMENT: Distance Vector routing.

OBJECTIVE: Obtain Routing table at each node using distance vector routing algorithm for a given subnet.



RESOURCE: Turbo C

PROGRAM LOGIC:
Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination. Distance vector routing algorithms are not preferable for complex networks and take longer to converge.

Viva Questions:
1. What is routing?
2. What is best algorithm among all routing algorithms?
3. What is static routing?
4. Differences between static and dynamic?
5. What is optimality principle?

Experiment 7

7. Implement data encryption and data decryption

7. a. NAME OF THE EXPERIMENT: encrypting DES.

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

RESOURCE: Turbo C

PROGRAM LOGIC:
Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

ALOGRITHM:

It is a simple type of substitution cipher, in this, each letter or word of a given text message is replaced by a letter some fixed number down the original alphabet.
We decide that fixed number, for example, if we select that number as 2 then A will be replaced by C, B will be replaced by D, and so on.
This fixed number here indicates the shift, which means the number of positions by which each letter of the text has to be moved down.

SOURCE CODE:

```
#include<stdio.h>
int main()
{
char message[100], ch;
int i, key;
printf("Enter a message to encrypt: ");
gets(message);
printf("Enter key: ");
scanf("%d", &key);
for(i = 0; message[i] != '\0'; ++i){
ch = message[i];
if(ch >= 'a' && ch <= 'z'){
```

```
ch = ch + key;
if(ch > 'z'){
ch = ch - 'z' + 'a' - 1;
}
message[i] = ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = ch + key;
if(ch > 'Z'){
ch = ch - 'Z' + 'A' - 1;
}
message[i] = ch;
}
}
printf("Encrypted message: %s", message);
return 0;

}
```

Viva Questions:

1. What is cipher text?
2. What is plain text?
3. Define public key?
4. Define encryption?

7. b. NAME OF THE EXPERIMENT: Decrypting DES.

OBJECTIVE: Write a program to break the above DES coding

RESOURCE: Turbo C

PROGRAM LOGIC:

Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

ALOGRITHM:

It is a simple type of substitution cipher, in this, each letter or word of a given text message is replaced by a letter some fixed number down the original alphabet.
We decide that fixed number, for example, if we select that number as 2 then A will be replaced by C, B will be replaced by D, and so on.
This fixed number here indicates the shift, which means the number of positions by which each letter of the text has to be moved down.

SOURCE CODE:

```
/*Write a program to break the above DES coding*/

#include<stdio.h>
int main()
{
char message[100], ch;
int i, key;
printf("Enter a message to decrypt: ");
gets(message);
printf("Enter key: ");
scanf("%d", &key);
for(i = 0; message[i] != '\0'; ++i){
ch = message[i];
if(ch >= 'a' && ch <= 'z'){
ch = ch - key;
if(ch < 'a'){
ch = ch + 'z' - 'a' + 1;
}
message[i] = ch;
}
else if(ch >= 'A' && ch <= 'Z'){
ch = ch - key;
if(ch < 'A'){
ch = ch + 'Z' - 'A' + 1;
}
message[i] = ch;
} }
printf("Decrypted message: %s", message);
return 0;
}
```

Viva Questions:

1. Define decryption
2. What is private key?
3. What is cipher feedback mode?
4. Define product cipher
5. What is DES chaining?

Experiment 8

8. Write a program for congestion control using Leaky bucket algorithm.

SOURCE CODE:

```
#include<stdio.h>
int main(){
int incoming, outgoing, buck_size, n, store = 0;
printf("Enter bucket size, outgoing rate and no of inputs: ");
scanf("%d %d %d", &buck_size, &outgoing, &n);

while (n != 0) {
printf("Enter the incoming packet size : ");
scanf("%d", &incoming);
printf("Incoming packet size %d\n", incoming);
if (incoming <= (buck_size - store)){
store += incoming;
printf("Bucket buffer size %d out of %d\n", store, buck_size);
        } else {
printf("Dropped %d no of packets\n", incoming - (buck_size - store));
printf("Bucket buffer size %d out of %d\n", store, buck_size);
store = buck_size;
        }
store = store - outgoing;
printf("After outgoing %d packets left out of %d in buffer\n", store, buck_size);
n--;
    }
}
```

OUTPUT:

Experiment 9

9. Write a program for frame sorting technique used in buffers.

SOURCE CODE:

```
#include<stdio.h>
#include<string.h>
#define FRAM_TXT_SIZ 3
#define MAX_NOF_FRAM 127
char str[FRAM_TXT_SIZ*MAX_NOF_FRAM];
struct frame // structure maintained to hold frames
{ char text[FRAM_TXT_SIZ];
int seq_no;
}fr[MAX_NOF_FRAM], shuf_ary[MAX_NOF_FRAM];
int assign_seq_no() //function which splits message
{ int k=0,i,j; //into frames and assigns sequence no
for(i=0; i < strlen(str); k++)
{ fr[k].seq_no = k;
for(j=0; j < FRAM_TXT_SIZ && str[i]!='\0'; j++)
fr[k].text[j] = str[i++];
}
printf("\nAfter assigning sequence numbers:\n");
for(i=0; i < k; i++)
printf("%d:%s ",i,fr[i].text);
return k; //k gives no of frames
}
void generate(int *random_ary, const int limit) //generate array of random nos
{ int r, i=0, j;
while(i < limit)
{ r = random() % limit;
for(j=0; j < i; j++)
if( random_ary[j] == r )
break;
if( i==j ) random_ary[i++] = r;
} }
void shuffle( const int no_frames ) // function shuffles the frames
{
int i, k=0, random_ary[no_frames];
generate(random_ary, no_frames);
for(i=0; i < no_frames; i++)
shuf_ary[i] = fr[random_ary[i]];
```

```
printf("\n\nAFTER SHUFFLING:\n");
for(i=0; i < no_frames; i++)
printf("%d:%s ",shuf_ary[i].seq_no,shuf_ary[i].text);
}
void sort(const int no_frames) // sorts the frames
{
int i,j,flag=1;
struct frame hold;
for(i=0; i < no_frames-1 && flag==1; i++) // search for frames in sequence
{
flag=0;
for(j=0; j < no_frames-1-i; j++) //(based on seq no.) and display
if(shuf_ary[j].seq_no > shuf_ary[j+1].seq_no)
{
hold = shuf_ary[j];
shuf_ary[j] = shuf_ary[j+1];
shuf_ary[j+1] = hold;
flag=1;
}
}
}
int main()
{
int no_frames,i;
printf("Enter the message: ");
gets(str);
no_frames = assign_seq_no();
shuffle(no_frames);
sort(no_frames);
printf("\n\nAFTER SORTING\n");
for(i=0;i<no_frames;i++)
printf("%s",shuf_ary[i].text);
printf("\n\n");
}
```

Experiment 10

Wireshark

What is Wireshark?

Wireshark is an open-source network protocol analysis software program started by Gerald Combs in 1998. A global organization of network specialists and software developers support Wireshark and continue to make updates for new network technologies and encryption methods.
Wireshark is absolutely safe to use. Government agencies, corporations, non-profits, and educational institutions use Wireshark for troubleshooting and teaching purposes. There isn't a better way to learn networking than to look at the traffic under the Wireshark microscope.

There are questions about the legality of Wireshark since it is a powerful packet sniffer. The Light side of the Force says that you should only use Wireshark on networks where you have permission to inspect network packets. Using Wireshark to look at packets without permission is a path to the Dark Side.

How does Wireshark work?

Wireshark is a packet sniffer and analysis tool. It captures network traffic on the local network and stores that data for offline analysis. Wireshark captures network traffic from Ethernet, Bluetooth, Wireless (IEEE.802.11), Token Ring, Frame Relay connections, and more.

Ed. Note: A "packet" is a single message from any network protocol (i.e., TCP, DNS, etc.)
Ed. Note 2: LAN traffic is in broadcast mode, meaning a single computer with Wireshark can see traffic between two other computers. If you want to see traffic to an external site, you need to capture the packets on the local computer.
Wireshark allows you to filter the log either before the capture starts or during analysis, so you can narrow down and zero into what you are looking for in the network trace. For example, you can set a filter to see TCP traffic between two IP addresses. You can set it only to show you the packets sent from one computer. The filters in Wireshark are one of the primary reasons it became the standard tool for packet analysis.

How to Download Wireshark

Downloading and installing Wireshark is easy. Step one is to check the official Wireshark Download page for the operating system you need. The basic version of Wireshark is free.

Wireshark for Windows

Wireshark comes in two flavors for Windows, 32 bit and 64 bit. Pick the correct version for your OS. The current release is 3.0.3 as of this writing. The installation is simple and shouldn't cause any issues.

Wireshark for Mac

Wireshark is available on Mac as a Homebrew install.
To install Homebrew, you need to run this command at your Terminal prompt:

/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
Once you have the Homebrew system in place, you can access several open-source projects for your Mac. To install Wireshark run this command from the Terminal:

brew install wireshark
Homebrew will download and install Wireshark and any dependencies so it will run correctly.

Wireshark for Linux

Installing Wireshark on Linux can be a little different depending on the Linux distribution. If you aren't running one of the following distros, please double-check the commands.

Ubuntu

From a terminal prompt, run these commands:

1. sudo apt-get install wireshark
2. sudo dpkg-reconfigure wireshark-common
3. sudo adduser $USER wireshark

Those commands download the package, update the package, and add user privileges to run Wireshark.

Red Hat Fedora

From a terminal prompt, run these commands:

1. sudo dnf install wireshark-qt
2. sudo usermod -a -G wireshark username

The first command installs the GUI and CLI version of Wireshark, and the second adds permissions to use Wireshark.

Kali Linux

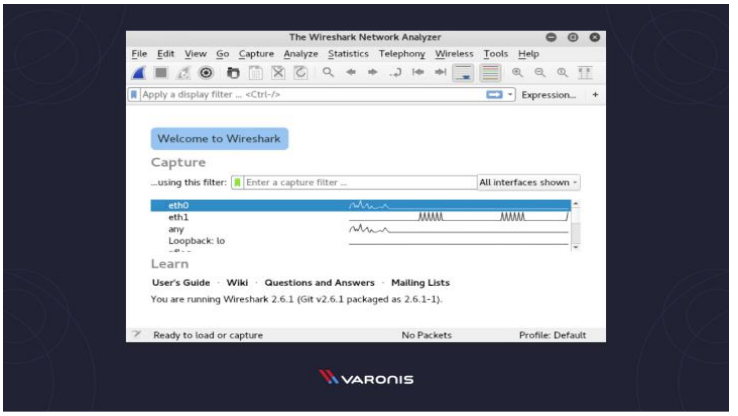
Wireshark is probably already installed! It's part of the basic package. Check your menu to verify. It's under the menu option "Sniffing & Spoofing."

Data Packets on Wireshark

Now that we have Wireshark installed let's go over how to enable the Wireshark packet sniffer and then analyze the network traffic.

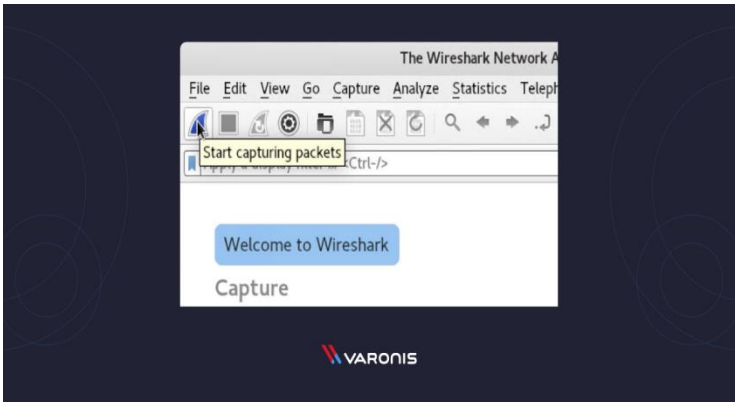
i) Capturing Data Packets on Wireshark

When you open Wireshark, you see a screen that shows you a list of all of the network connections you can monitor. You also have a capture filter field, so you only capture the network traffic you want to see.

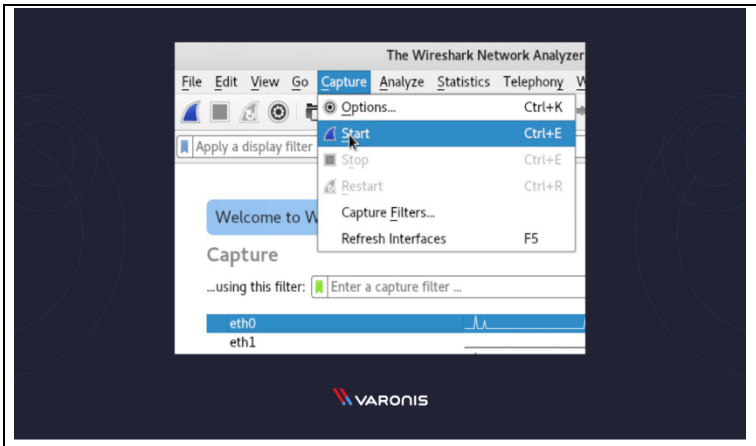


You can select one or more of the network interfaces using “shift left-click.” Once you have the network interface selected, you can start the capture, and there are several ways to do that.

Click the first button on the toolbar, titled “Start Capturing Packets.”

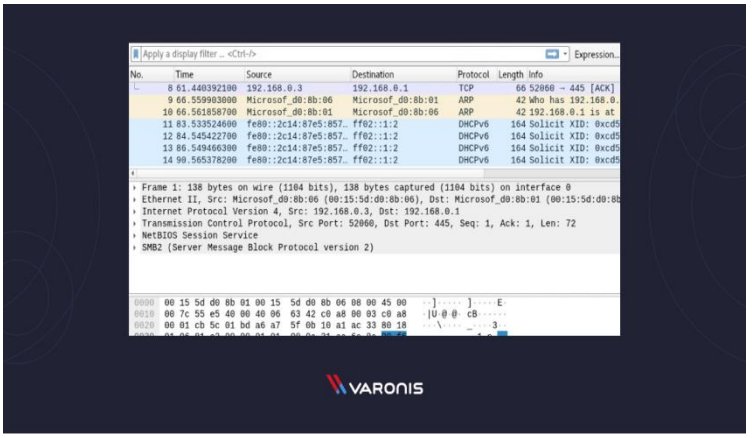


You can select the menu item Capture -> Start.



Or you could use the keystroke Control – E.

During the capture, Wireshark will show you the packets that it captures in real-time.



Once you have captured all the packets you need, you use the same buttons or menu options to stop the capture.

Best practice says that you should stop Wireshark packet capture before you do analysis.

iv) Analyzing Data Packets on Wireshark

Wireshark shows you three different panes for inspecting packet data. The Packet List, the top pane, is a list of all the packets in the capture. When you click on a packet, the other two panes change to show you the details about the selected packet. You can also tell if the packet is part of a conversation. Here are some details about each column in the top pane:

- **No.**: This is the number order of the packet that got captured. The bracket indicates that this packet is part of a conversation.
- **Time**: This column shows you how long after you started the capture that this packet got captured. You can change this value in the Settings menu if you need something different displayed.
- **Source**: This is the address of the system that sent the packet.
- **Destination**: This is the address of the destination of that packet.
- **Protocol**: This is the type of packet, for example, TCP, DNS, DHCPv6, or ARP.
- **Length**: This column shows you the length of the packet in bytes.
- **Info**: This column shows you more information about the packet contents, and will vary depending on what kind of packet it is.
-

Packet Details, the middle pane, shows you as much readable information about the packet as possible, depending on what kind of packet it is. You can right-click and create filters based on the highlighted text in this field.

The bottom pane, Packet Bytes, displays the packet exactly as it got captured in hexadecimal.

When you are looking at a packet that is part of a conversation, you can right-click the packet and select Follow to see only the packets that are part of that conversation.

Wireshark Filters

One of the best features of Wireshark is the Wireshark Capture Filters and Wireshark Display Filters. Filters allow you to view the capture the way you need to see it so you can troubleshoot the issues at hand. Here are several filters to get you started.

Wireshark Capture Filters

Capture filters limit the captured packets by the filter. Meaning if the packets don't match the filter, Wireshark won't save them. Here are some examples of capture filters:
host IP-*address*: this filter limits the capture to traffic to and from the IP address
net 192.168.0.0/24: this filter captures all traffic on the subnet.

dst host IP-*address*: capture packets sent to the specified host.
port 53: capture traffic on port 53 only.

port not 53 and not arp: capture all traffic except DNS and ARP traffic

Wireshark Display Filters

Wireshark Display Filters change the view of the capture during analysis. After you have stopped the packet capture, you use display filters to narrow down the packets in the Packet List so you can troubleshoot your issue.
The most useful (in my experience) display filter is:

ip.src==*IP-address* and ip.dst==*IP-address*
This filter shows you packets from one computer (ip.src) to another (ip.dst). You can also use ip.addr to show you packets to and from that IP. Here are some others:

tcp.port eq 25: This filter will show you all traffic on port 25, which is usually SMTP traffic.

icmp: This filter will show you only ICMP traffic in the capture, most likely they are pings.

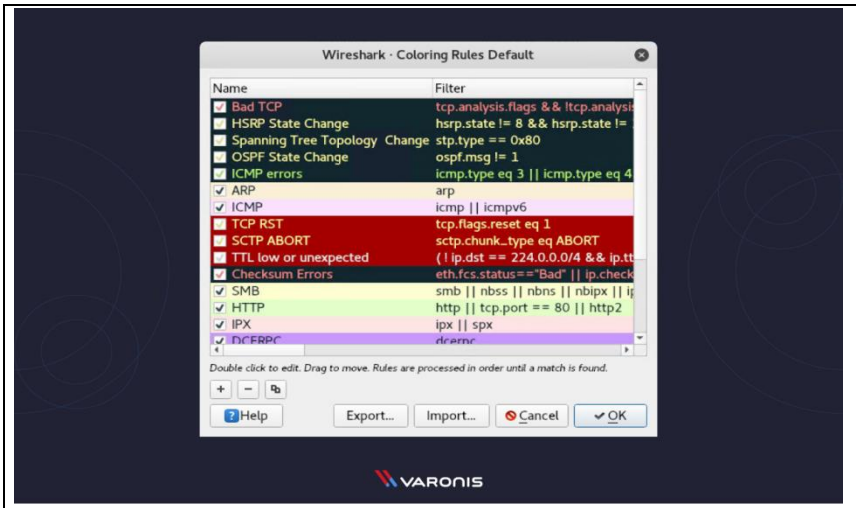
ip.addr != *IP_address*: This filter shows you all traffic except the traffic to or from the specified computer.
Analysts even build filters to detect specific attacks, like this filter to detect the Sasser worm:
ls_ads.opnum==0x09

Additional Wireshark Features

Beyond the capture and filtering, there are several other features in Wireshark that can make your life better.

Wireshark Colorization Options

You can setup Wireshark so it colors your packets in the Packet List according to the display filter, which allows you to emphasize the packets you want to highlight. Check out some examples here.



Wireshark Promiscuous Mode

By default, Wireshark only captures packets going to and from the computer where it runs. By checking the box to run Wireshark in Promiscuous Mode in the Capture Settings, you can capture most of the traffic on the LAN.

Wireshark Command Line

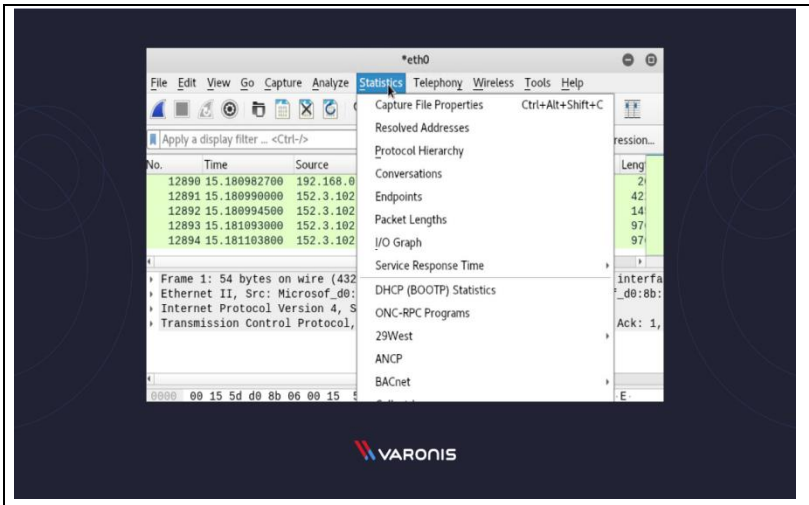
Wireshark does provide a Command Line Interface (CLI) if you operate a system without a GUI. Best practice would be to use the CLI to capture and save a log so you can review the log with the GUI.

Wireshark Commands

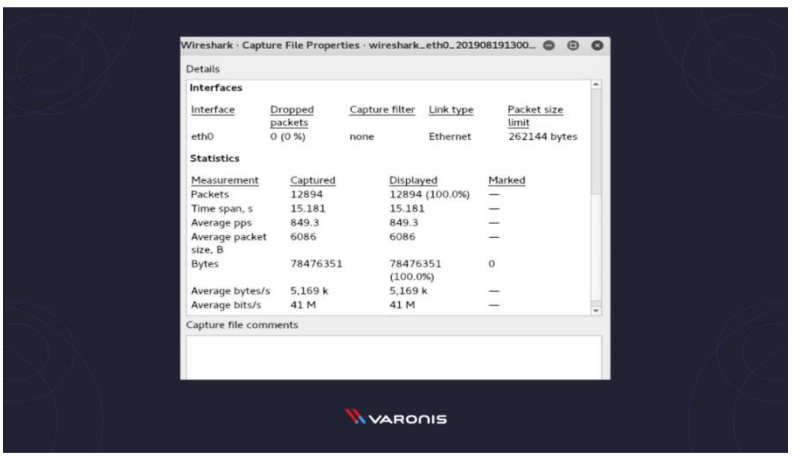
- wireshark : run Wireshark in GUI mode
- wireshark –h : show available command line parameters for Wireshark
- wireshark –a duration:300 –i eth1 –w wireshark. : capture traffic on the Ethernet interface 1 for 5 minutes. –a means automatically stop the capture, -i specifics which interface to capture

Metrics and Statistics

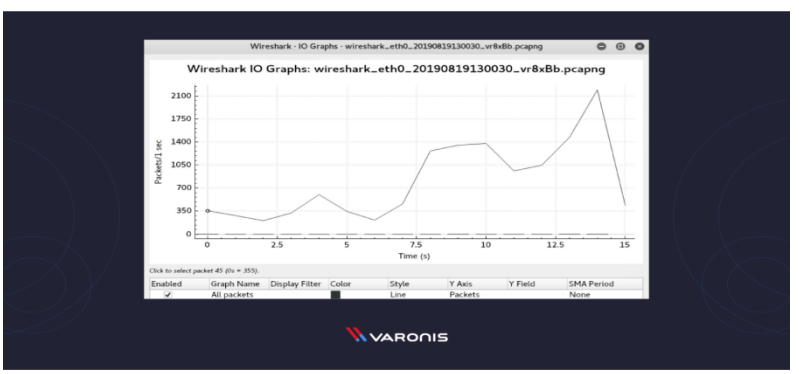
Under the Statistics menu item, you will find a plethora of options to show details about your capture.



Capture File Properties:



Wireshark I/O Graph:



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

1. Use Wireshark to perform a packet capture of network traffic

In order to investigate your issues further, we would like to run an analysis of the traffic being sent between you and Salesforce. To do this, we will use the Wireshark application. Wireshark is a tool that allows packet traces to be monitored, captured and analysed. Please follow the steps below in order to obtain a capture of your network traffic using Wireshark.

Note: *You will require some administrator rights on your machine in order to complete these tests. If you are unsure, please contact your IT administrator.*

Installing the Wireshark package

Visit the Wireshark download site, and download the appropriate Wireshark package or installer for the operating system running on the system which is to be used for packet capture.

When installing, ensure all components are selected for installation, including the optional "Winpcap" application.

Once complete, start Wireshark via shortcut or start menu.

Capturing your traffic with Wireshark

After starting Wireshark, do the following:

1. Select **Capture | Interfaces**
2. Select the interface on which packets need to be captured. This will usually be the interface where the Packet/s column is constantly changing, which would indicate the presence of live traffic). If you have multiple network interface cards (i.e. LAN card and Wi-Fi adapter) you may need to check with your IT administrator to determine the right interface.
3. Click the **Start** button to start the capture.

4. Recreate the problem. The capture dialog should show the number of packets increasing. Try to avoid running any other internet applications while capturing, closing other browsers, Instant messengers etc.
5. Once the problem which is to be analyzed has been reproduced, click on **Stop**. It may take a few seconds for Wireshark to display the packets captured.
6. Save the packet trace in the default format. Click on the **File** menu option and select **Save As**. By default Wireshark will save the packet trace in libpcap format. This is a filename with a.pcap extension.

Returning the information to Salesforce support

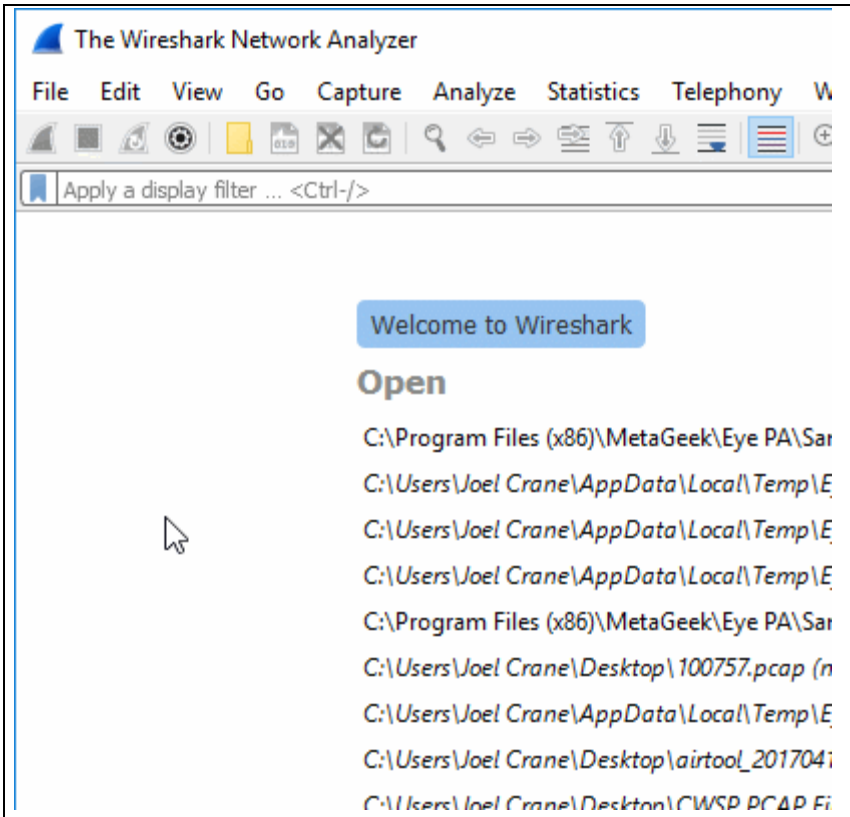
Forward the resulting .pcap file to your support representative, either by email, or attaching it to your open case. Please also include the following information:

- Your external IP address (get this from http://www.whatismyip.com/)
- The internal IP address of the local machine where traffic is being captured
- A click path of the steps you took to reproduce, including links to each page/record accessed

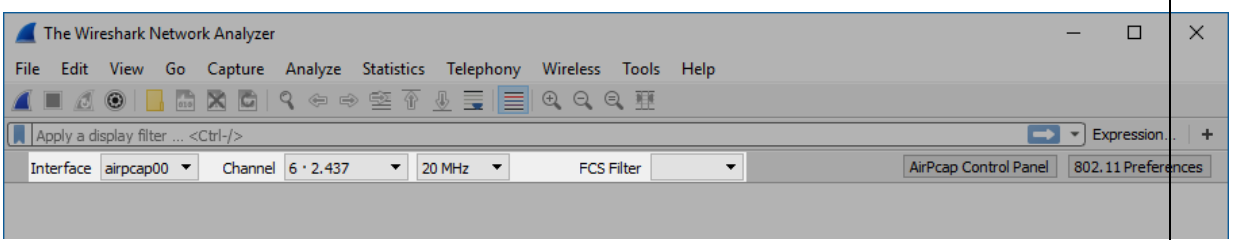
In some cases, you may want to perform packet captures with Wireshark. One case might be when you want to perform a packet capture on channel 12 or 13.

Set up the Packet Capture

1. Click **View** > **Wireless Toolbar.** The Wireless Toolbar will appear just below the Main toolbar.



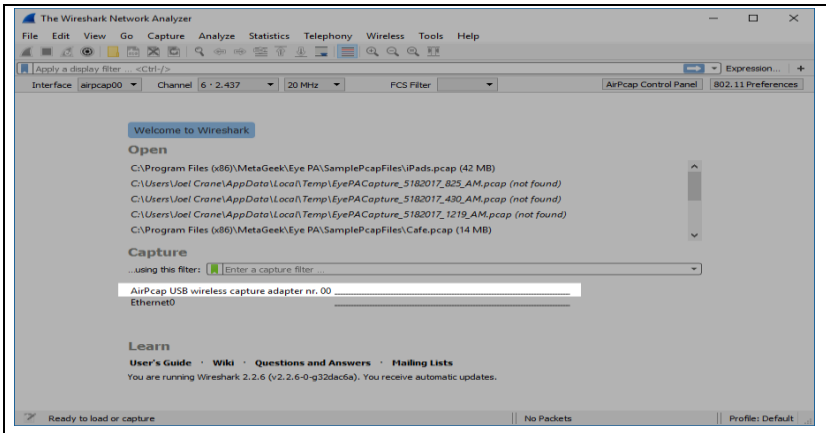
2. Use the **Wireless Toolbar** to configure the desired channel and channel width.



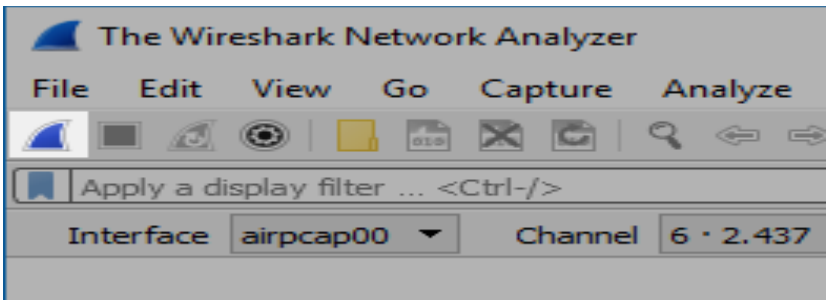
3. Under **Capture**, click on **AirPcap USB wireless capture adapter** to select the capture interface.

*Note: If the AirPcap isn't listed, press **F5** to refresh the list of available packet capture interfaces.*

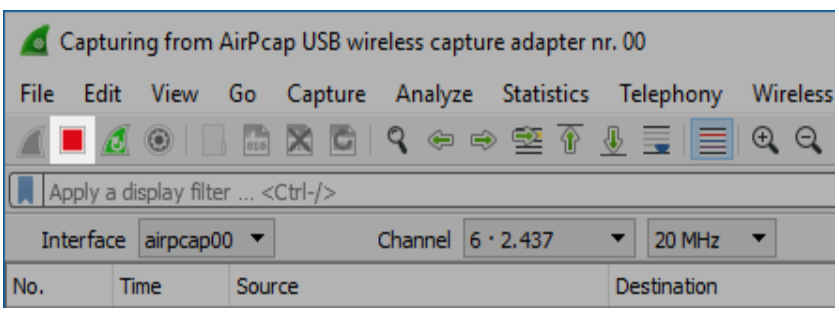
Note: The AirPcap has been discontinued by RiverBed and is 802.11n only.



4. Click the **Start Capture** button to begin the capture.

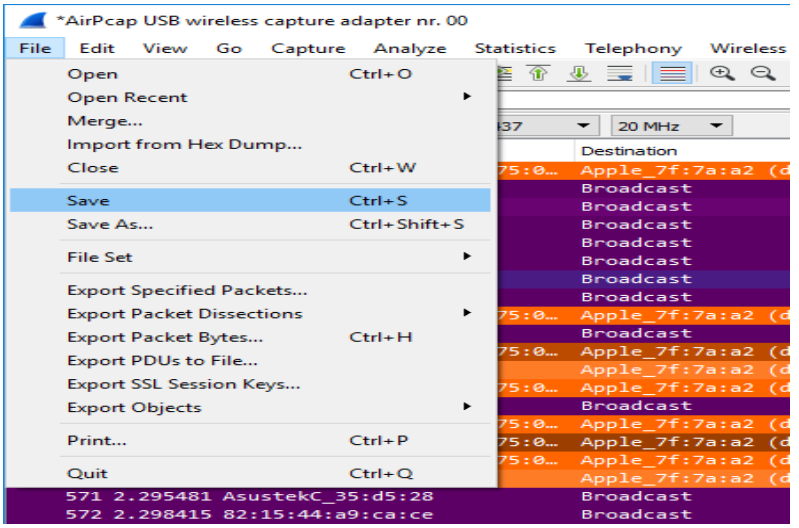


5. When you are finished capturing, click the **Stop** button.



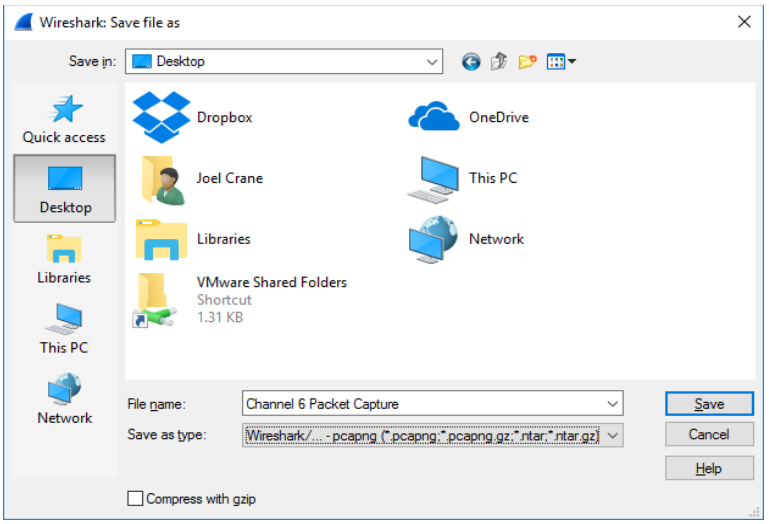
Saving the Capture

1. To save the capture, click **File** > **Save**.



2. Name the file, and click **Save**.

Note: ***.Pcap*** *and* ***.Pcap-ng*** *are good filetypes to use for the capture if you plan to use Eye P.A. to open the capture.*

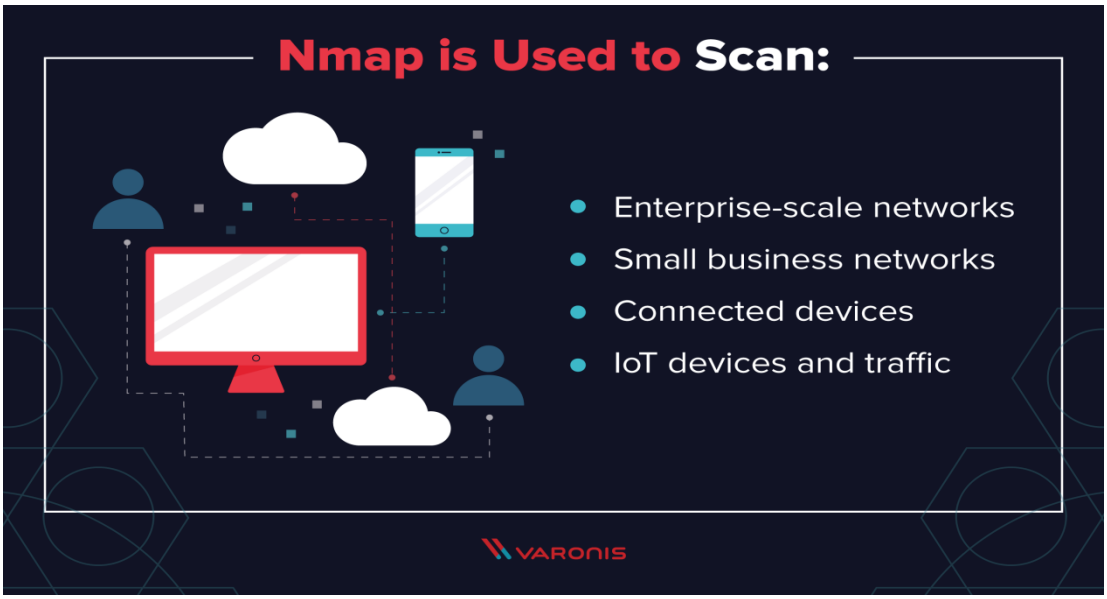


3. Eye P.A. can now open the capture file.

Experiment 11

11. How to run Nmapscan

What is Nmap?

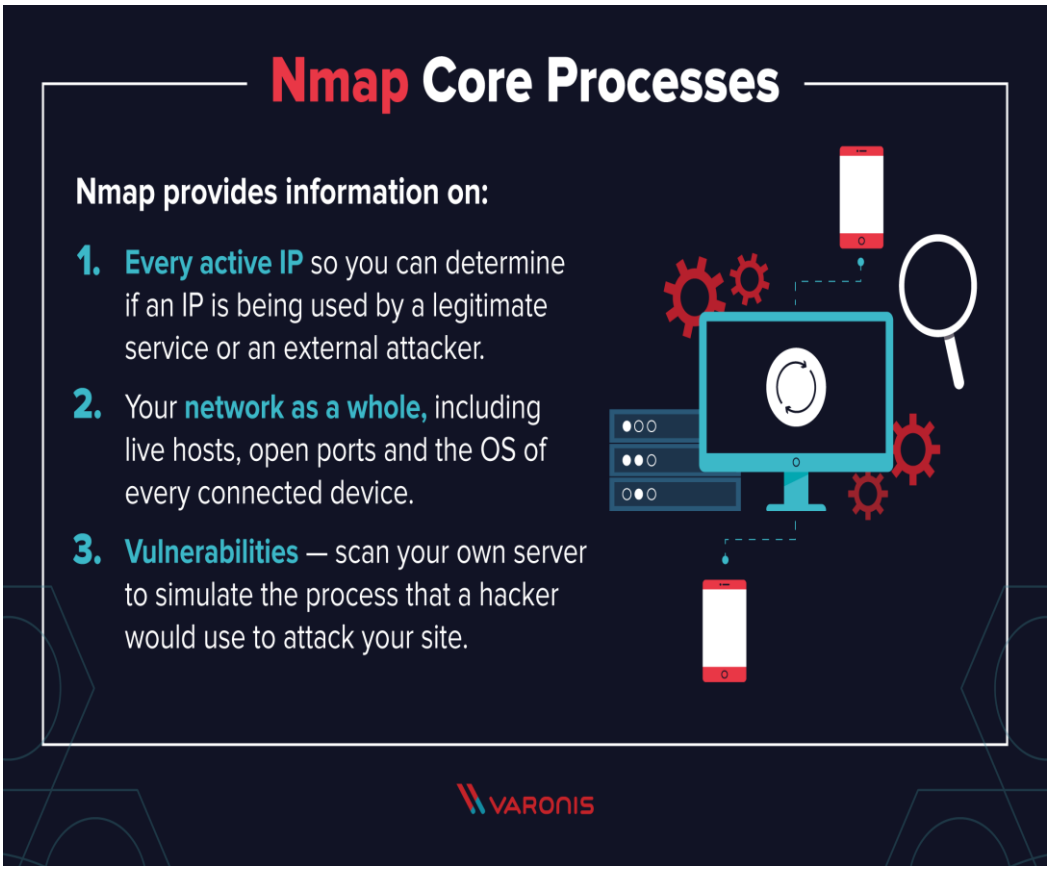


At its core, Nmap is a network scanning tool that uses IP packets to identify all the devices connected to a network and to provide information on the services and operating systems they are running.

The program is most commonly used via a command-line interface (though GUI front-ends are also available) and is available for many different operating systems such as Linux, Free BSD, and Gentoo. Its popularity has also been bolstered by an active and enthusiastic user support community.

Nmap was developed for enterprise-scale networks and can scan through thousands of connected devices. However, in recent years Nmap is being increasingly used by smaller companies. The rise of the IoT, in particular, now means that the networks used by these companies have become more complex and therefore harder to secure.
This means that Nmap is now used in many website monitoring tools to audit the traffic between web servers and IoT devices. The recent emergence of IoT botnets, like Mirai, has also stimulated interest in Nmap, not least because of its ability to interrogate devices connected via the UPnP protocol and to highlight any devices that may be malicious.

What Does Nmap Do?



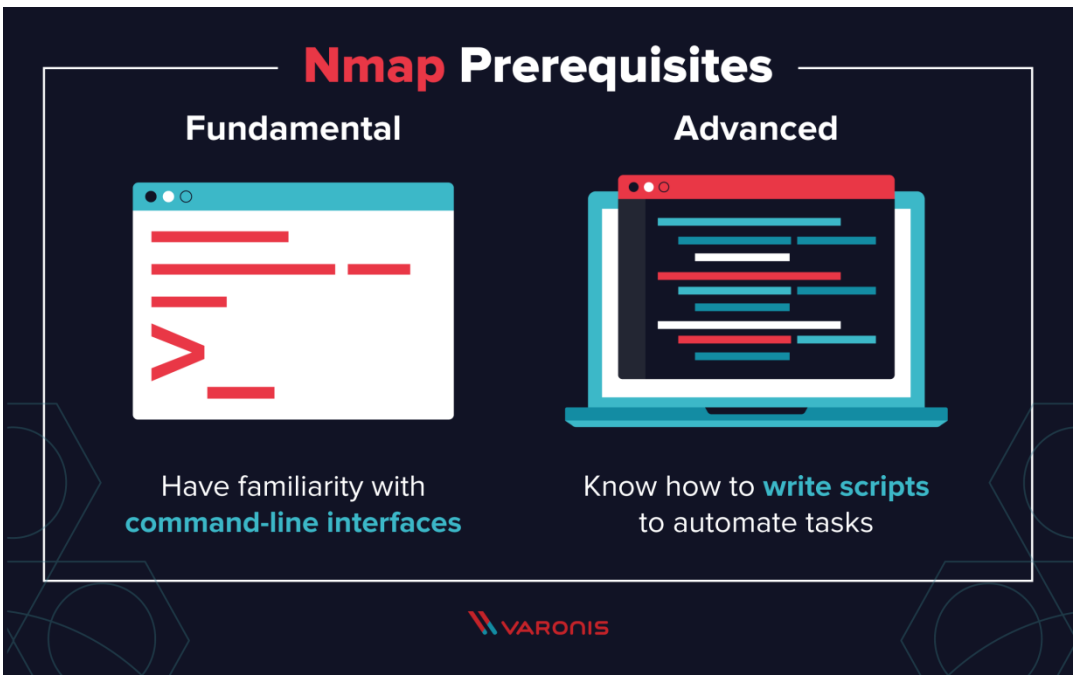
At a practical level, Nmap is used to provide detailed, real-time information on your networks, and on the devices connected to them.

The primary uses of Nmap can be broken into three core processes. First, the program gives you detailed information on every IP active on your networks, and each IP can then be scanned. This allows administrators to check whether an IP is being used by a legitimate service, or by an external attacker.

Secondly, Nmap provides information on your network as a whole. It can be used to provide a list of live hosts and open ports, as well as identifying the OS of every connected device. This makes it a valuable tool in ongoing system monitoring, as well as

a critical part of pentesting. Nmap can be used alongside the Metasploit framework, for instance, to probe and then repair network vulnerabilities.
Thirdly, Nmap has also become a valuable tool for users looking to protect personal and business websites. Using Nmap to scan your own web server, particularly if you are hosting your website from home, is essentially simulating the process that a hacker would use to attack your site. “Attacking” your own site in this way is a powerful way of identifying security vulnerabilities.

How To Use Nmap



Nmap is straightforward to use, and most of the tools it provides are familiar to system admins from other programs. The advantage of Nmap is that it brings a wide range of these tools into one program, rather than forcing you to skip between separate and discrete network monitoring tools.

In order to use Nmap, you need to be familiar with command-line interfaces. Most advanced users are able to write scripts to automate common tasks, but this is not necessary for basic network monitoring.

How To Install Nmap

The process for installing Nmap is easy but varies according to your operating system. The Windows, Mac, and Linux versions of the program can be downloaded here.

- For Windows, Nmap comes with a custom installer (namp<version>setup.exe). Download and run this installer, and it automatically configures Nmap on your system.
- On Mac, Nmap also comes with a dedicated installer. Run the Nmap-<version>mpkg file to start this installer. On some recent versions of macOS, you might see a warning that Nmap is an "unidentified developer", but you can ignore this warning.
- Linux users can either compile Nmap from source or use their chosen package manager. To use apt, for instance, you can run Nmap –version to check if Nmap is installed, and sudo apt-get install Nmap to install it.

Nmap Tutorial and Examples

Once you've installed Nmap, the best way of learning how to use it is to perform some basic network scans.

How To Run a Ping Scan

One of the most basic functions of Nmap is to identify active hosts on your network. Nmap does this by using a ping scan. This identifies all of the IP addresses that are currently online without sending any packers to these hosts.

To run a ping scan, run the following command:

1. # nmap -sp 192.100.1.1/24

This command then returns a list of hosts on your network and the total number of assigned IP addresses. If you spot any hosts or IP addresses on this list that you cannot account for, you can then run further commands (see below) to investigate them further.

How To Run A Host Scan

A more powerful way to scan your networks is to use Nmap to perform a host scan. Unlike a ping scan, a host scan actively sends ARP request packets to all the hosts connected to your network. Each host then responds to this packet with another ARP packet containing its status and MAC address.

To run a host scan, use the following command:

1. # nmap -sp <target IP range>

This returns information on every host, their latency, their MAC address, and also any description associated with this address. This can be a powerful way of spotting suspicious hosts connected to your network.

If you see anything unusual in this list, you can then run a DNS query on a specific host, by using:

1. # namp -sL <IP address>

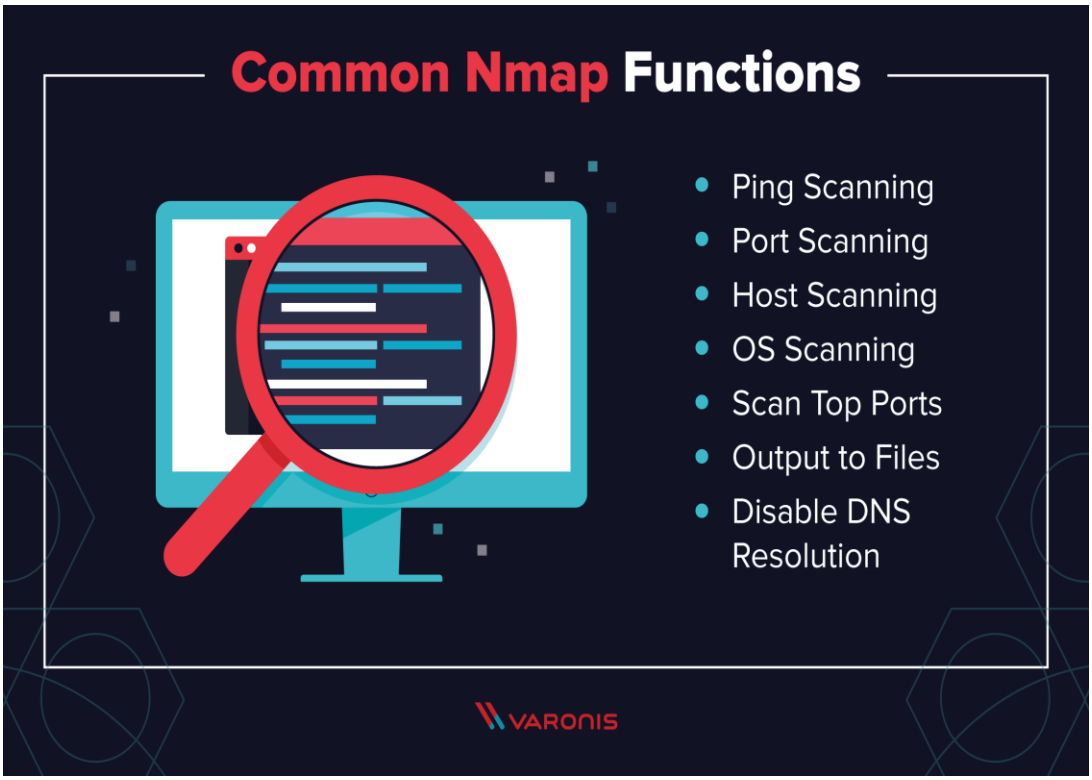
This returns a list of names associated with the scanned IP. This description provides information on what the IP is actually for.

How To Use Nmap in Kali Linux

Using Nmap in Kali Linux can be done in an identical way to running the program on any other flavor of Linux.

That said, there are advantages to using Kali when running Nmap scans. Most modern distros of Kali now come with a fully-features Nmap suite, which includes an advanced GUI and results viewer (Zenmap), a flexible data transfer, redirection, and debugging tool (Ncat), a utility for comparing scan results (Ndiff), and a packet generation and response analysis tool (Nping).

Nmap Commands



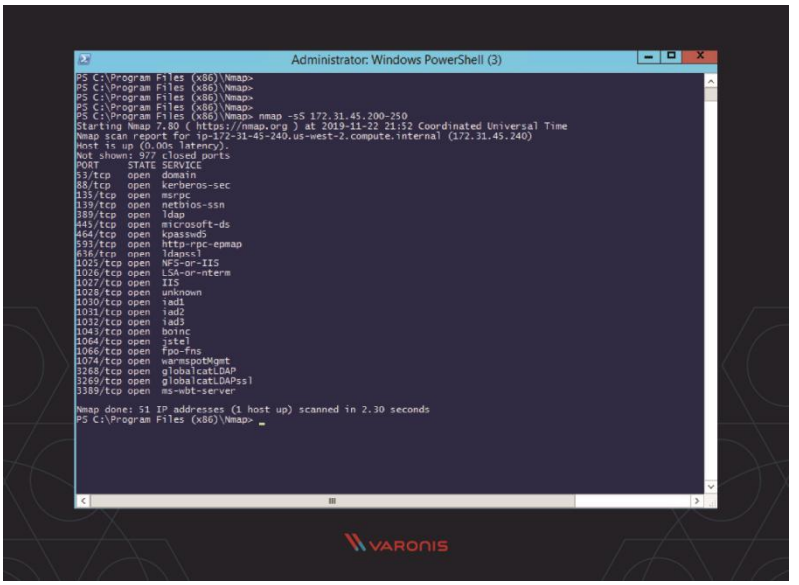
Most of the common functions of Nmap can be executed using a single command, and the program also uses a number of 'shortcut' commands that can be used to automate common tasks.

Here is a quick run-down:

1. Ping Scanning

As mentioned above, a ping scan returns information on every active IP on your network. You can execute a ping scan using this command:

2. Port Scanning



There are several ways to execute port scanning using Nmap. The most commonly used are these:

1. # sS TCP SYN scan
2.
3. # sT TCP connect scan
4.
5. # sU UDP scans
6.
7. # sY SCTP INIT scan
8.
9. # sN TCP NULL

The major differences between these types of scans are whether they cover TCP or UDP ports and whether they execute a TCP connection. Here are the basic differences:

- The most basic of these scans is the sS TCP SYN scan, and this gives most users all the information they need. It scans thousands of ports per second, and because it doesn't complete a TCP connection it does not arouse suspicion.
- The main alternative to this type of scan is the TCP Connect scan, which actively queries each host, and requests a response. This type of scan takes longer than a SYN scan, but can return more reliable information.

- The UDP scan works in a similar way to the TCP connect scan but uses UDP packets to scan DNS, SNMP, and DHCP ports. These are the ports most frequently targeted by hackers, and so this type of scan is a useful tool for checking for vulnerabilities.
- The SCTP INIT scan covers a different set of services: SS7 and SIGTRAN. This type of scan can also be used to avoid suspicion when scanning an external network because it doesn't complete the full SCTP process.
- The TOP NULL scan is also a very crafty scanning technique. It uses a loophole in the TCP system that can reveal the status of ports without directly querying them, which means that you can see their status even where they are protected by a firewall.

3. Host Scanning

Host scanning returns more detailed information on a particular host or a range of IP addresses. As mentioned above, you can perform a host scan using the following command:

1. # nmap -sp <target IP range>

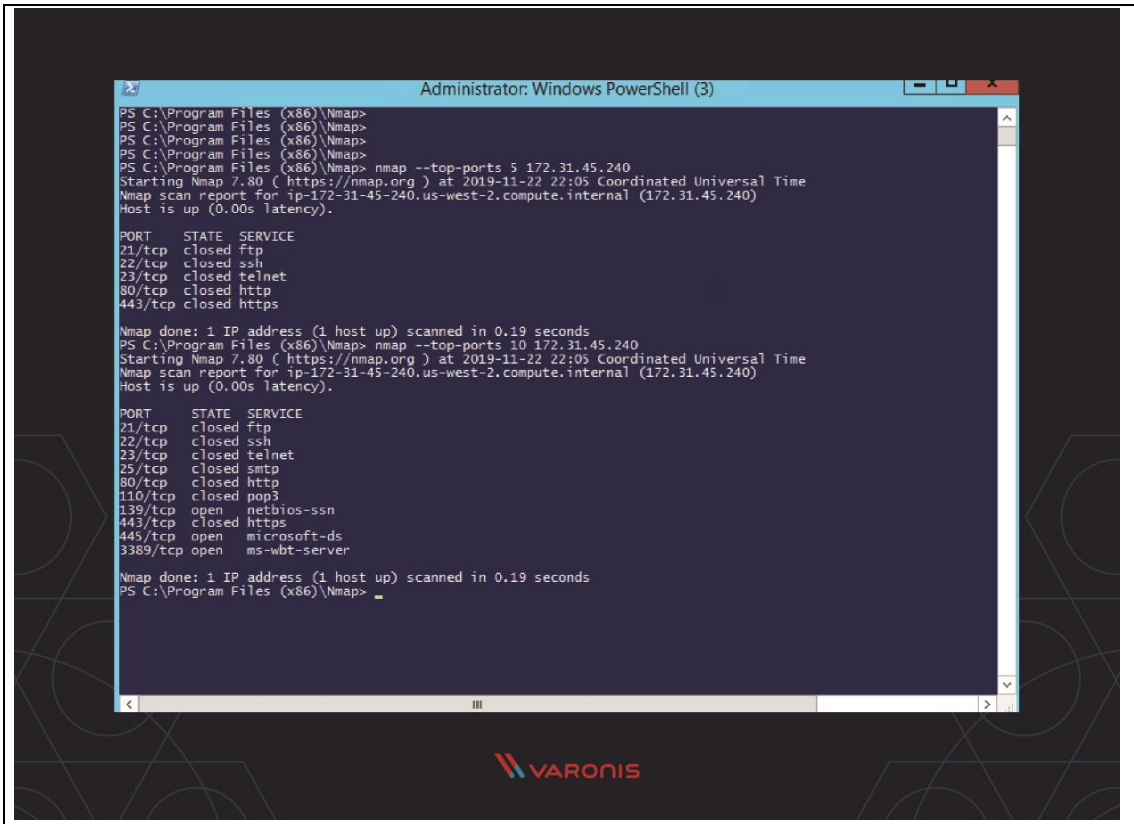
4. OS Scanning

OS scanning is one of the most powerful features of Nmap. When using this type of scan, Nmap sends TCP and UDP packets to a particular port, and then analyze its response. It compares this response to a database of 2600 operating systems, and return information on the OS (and version) of a host.

To run an OS scan, use the following command:

1. nmap -O <target IP>

5. Scan The Most Popular Ports



If you are running Nmap on a home server, this command is very useful. It automatically scans a number of the most 'popular' ports for a host. You can run this command using:

1. nmtop-ports 20 192.168.1.106

Replace the "20" with the number of ports to scan, and Nmap quickly scans that many ports. It returns a concise output that details the status of the most common ports, and this lets you quickly see whether you have any unnecessarily open ports.

6. Output to a File

If you want to output the results of your Nmap scans to a file, you can add an extension to your commands to do that. Simply add:

1. -oN output.txt

To your command to output the results to a text file, or:

1. -oX output.xml

To output to an XML.

7. Disable DNS Name Resolution

Finally, you can speed up your Nmap scans by using the -n parameter to disable reverse DNS resolution. This can be extremely useful if you want to scan a large network. For example, to turn off DNS resolution for the basic ping scan mentioned above, add -n:

1. # nmap -sp -n 192.100.1.1/24

Nmap FAQ

The commands above cover most of the basic functionality of Nmap. You might still have some questions though, so let's run through the most common ones.

Q: What Are Some Nmap Alternatives?

There are some alternatives to Nmap, but most of them are focused on providing specific, niche functionality that the average system administrator does need frequently. MASSCAN, for instance, is much faster than Nmap but provides less detail. Umit, by contrast, allows you to run several scans at once.

In reality, however, Nmap provides all the functionality and speed that the average user requires, especially when used alongside other similarly popular tools like NetCat (which can be used to manage and control network traffic) and ZenMap (which provides a GUI for Nmap)

Q: How Does Nmap Work?

Nmap builds on previous network auditing tools to provide quick, detailed scans of network traffic. It works by using IP packets to identify the hosts and IPs active on a

network and then analyze these packets to provide information on each host and IP, as well as the operating systems they are running.

Q: Is Nmap Legal?

Yes. If used properly, Nmap helps protect your network from hackers, because it allows you to quickly spot any security vulnerabilities in your systems.

Whether port scanning on external servers is legal is another issue. The legislation in this area is complex and varies by territory. Using Nmap to scan external ports can lead to you being banned by your ISP, so make sure you research the legal implications of using the program before you start using it more widely.

Experiment 12

12. Operating System Detection using Nmap

OS Detection

One of Nmap's best-known features is remote OS detection using TCP/IP stack fingerprinting. Nmap sends a series of TCP and UDP packets to the remote host and examines practically every bit in the responses. After performing dozens of tests such as TCP ISN sampling, TCP options support and ordering, IP ID sampling, and the initial window size check, Nmap compares the results to its nmap-os-db database of more than 2,600 known OS fingerprints and prints out the OS details if there is a match. Each fingerprint includes a freeform textual description of the OS, and a classification which provides the vendor name (e.g. Sun), underlying OS (e.g. Solaris), OS generation (e.g. 10), and device type (general purpose, router, switch, game console, etc). Most fingerprints also have a Common Platform Enumeration (CPE) representation, like cpe:/o:linux:linux_kernel:2.6.

If Nmap is unable to guess the OS of a machine, and conditions are good (e.g. at least one open port and one closed port were found), Nmap will provide a URL you can use to submit the fingerprint if you know (for sure) the OS running on the machine. By doing this you contribute to the pool of operating systems known to Nmap and thus it will be more accurate for everyone.

OS detection enables some other tests which make use of information that is gathered during the process anyway. One of these is TCP Sequence Predictability Classification. This measures approximately how hard it is to establish a forged TCP connection against the remote host. It is useful for exploiting source-IP based trust relationships (rlogin, firewall filters, etc) or for hiding the source of an attack. This sort of spoofing is rarely performed any more, but many machines are still vulnerable to it. The actual difficulty number is based on statistical sampling and may fluctuate. It is generally better to use the English classification such as “worthy challenge” or “trivial joke”. This is only reported in normal output in verbose (-v) mode. When verbose mode is enabled along with -O, IP ID sequence generation is also reported. Most machines are in the “incremental” class, which means that they increment the ID field in the IP header for each packet they send. This makes them vulnerable to several advanced information gathering and spoofing attacks.

Another bit of extra information enabled by OS detection is a guess at a target's uptime. This uses the TCP timestamp option (RFC 1323) to guess when a machine was last rebooted. The guess can be inaccurate due to the timestamp counter not being initialized to zero or the counter overflowing and wrapping around, so it is printed only in verbose mode.

OS detection is enabled and controlled with the following options:

-O (Enable OS detection)

Enables OS detection, as discussed above. Alternatively, you can use -A to enable OS detection along with other things.

--osscan-limit (Limit OS detection to promising targets)

OS detection is far more effective if at least one open and one closed TCP port are found. Set this option and Nmap will not even try OS detection against hosts that do not meet this criteria. This can save substantial time, particularly on -Pn scans against many hosts. It only matters when OS detection is requested with -O or -A.

--osscan-guess; --fuzzy (Guess OS detection results)

When Nmap is unable to detect a perfect OS match, it sometimes offers up near-matches as possibilities. The match has to be very close for Nmap to do this by default. Either of these (equivalent) options make Nmap guess more aggressively. Nmap will still tell you when an imperfect match is printed and display its confidence level (percentage) for each guess.

--max-os-tries (Set the maximum number of OS detection tries against a target)

When Nmap performs OS detection against a target and fails to find a perfect match, it usually repeats the attempt. By default, Nmap tries five times if conditions are favorable for OS fingerprint submission, and twice when conditions aren't so good. Specifying a lower --max-os-tries value (such as 1) speeds Nmap up, though you miss out on retries which could potentially identify the OS. Alternatively, a high value may be set to allow even more retries when conditions are favorable. This is rarely done, except to generate better fingerprints for submission and integration into the Nmap OS database.

Experiment 13

13. Do the following using NS2Simulator

i. NS2Simulator-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.

vi. Simulate to Plot Congestion for Different Source/Destination

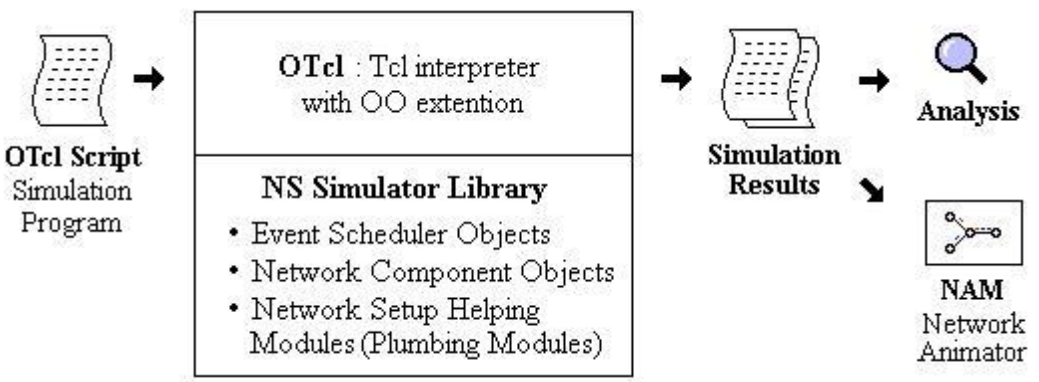
vii. Simulate to Determine the Performance with respect to Transmission of Packets

SIMULATION USING NS-2

i)Introduction to NS-2:

NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks.
Widely known as NS2, is simply an event driven simulation tool.
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.

Basic Architecture of NS2



TCL – Tool Command Language

Tcl is a very simple programming language. If you have programmed before, you can learn enough to write interesting Tcl programs within a few hours. This page provides a quick overview of the main features of Tcl. After reading this you'll probably be able to start writing simple Tcl scripts on your own; however, we recommend that you consult one of the many available Tcl books for more complete information.

Basic syntax

Tcl scripts are made up of *commands* separated by newlines or semicolons.
Commands all have the same basic form illustrated by the following example:

```
expr 20 + 10
```

This command computes the sum of 20 and 10 and returns the result, 30. You can try out this example and all the others in this page by typing them to a Tcl application such as tclsh; after a command completes, tclsh prints its result.

Each Tcl command consists of one or more *words* separated by spaces. In this example there are four words: expr, 20, +, and 10. The first word is the name of a command and the other words are *arguments* to that command. All Tcl commands consist of words, but different commands treat their arguments differently. The expr command treats all of its arguments together as an arithmetic expression, computes the result of that expression, and returns the result as a string. In the expr command the division into words isn't significant: you could just as easily have invoked the same command as

```
expr 20+10
```

However, for most commands the word structure is important, with each word used for a distinct purpose.

All Tcl commands return results. If a command has no meaningful result then it returns an empty string as its result.

Variables

Tcl allows you to store values in variables and use the values later in commands. The set command is used to write and read variables. For example, the following command modifies the variable x to hold the value 32:

```
set x 32
```

The command returns the new value of the variable. You can read the value of a variable by invoking set with only a single argument:

```
set x
```

You don't need to declare variables in Tcl: a variable is created automatically the first time it is set. Tcl variables don't have types: any variable can hold any value.

To use the value of a variable in a command, use *variable substitution* as in the following example:

```
expr $x*3
```

When a $ appears in a command, Tcl treats the letters and digits following it as a variable name, and substitutes the value of the variable in place of the name. In this example, the actual argument received by the expr command will be 32*3 (assuming that variable x was set as in the previous example). You can use variable substitution in any word of any command, or even multiple times within a word:

```
set cmd expr
set x 11
$cmd $x*$x
```

Command substitution

You can also use the result of one command in an argument to another command.
This is called *command substitution*:

```
set a 44
set b [expr $a*4]
```

When a [appears in a command, Tcl treats everything between it and the matching] as a nested Tcl command. Tcl evaluates the nested command and substitutes its result into the enclosing command in place of the bracketed text. In the example above the second argument of the second set command will be 176.

Quotes and braces

Double-quotes allow you to specify words that contain spaces. For example, consider the following script:

```
set x 24
set y 18
set z "$x + $y is [expr $x + $y]"
```

After these three commands are evaluated variable z will have the value 24 + 18 is 42. Everything between the quotes is passed to the set command as a single word. Note that (a) command and variable substitutions are performed on the text between the quotes, and (b) the quotes themselves are not passed to the command. If the quotes were not present, the set command would have received 6 arguments, which would have caused an error.
Curly braces provide another way of grouping information into words. They are different from quotes in that no substitutions are performed on the text between the curly braces:

```
set z {$x + $y is [expr $x + $y]}
```

This command sets variable z to the value "$x + $y is [expr $x + $y]".

Control structures

Tcl provides a complete set of control structures including commands for conditional execution, looping, and procedures. Tcl control structures are just commands that take Tcl scripts as arguments. The example below creates a Tcl procedure called power, which raises a base to an integer power:

```
proc power {base p} {
set result 1
```

```
while {$p > 0} {
set result [expr $result * $base]
set p [expr $p - 1]
}
return $result
}
```

This script consists of a single command, proc. The proc command takes three arguments: the name of a procedure, a list of argument names, and the body of the procedure, which is a Tcl script. Note that everything between the curly brace at the end of the first line and the curly brace on the last line is passed verbatim to proc as a single argument. The proc command creates a new Tcl command named power that takes two arguments. You can then invoke power with commands like the following:

```
power 2 6
power 1.15 5
```

When power is invoked, the procedure body is evaluated. While the body is executing it can access its arguments as variables: base will hold the first argument and p will hold the second.

The body of the power procedure contains three Tcl commands: set, while, and return. The while command does most of the work of the procedure. It takes two arguments, an expression ($p > 0) and a body, which is another Tcl script. The while command evaluates its expression argument using rules similar to those of the C programming language and if the result is true (nonzero) then it evaluates the body as a Tcl script. It repeats this process over and over until eventually the expression evaluates to false (zero). In this case the body of the while command multiplied the result value by base and then decrements p. When p reaches zero the result contains the desired power of base. The return command causes the procedure to exit with the value of variable result as the procedure's result.

Where do commands come from?

As you have seen, all of the interesting features in Tcl are represented by commands. Statements are commands, expressions are evaluated by executing commands, control structures are commands, and procedures are commands.

Tcl commands are created in three ways. One group of commands is provided by the Tcl interpreter itself. These commands are called *builtin commands*. They include all of the commands you have seen so far and many more (see below). The builtin commands are present in all Tcl applications.

The second group of commands is created using the Tcl extension mechanism. Tcl provides APIs that allow you to create a new command by writing a *command procedure* in C or C++ that implements the command. You then register the command procedure with the Tcl interpreter by telling Tcl the name of the command that the procedure implements. In the future, whenever that particular name is used for a Tcl command, Tcl will call your command procedure to execute the command. The builtin commands are

also implemented using this same extension mechanism; their command procedures are simply part of the Tcl library.
When Tcl is used inside an application, the application incorporates its key features into Tcl using the extension mechanism. Thus the set of available Tcl commands varies from application to application. There are also numerous extension packages that can be incorporated into any Tcl application. One of the best known extensions is Tk, which provides powerful facilities for building graphical user interfaces. Other extensions provide object-oriented programming, database access, more graphical capabilities, and a variety of other features. One of Tcl's greatest advantages for building integration applications is the ease with which it can be extended to incorporate new features or communicate with other resources.
The third group of commands consists of procedures created with the proc command, such as the power command created above. Typically, extensions are used for lower-level functions where C programming is convenient, and procedures are used for higher-level functions where it is easier to write in Tcl.

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

Features of NS2
NS2 can be employed in most unix systems and windows. Most of the NS2 code is in C++. It uses TCL as its scripting language, Otcl adds object orientation to TCL.NS(version 2) is an object oriented, discrete event driven network simulator that is freely distributed and open source.
• Traffic Models: CBR, VBR, Web etc

• Protocols: TCP, UDP, HTTP, Routing algorithms,MAC etc

• Error Models: Uniform, bursty etc

• Misc: Radio propagation, Mobility models , Energy Models

• Topology Generation tools

• Visualization tools (NAM), Tracing

Structure of NS

● NS is an object oriented discrete event simulator
– Simulator maintains list of events and executes one event after another
– Single thread of control: no locking or race conditions
● Back end is C++ event scheduler

– Protocols mostly
– Fast to run, more control
• Front end is OTCL

Creating scenarios, extensions to C++ protocols
fast to write and change

Platforms

It can be employed in most unix systems(FreeBSD, Linux, Solaris) and Windows.

Source code

Most of NS2 code is in C++

Scripting language

It uses TCL as its scripting language OTcl adds object orientation to TCL.

Protocols implemented in NS2

Transport layer(Traffic Agent) – TCP, UDP
Network layer(Routing agent)
Interface queue – FIFO queue, Drop Tail queue, Priority queue
Logic link contol layer – IEEE 802.2, AR

How to use NS2

Design Simulation – Determine simulation scenario
Build ns-2 script using tcl.
Run simulation

Simulation with NS2

Define objects of simulation.
Connect the objects to each other
Start the source applications. Packets are then created and are transmitted through network.
Exit the simulator after a certain fixed time.

NS programming Structure

- Create the event scheduler
- Turn on tracing
- Create network topology
- Create transport connections
- Generate traffic
- Insert errors

ii& iii) Simulate a three-node point-to-point network with a duplex link between them. Set the queue size and vary the bandwidth and find the number of packets dropped.

STEPS:

Step1: Select the hub icon on the toolbar and drag it onto the working window.
Step2: Select the host icon on the toolbar and drag it onto the working window. Repeat this for another host icon.
Step3: Select the link icon on the toolbar and drag it on the screen from host (node 1) to the hub and again from host(node 2) to the hub. Here the hub acts as node 3 in the point-to-point network. This leads to the creation of the 3-node point-to-point network topology. Save this topology as a .tpl file.
Step4: Double-click on host(node 1), a host dialog box will open up. Click on Node editor and you can see the different layers- interface, ARP, FIFO, MAC, TCPDUMP, Physical layers. Select MAC and then select full-duplex for switches and routers and half duplex for hubs, and in log Statistics, select Number of Drop Packets, Number of Collisions, Throughput of incoming packets and Throughput of outgoing packets. Select FIFO and set the queue size to 50 and press OK. Then click on Add. Another dialog box pops up. Click on the Command box and type the Command according to the following syntax:
stg [-t duration(sec)] [-p port number]HostIPaddr
and click OK.
Step 5: Double-click on host (node 2), and follow the same step as above with only change in command according to the following syntax:
rtg [-t] [-w log] [-p port number]
and click OK.
Step 6: Double click on the link between node 1 and the hub to set the bandwidth to some initial

value say, 10 Mbps. Repeat the same for the other node.

Step 7: Click on the E button (Edit Property) present on the toolbar in order to save the changes made to the topology. Now click on the R button (RunSimulation). By doing so a user can run/pause/continue/stop/abort/disconnect/reconnect/submit a simulation. No simulation settings can be changed in this mode.

Step 8: Now go to Menu->Simulation->Run. Executing this command will submit he current simulation job to one available simulation server managed by the dispatcher. When the simulation server is executing, the user will see the time knot at the bottom of the screen move. The time knot reflects the current virtual time (progress) of the simulation case.

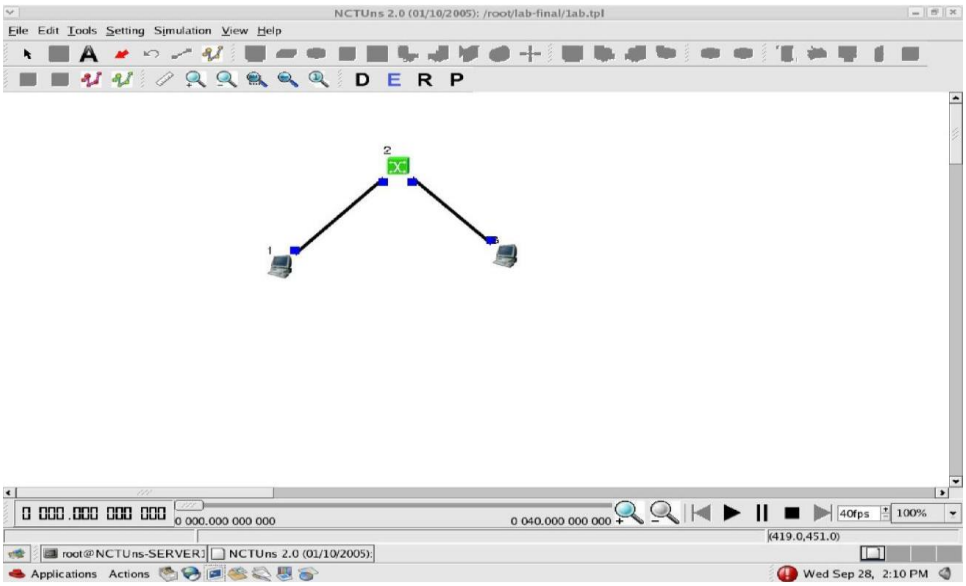
Step 9: To start the playback, the user can left-click the start icon(|>) of the time bar located at the bottom. The animation player will then start playing the recorded packet animation.

Step 10: Change the bandwidth say, 9 Mbps, and run the simulation and compare the two results.

Step 11: To view the results, go to the filename. results folder**.**

Note: To get the syntax of any command, double click on the host icon. Host dialog boxes appear and then choose App. Usage.

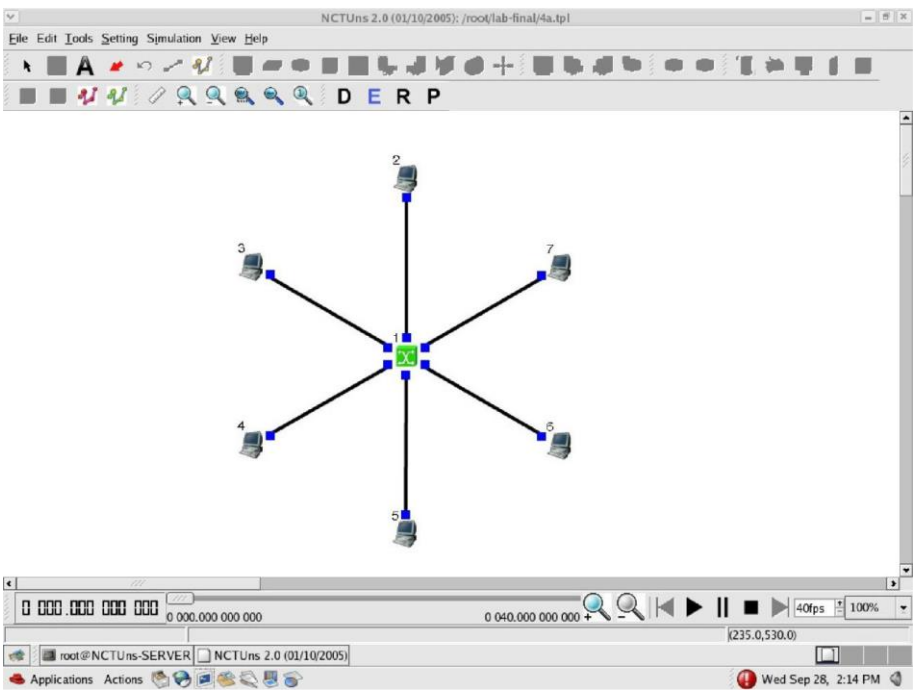
The screenshot below explain the topology.



iv)Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

STEPS:

Step 1: Click on the subnet icon on the toolbar and then click on the screen of the working window.
Step 2: Select the required number of hosts and a suitable radius between the host and the switch.
Step 3: In the edit mode, get the IP address of one of the hosts say, host 1 and then for the other host say, host2 set the drop packet and no: of collisions statistics as described in the earlier experiments.
Step 4: Now run the simulation.
Step 5: Now click on any one of the hosts and click on command console and ping the destination node.
ping IP Address of the host
Note: The no: of drop packets are obtained only when the traffic is more in the network. For checking the no of packets dropped press ctrl+C
The screenshot of the topology is shown below:

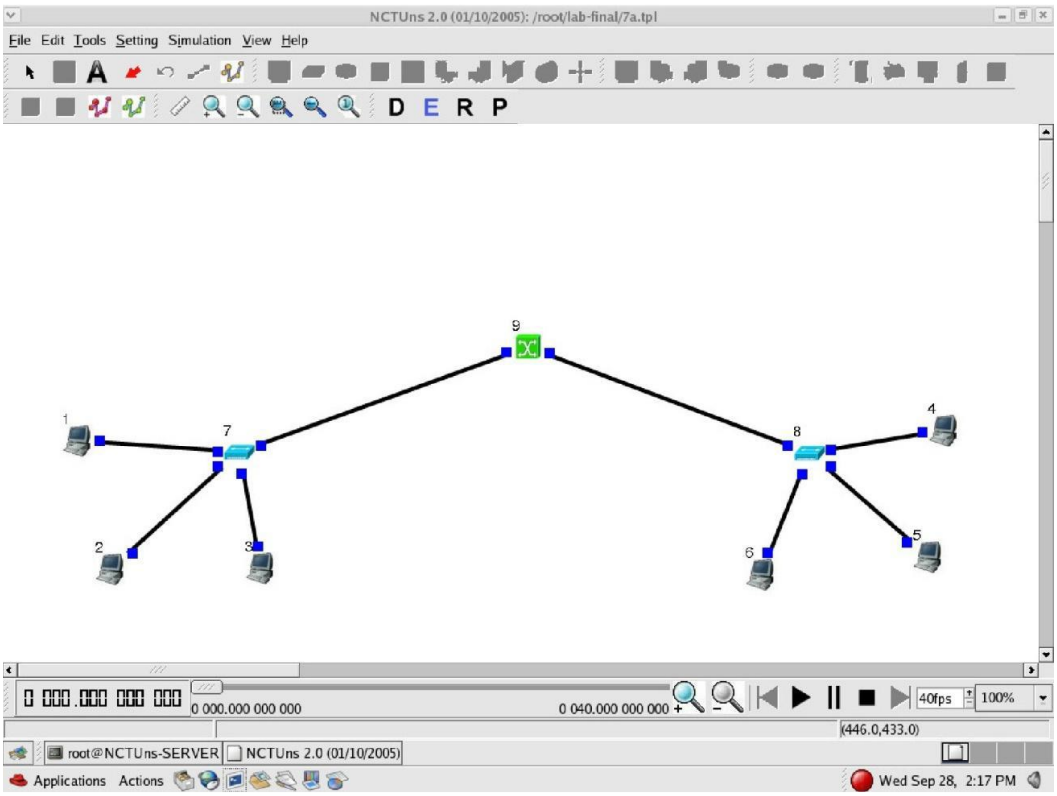


vi) Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and plot congestion window for different source/destination.

STEPS:

Step 1: Connect one set of hosts with a hub and another set of hosts also through a hub and connect these two hubs through a switch. This forms an Ethernet LAN.
Step 2: Setup multiple traffic connections between the hosts on one hub and hosts on another hub using the following command:
stcp [-p port] [-l writesize] hostIPaddr rtcp [-p port] [-l readsize]
Step 3: Setup the collision log at the destination hosts in the MAC layer as described in the earlier experiments.
Step 4: To plot the congestion window go to Menu->Tools->Plot Graph->File->open->filename.results->filename.coll.log
Step 5: View the results in the filename.results.

The screenshot of the topology is shown below:

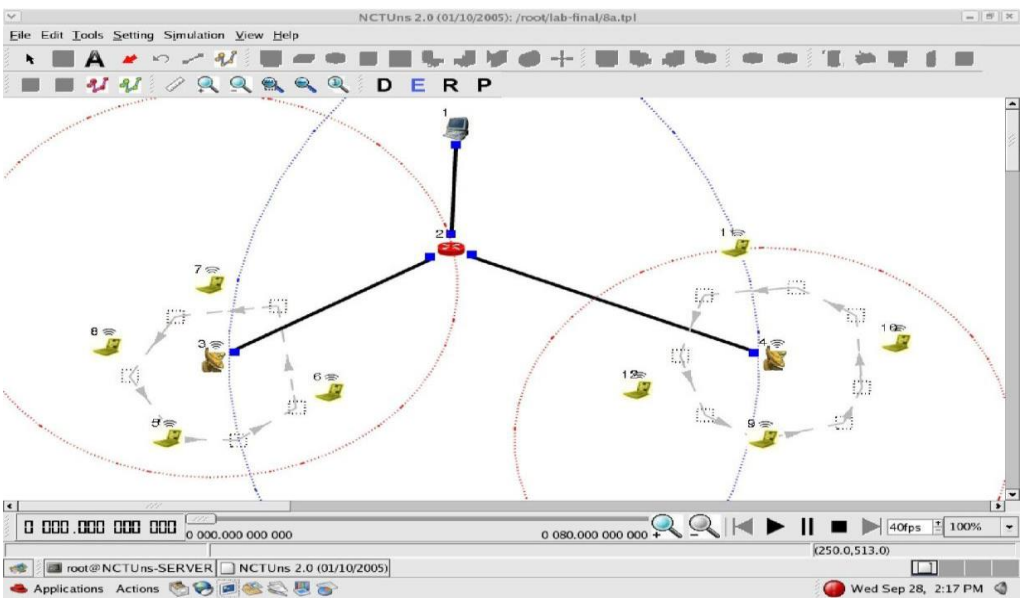


vii) Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

STEPS:

Step 1: Connect a host and two WLAN access points to a router.
Step 2: Setup multiple mobile nodes around the two WLAN access points and set the path for each mobile node.
Step 3: Setup a ttcp connection between the mobile nodes and host using the following command:
Mobile Host 1
ttcp –t –u –s –p 3000 IPAddrOf Receiver
Mobile Host 1
ttcp –t –u –s –p 4000 IPAddrOf Receiver
Host(Receiver)
ttcp –r –u –s –p 3000 ttcp –r –u –s –p 4000
Step 4: Setup the input throughput log at the destination host.
Step 5: To set the transmission range go to Menu->Settings->WLAN mobile node->Show
transmission range.
Step 5: View the results in the filename. results.

Screenshot:



ADD ON PROGRAMS

1. Implement RSA

NAME OF THE EXPERIMENT: Implementation of RSA.
OBJECTIVE: Using RSA algorithm encrypt a text data and Decrypt the same.
RESOURCE: Turbo C
PROGRAM LOGIC:
`RSA method is based on some principles from number theory. In encryption process divide the plain text into blocks, so that each plain text message p falls in the interval 0<p<n this can be done by grouping the plain text into blocks of k bits. Where k is the largest integer for which 2 power k <n is true. The security of this method is based on the difficulty of factoring large numbers. The encryption and decryption functions are inverses
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

SOURCE CODE:

```
/*Using RSA algorithm encrypt a text data and Decrypt the same*/
#include<stdio.h>
#include<conio.h>
#include<math.h>
main()
{
long C,C1,T,P,Q,N,Z,Z1,E,D,De,i,k,rem,flag=0;
char text,c;
clrscr();
printf("enter the text:");
scanf("%c",&text);
T=text-64;
printf("enter the values of PandQ:");
scanf("%ld%ld",&P,&Q);
N=P*Q;
Z=(P-1)*(Q-1);
Z1=Z;
for(i=2;i<Z;i++) {
k=i;
Z1=Z;
while(1) {
rem=Z1%k;
if(rem==0&&k==1)
{
D=i;
flag=1;
```

```
break;
}
else
{
Z1=k;
k=rem;
if(k==0)
break;
}
}
if(flag==1)
{

break;
}
}
E=(abs(Z)+1)/D;
C1=pow(T,E);
C=C1%N;
printf("\nC:%ld\n",C);
c=C+64;
printf("%c",c);
printf("T:%ld\tP:%ld\TQ:%ld\tN:%ld\tZ:%ld\tD:%ld\tE:%ld\t",T,P,Q,N,Z,D,E);
printf("\n after decrypt....");
De=pow(C,D);
De=De%N;
De=De+64;
printf("\n%c",De);
getch();
}
```

OUTPUT:

```
[187y1a1230@mlritm-programmingserver ~]$ vi rsa.c
[187y1a1230@mlritm-programmingserver ~]$ cc -lm rsa.c
[187y1a1230@mlritm-programmingserver ~]$ ./a.out
enter the text:s
enter the values of pandQ:5 10

C:39
qT:19       p:5         Q:10        N:50        Z:36        D:5    `E:7
after decrypt....
q[187y1a1230@mlritm-programmingserver ~]$
```

Viva Questions:

1. Expand RSA
2. What is encryption and decryption in RSA?
3. To encrypt a message P, Compute c=----------?
4. To compute c compute P=----------?
5. Define cryptography.

2. Implements Rail Fence Cipher

AIM: To write a C program that implements Rain Fence Cipher

ALOGRITHM:

In the rail fence cipher, the plain text is written downwards and diagonally on
successive “rails” of an imaginary fence, then moving up when we reach the bottom rail.
When we reach the top rail, the message is written downwards again until the whole plaintext is written out.

PROGRAM:

```
#include<stdio.h>
int main()
{
  char str[20], str1[10]="", str2[10]="";
int i, cnt1=0, cnt2=0;

printf("Enter your plain text:");
gets(str);

for(i=0; i<strlen(str); i++)
  {
if( i%2 == 0)
    {
str1[cnt1++]=str[i];
    }
else
str2[cnt2++]=str[i];
    }
printf("Encrypted Text = %s%s",str1,str2);
return 0;
}
```

OUTPUT:

```
[187y1a1230@mlritm-programmingserver ~]$ vi rain.c
[187y1a1230@mlritm-programmingserver ~]$ cc rain.c
rain.c: In function 'main':
rain.c:8:3: warning: 'gets' is deprecated (declared at /usr/include/stdio.h:638)
 [-Wdeprecated-declarations]
   gets(str);
   ^
rain.c:10:14: warning: incompatible implicit declaration of built-in function 's
trlen' [enabled by default]
   for(i=0; i<strlen(str); i++)
              ^
/tmp/ccGFrdSC.o: In function `main':
rain.c:(.text+0x4a): warning: the `gets' function is dangerous and should not be
 used.
[187y1a1230@mlritm-programmingserver ~]$ ./a.out
Enter your plain text:priyanka
Encrypted Text = piakryna[187y1a1230@mlritm-programmingserver ~]$
```

3. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.

4. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

NS3 LTE Simulation

LTE is the latest high-speed cellular transmission network.LTE is a 4G technology with download speeds that run the gamut from 3 to 28 Mbps worldwide.4G LTE is one of several competing 4G standards along with Ultra Mobile Broadband (UMB) andWiMax (IEEE
802.16).Ns3 is the best choice among network simulator for simulating LTE framework.We provide ***customized NS3 LTE Simulation Projects*** based on customer Requirements.

Advantages of LTE:

• LTE will supports seamless connection to existing networks like GSM,CDMA and

WCDMA.

• It has simple architecture because of low operating expenditure

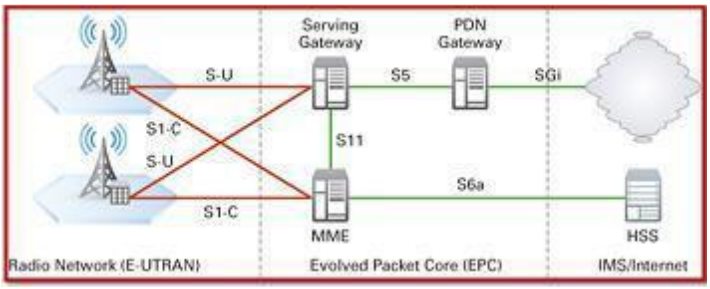
• Time required for connecting network and is in range of a few hundred ms and power savings states can now be entered and exited very quickly

• High data rates can be achieved in both downlink as well as uplink.

• Both FDD and TDD can be used on same platform.

• Optimized signaling for connection establishment and other air interface and mobility management procedures have further improved the user experience.

Architecture of LTE:



LTE parameters:

• Transmission bandwidth.
• Mobility.
• Frequency range.
• Duplexing.
• Channel bandwidth.
• Channel coding.
• MIMO.
• Multi-antenna technology.

Sample code for LTE:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/config-store-module.h"
using namespace ns3;
int main (int argc, char *argv[])
{
CommandLine cmd;
cmd.Parse (argc, argv);
ConfigStore inputConfig;
inputConfig.ConfigureDefaults ();
cmd.Parse (argc, argv);
Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
lteHelper->SetAttribute ("PathlossModel", StringValue
("ns3::FriisSpectrumPropagationLossModel"));
NodeContainer enbNodes;
NodeContainer ueNodes;
enbNodes.Create (1);
ueNodes.Create (3);
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel"); mobility.Install
(enbNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (ueNodes);
NetDeviceContainer enbDevs;
NetDeviceContainer ueDevs;
```

```
enbDevs = lteHelper->InstallEnbDevice (enbNodes); ueDevs = lteHelper->InstallUeDevice (ueNodes); lteHelper->Attach (ueDevs, enbDevs.Get (0));
enum EpsBearer::Qci q = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearer (q);
lteHelper->ActivateDataRadioBearer (ueDevs, bearer);
Simulator::Stop (Seconds (0.5));
lteHelper->EnablePhyTraces ();
lteHelper->EnableMacTraces ();
lteHelper->EnableRlcTraces ();
double distance_temp [] = { 1000,1000,1000};
std::vector<double> userDistance;
userDistance.assign (distance_temp, distance_temp + 3);
for (int i = 0; i < 3; i++)
{
Ptr<ConstantPositionMobilityModel > mm = ueNodes.Get (i)->GetObject<ConstantPositionMobilityModel> (); mm->SetPosition (Vector (userDistance[i], 0.0, 0.0)); }
Simulator::Run ();
Simulator::Destroy ();
return 0;
    }
```