List of Experiments

SNO	PROGRAM
1	**Week 1-:** Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.
2	**Week 2 -:** Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP
3	**Week 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.
4	**Week 4 - :** Implement Dijsktra's algorithm to compute the shortest path through a network
5	**Week 5- :** Take an example subnet of hosts and obtain a broadcast tree for the subnet.
6	**Week 6- :** Implement distance vector routing algorithm for obtaining routing tables at each node.
7	**Week 7- :** Implement data encryption and data decryption.
8	**Week 8- :** Write a program for congestion control using Leaky bucket algorithm.
9	**Week 9 - :** Write a program for frame sorting technique used in buffers.
10	**Week 10- :** Wireshark i. Packet Capture Using Wire shark ii. Starting Wire shark iii. Viewing Captured Traffic iv. Analysis and Statistics & Filters.
11	**Week 11- :** How to run Nmap scan
12	/**Week 12 - :** Operating System Detection using Nmap
13	/**Week 13 - :** 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. vi. Simulate to Plot Congestion for Different Source/Destination vii. Simulate to Determine the Performance with respect to Transmission of Packets

EXPERIMENT NO: 1. (a)

NAME OF THE EXPERIMENT: Bit Stuffing.

AIM: Implement the data link layer framing methods such as and bit stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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. When ever the senders data link layer encounters five consecutive one's in the data, it automatically stuffs a 0 bit into the out going bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the out going character stream before DLE in the data

ALGORITHM:

Begin

Step 1: Read frame length n

Step 2: Repeat step (3 to 4) until i<n(**: R**ead values in to the input frame (0's and 1'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

SOURCE CODE:

```
// BIT Stuffing program
#include<stdio.h>
#include<string.h>
void main()
{
int a[20],b[30],i,j,k,count,n;
printf("Enter frame length:");
scanf("%d",&n);
printf("Enter input frame (0's & 1's only):");
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 stuffing the frame is:");
for(i=0;i<j;i++)
printf("%d",b[i]);
}
```

OUTPUT:

```
Enter frame length:5
Enter input frame (0's & 1's only):
1
1
1
1
1
After stuffing the frame is:111110
------------------
(program exited with code: 6)
Press return to continue
```

VIVA QUESTIONS:

1. What is bit stuffing?

2. What is the use of bit stuffing?

3. with bit stuffing the boundary b/w 2 frames can be unambiguously recognized by ------------------------

4. -------------------- is analogous to character stuffing

5. Each frame begins and ends with a special bit pattern 01111110 called ---------

6. The senders data link layer encounters ----------------no of 1’s consecutively

EXPERIMENT NO: 1. (b)

NAME OF THE EXPERIMENT: Character Stuffing.

AIM: Implement the data link layer framing methods such as character, character stuffing.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:**-RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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

ALGORITHM:

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: end if

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: end if

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: end if

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 stiuffing

Step 25: print b

End

SOURCE CODE:

```
//PROGRAM FOR CHARACTER STUFFING
#include<stdio.h>
#include<string.h>
#include<process.h>
void main()
{
int i=0,j=0,n,pos;
char a[20],b[50],ch;
printf("enter string\n");
scanf("%s",&a);
n=strlen(a);
printf("enter position\n");
scanf("%d",&pos);
if(pos>n)
{
printf("invalid position, Enter again :");
scanf("%d",&pos);}
```

```
printf("enter the character\n");
ch=getche();
b[0]='d';
b[1]='l';
b[2]='e';
b[3]='s';
b[4]='t';
b[5]='x';
j=6;
while(i<n)
{
if(i==pos-1)
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]=ch;
b[j+4]='d';
b[j+5]='l';
b[j+6]='e';
j=j+7;
}
if(a[i]=='d' && a[i+1]=='l' && a[i+2]=='e')
{
b[j]='d';
b[j+1]='l';
b[j+2]='e';
j=j+3;
}
b[j]=a[i];
i++;
j++;
}
b[j]='d';
b[j+1]='l';
b[j+2]='e';
b[j+3]='e';
b[j+4]='t';
b[j+5]='x';
b[j+6]='\0';
printf("\nframe after stuffing:\n");
printf("%s",b);
}
```

OUTPUT:

```
enter string
MLRITM
enter position
2
enter the character
frame after stuffing:
dlestxMdldleLRITMdleetx
------------------
(program exited with code: 0)
Press return to continue
```

VIVA QUESTIONS:

1. What is character stuffing?
2. What is the use of character stuffing?
3. ___________ is analogous to bit stuffing.
4. ____________ are the delimiters for character stuffing
5. Expand DLE STX___________
6. Expand DLE ETX__________

EXPERIMENT NO: 2.

NAME OF THE EXPERIMENT: Cyclic Redundancy Check.

AIM: Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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 some times known as Cyclic Redundancy Code

ALGORITHM/FLOWCHART:

```
Begin
Step 1: Declare I,j,fr[8],dupfr[11],recfr[11],tlen,flag,gen[4],genl,frl,
rem[4] as integer
Step 2: initialize frl=8 and genl=4
Step 3: initialize i=0
Step 4: Repeat step(5to7) until i<frl
Step 5: read fr[i]
Step 6: dupfr[i]=fr[i]
Step 7: increment i
Step 8: initialize i=0
Step 9: repeat step(10to11) until i<genl
Step 10: read gen[i]
Step 11: increment i
Step 12: tlen=frl+genl-1
Step 13: initialize i=frl
Step 14: Repeat step(15to16) until i<tlen
Step 15: dupfr[i]=0
Step 16: increment i
Step 17: call the function remainder(dupfr)
Step 18: initialize i=0
Step 19: repeat step(20 to 21) until j<genl
Step 20: recfr[i]=rem[j]
Step 21: increment I and j
Step 22: call the function remainder(dupfr)
Step 23: initialize flag=0 and i=0
Step 24: Repeat step(25to28) until i<4
Step 25: if rem[i]!=0 then
Step 26: increment flag
Step 27: end if
Step 28: increment i
Step 29: if flag=0 then
Step 25: print frame received correctly
Step 25: else
Step 25: print the received frame is wrong
End
Function: Remainder(int fr[])
Begin
Step 1: Declare k,k1,I,j as integer
Step 2: initialize k=0;
Step 3: repeat step(4 to 14) until k< frl
Step 4: if ((fr[k] == 1) then
Step 5: k1=k
Step 6: initialize i=0, j=k
Step 7: repeat step(8 to 9) until i< genl
Step 8: rem[i] =fr[j] exponential gen[i]
```

Step 9: increment I and j

Step 10: initialize I = 0

Step 11: repeat step(12to13) until I <genl

Step 12: fr[k1] = rem[i]

Step 13: increment k1 and i

Step 14: end if

End

SOURCE CODE:

```
//PROGRAM FOR CYCLIC REDUNDENCY CHECK
#include<stdio.h>
int gen[4],genl,frl,rem[4];
void main()
{
int i,j,fr[8],dupfr[11],recfr[11],tlen,flag;
frl=8; genl=4;
printf("enter frame:");
for(i=0;i<frl;i++)
{
scanf("%d",&fr[i]);
dupfr[i]=fr[i];
}
printf("enter generator:");
for(i=0;i<genl;i++)
scanf("%d",&gen[i]);
tlen=frl+genl-1;
for(i=frl;i<tlen;i++)
{
dupfr[i]=0;
}
```

```
remainder(dupfr);
for(i=0;i<frl;i++)
{
recfr[i]=fr[i];
}
for(i=frl,j=1;j<genl;i++,j++)
{
recfr[i]=rem[j];
}
remainder(recfr);
flag=0;
for(i=0;i<4;i++)
{
if(rem[i]!=0)
flag++;
}
if(flag==0)
{
printf("frame received correctly");
}
else
{
printf("the received frame is wrong");
}
}
remainder(int fr[])
{
int k,k1,i,j;
for(k=0;k<frl;k++)
{
```

```
if(fr[k]==1)
{
k1=k;
for(i=0,j=k;i<genl;i++,j++)
{
rem[i]=fr[j]^gen[i];
}
for(i=0;i<genl;i++)
{
fr[k1]=rem[i];
k1++;
}
}
}
}
```

OUTPUT:

```
enter frame:
1
1
0
1
0
1
1
0
enter generator:
1
0
1
1
Frame received correctly.
```

VIVA QUESTIONS:

1. What is CRC?
2. What is the use of CRC?
3. Name the CRC standards
4. Define checksum?
5. Define generator polynomial?
6. Polynomial arithmetic is done by_______

XPERIMENT NO: 3

NAME OF THE EXPERIMENT: flow control using the sliding window protocol
AIM: 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.

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

   if(f%w!=0)
      printf("\nAcknowledgement of above frames sent is received by sender\n");

   return 0;
}
```

Output:

```
Enter window size: 3

Enter number of frames to transmit: 5
Enter 5 frames: 12 5 89 4 6

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

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

12 5 89
Acknowledgement of above frames sent is received by sender

4 6
Acknowledgement of above frames sent is received by sender
```

VIVA QUESTIONS:

1. What is flow control?
2. What is sliding window protocol?
3. What is sliding window Go-back-N mechanism?
4. What one.bit sliding window protocol?

XPERIMENT NO: 4

NAME OF THE EXPERIMENT: Shortest Path.

AIM: Implement Dijkstra‘s algorithm to compute the Shortest path thru a given graph.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

ALGORITHM/FLOWCHART:

Begin

```
Step1: Declare array path [5] [5], min, a [5][5], index, t[5];
Step2: Declare and initialize st=1,ed=5
Step 3: Declare variables i, j, stp, p, edp
Step 4: print “enter the cost “
Step 5: i=1
Step 6: Repeat step (7 to 11) until (i<=5)
Step 7: j=1
Step 8: repeat step (9 to 10) until (j<=5)
Step 9: Read a[i] [j]
Step 10: increment j
Step 11: increment i
Step 12: print “Enter the path”
Step 13: read p
Step 14: print “Enter possible paths”
Step 15: i=1
Step 16: repeat step(17 to 21) until (i<=p)
Step 17: j=1
Step 18: repeat step(19 to 20) until (i<=5)
Step 19: read path[i][j]
Step 20: increment j
Step 21: increment i
Step 22: j=1
Step 23: repeat step(24 to 34) until(i<=p)
Step 24: t[i]=0
Step 25: stp=st
Step 26: j=1
Step 27: repeat step(26 to 34) until(j<=5)
Step 28: edp=path[i][j+1]
Step 29: t[i]= [ti]+a[stp][edp]
Step 30: if (edp==ed) then
Step 31: break;
Step 32: else
Step 33: stp=edp
Step 34: end if
Step 35: min=t[st]
Step 36: index=st
Step 37: repeat step( 38 to 41) until (i<=p)
Step 38: min>t[i]
Step 39: min=t[i]
Step 40: index=i
Step 41: end if
Step 42: print” minimum cost” min
Step 43: print" minimum cost pth"
Step 44: repeat step(45 to 48) until (i<=5)
Step 45: print path[index][i]
Step 46: if(path[idex][i]==ed) then
Step 47: break
Step 48: end if
End
```

SOURCE CODE:

```
//*********************************
// .PROGRAM FOR FINDING SHORTEST //PATH FOR A GIVEN GRAPH
//*********************************
#include<stdio.h>
void main()
{
int path[5][5],i,j,min,a[5][5],p,st=1,ed=5,stp,edp,t[5],index;
printf("enter the cost matrix\n");
for(i=1;i<=5;i++)
for(j=1;j<=5;j++)
scanf("%d",&a[i][j]);
printf("enter the paths\n");
scanf("%d",&p);
printf("enter possible paths\n");
for(i=1;i<=p;i++)
for(j=1;j<=5;j++)
scanf("%d",&path[i][j]);
for(i=1;i<=p;i++)
{
t[i]=0;
stp=st;
for(j=1;j<=5;j++)
{
edp=path[i][j+1];
t[i]=t[i]+a[stp][edp];
if(edp==ed)
break;
else
stp=edp;
}
}
min=t[st];index=st;
for(i=1;i<=p;i++)
{
if(min>t[i])
{
min=t[i];
index=i;
}
}
printf("minimum cost %d",min);
printf("\n minimum cost path ");
for(i=1;i<=5;i++)
{
printf("--> %d",path[index][i]);
if(path[index][i]==ed)
break;
}
}
```

OUTPUT:

```
enter the cost matrix
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
enter the paths
2
enter possible paths
1 2 3 4 5
1 2 3 4 5
minimum cost 14
```

VIVA QUESTIONS:

EXPERIMENT NO: 5

NAME OF THE EXPERIMENT: Broadcast Tree.

AIM: Implement broadcast tree for a given subnet of hosts

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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.

ALGORITHM/FLOWCHART:

```
step 1: declare variable as int p,q,u,v,n;
step 2: Initialize min=99,mincost=0;
step 3: declare variable as int t[50][2],i,j;
step 4: declare variable as int parent[50],edge[50][50];
step 5: Begin
step 6: write "Enter the number of nodes"
step 7: read "n"
step 8: Initialize i=0
step 9: repeat step(10-12) until i<n
step10: increment i
step11: write"65+i"
step12: Initialize parent[i]=-1
step13:wite "\n"
step14: Initialize i=0
step15: repeat step(15-21) until i<n
step16: increment i
step17: write"65+i"
step18: Initialize j=0
```

```
step19: repeat until j<n
step20: increment j
step21: read edge[i][j]
step22: Initialize i=0
step23: repeat step(23-43) until i<n
step24: increment i
step25: Initialize j=0
step26: repeat until j<n
step27: increment j
step28: if'edge[i]j]!=99
step29: if'min>edge[i][j] repeat step (29-32)
step30: intialize min=edge[i][j]
step31: intialize u=i
step32: intialize v=j
step33: calling function p=find(u);
step34: calling function q=find(v);
step35: if'P!=q repeat steps(35-39)
step36: intialize t[i][0]=u
step37: intialize t[i][1]=v
step38: initialize mincost=mincost+edge[u][v]
step39: call function sunion(p,q)
step40: else repeat steps(40-42)
step41: Intialize t[i][0]=-1;
step42: Intialize t[i][1]=-1;
step43: intialize min=99;
step44; write"Minimum cost is %d\n Minimum spanning tree is",mincost
step45: Initialize i=0
step46: repeat until i<n
step47: increment i
step48: if't[i][0]!=-1 && t[i][1]!=-1'repeat step(48-50)
```

```
step49: write "%c %c %d", 65+t[i][0], 65+t[i][1], edge[t[i][0]][t[i][1]]
step50: write"\n"
step51: end
step52: called function sunion(int l,int m) repeat step(51-52)
step53: intialize parent[l]=m
step54: called function find(int l) repeat step(53-56)
step55: if parent([l]>0)
step56: initialize l=parent
step57: return l
```

SOURCE CODE:

```
// Write a ‘c’ program for Broadcast tree from subnet of host
#include<stdio.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");
}
}
sunion(int l,int m)
{
parent[l]=m;
}
find(int l)
{
if(parent[l]>0)
l=parent[l];
return l;
}
```

OUTPUT:

```
Enter the number of nodes3
A       B       C
A1 2 3 4
B1 2 3 4
C4 5 6 7
Minimum cost is 3
Minimum spanning tree is
C A 3
```

VIVA QUESTIONS:

1. What is spanning tree
2. What is broad cast tree?
3. What are the advantages of broad cast tree?
4. Where we should use the broad cast tree

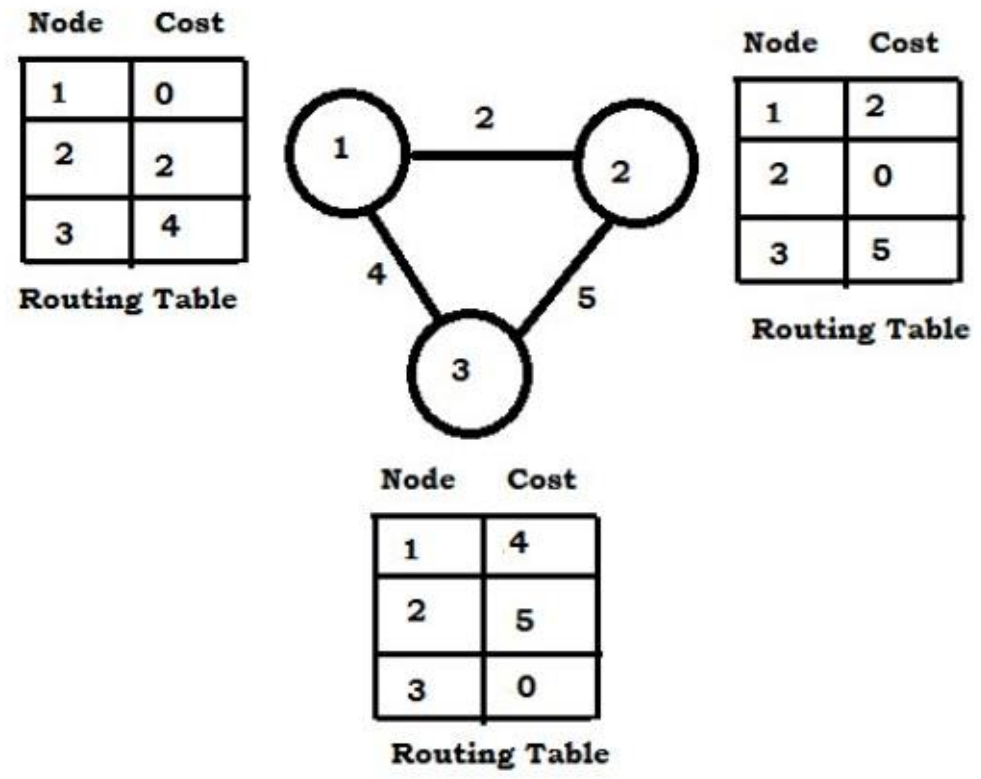
5. What is flooding?

6. What is the subnet?

EXPERIMENT NO: 6

NAME OF THE EXPERIMENT: Distance Vector routing.

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



Node	Cost
1	0
2	2
3	4

Node	Cost
1	2
2	0
3	5

Node	Cost
1	4
2	5
3	0

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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.

ALGORITHM:

```
Begin
Step1: Create struct node unsigned dist[20],unsigned from[20],rt[10]
Step2: initialize int dmat[20][20], n,i,j,k,count=0,
Step3: write "the number of nodes "
Step4: read the number of nodes "n"
Step5: write" the cost matrix :"
Step6: intialize i=0
Step7: repeat until i<n
Step8: increment i
Step9: initialize j=0
Step10: repeat Step(10-16)until j<n
Step11: increment j
Step12:read dmat[i][j]
Step13:intialize dmat[i][j]=0
Step14:intialize rt[i].dist[j]=dmat[i][j]
Step15:intialize rt[i].from[j]=j
Step16:end
Step17:start do loop Step (17-33)until
Step18:intilialize count =0
Step19:initialize i=0
Step20:repeat until i<n
Step21:increment i
Step22:initialize j=0
Step23:repeat until j<n
Step24:increment j
Step25:initialize k=0
Step26:repeat until k<n
Step27:increment k
Step28:if repeat Step(28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]
```

```
Step29:intialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]
Step30:intialize rt[i].from[j]=k;
Step31:increment count
Step32:end if
Step33:end do stmt
Step34:while (count!=0)
Step35:initialize i=0
Step36:repeat Steps(36-44)until i<n
Step37:increment i
Step38:write ' state values for router',i+1
Step39:initialize j=0
Step40:repeat Steps ( 40-43)until j<n
Step41:increment j
Step42:write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]
Step43:end
Step44:end
end
```

SOURCE CODE:

```c
#include<stdio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
int dmat[20][20];
int n,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
```

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

OUTPUT:

```
Enter the number of nodes : 2
Enter the cost matrix :
1 2
1 2
State value for router 1 is
node 1 via 1 Distance0
node 2 via 2 Distance2
State value for router 2 is
node 1 via 1 Distance1
node 2 via 2 Distance0
```

VIVA QUESTIONS:

1. What is routing
2. What is best algorithm among all routing algorithms?
3. What is static routing?
4. Difference between static and dynamic
5. How distance vector routing works
6. What is optimality principle?

EXPERIMENT NO: 7

NAME OF THE EXPERIMENT: data encryption and data decryption.

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

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

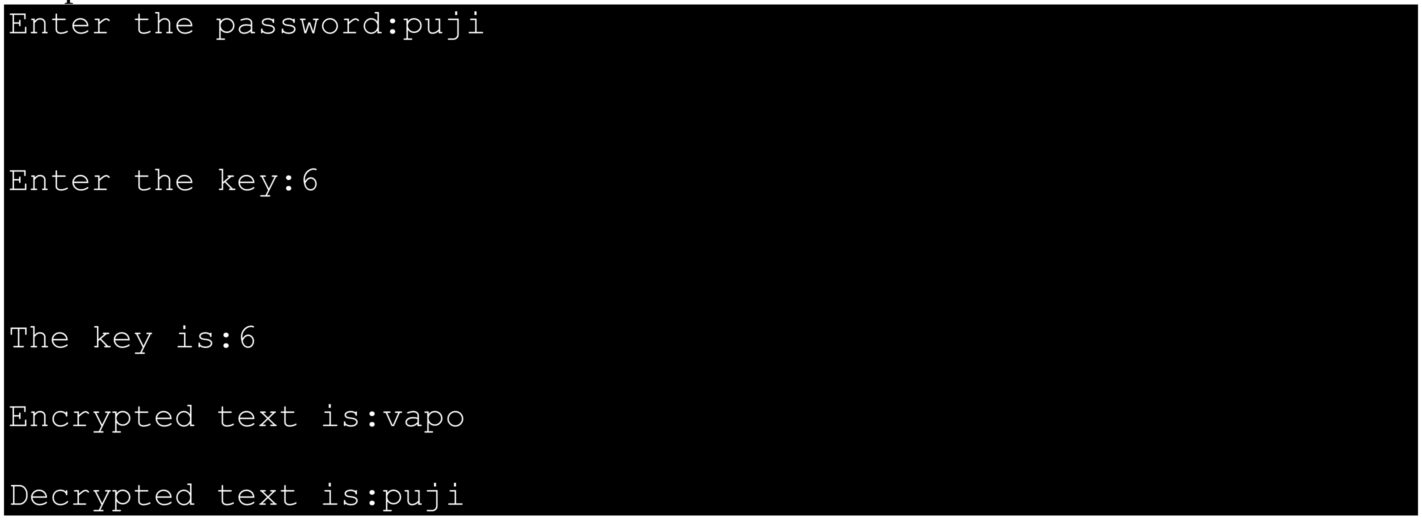
SOFTWARE REQUIREMENTS: Turbo C / Borland C.

THEORY:

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

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
void main()
{
        char pwd[20];
        char alpha[26]="abcdefghijklmnopqrstuvwxyz";
        int num[20],i,n,key;
        //clrscr();
        printf("\nEnter the password:");
        scanf("%s",&pwd);
        n=strlen(pwd);
        for(i=0;i<n;i++)
                num[i]=toascii(tolower(pwd[i]))-'a';
        printf("\nEnter the key:");
        scanf("%d",&key);
        for(i=0;i<n;i++)
                num[i]=(num[i]+key)%26;
        for(i=0;i<n;i++)
                pwd[i]=alpha[num[i]];
        printf("\nThe key is:%d",key);
        printf("\nEncrypted text is:%s",pwd);
        for(i=0;i<n;i++)
        {
                num[i]=(num[i]-key)%26;
                if(num[i]<0)
```

```
                        num[i]=26+num[i];
                pwd[i]=alpha[num[i]];
        }
        printf("\nDecrypted text is:%s",pwd);
        getch();
}
```
Output:



```
Enter the password:puji



Enter the key:6



The key is:6

Encrypted text is:vapo

Decrypted text is:puji
```

VIVA QUESTIONS:

1. Expand DES_____________
2. What is cipher text?
3. What is plain text?
4. Define public key?
5. Define encryption?
6. Substitutions are performed by____________boxes

EXPERIMENT NO: 8

NAME OF THE EXPERIMENT: congestion control using Leaky bucket algorithm

AIM: Implement a program for congestion control using Leaky bucket algorithm.

HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

Program

```
#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

```
Enter bucket size, outgoing rate and no of inputs: 50 40 3

Enter the incoming packet size : 40

Incoming packet size 40

Bucket buffer size 40 out of 50

After outgoing 0 packets left out of 50 in buffer

Enter the incoming packet size : 20

Incoming packet size 20

Bucket buffer size 20 out of 50

After outgoing -
20 packets left out of 50 in buffer

Enter the incoming packet size : 60

Incoming packet size 60

Bucket buffer size 40 out of 50

After outgoing 0 packets left out of 50 in buffer
```

VIVA QUESTIONS:

1. What is Congestion?
2. What is Congestion Control?
3. What are Congestion control algorithms?
4. Differentiate between Leaky bucket and Token bucket?
5. What is the purpose of leaky bucket algorithm?

EXPERIMENT NO: 9

NAME OF THE EXPERIMENT: frame sorting technique used in buffers.
AIM: Implement frame sorting technique used in buffers.

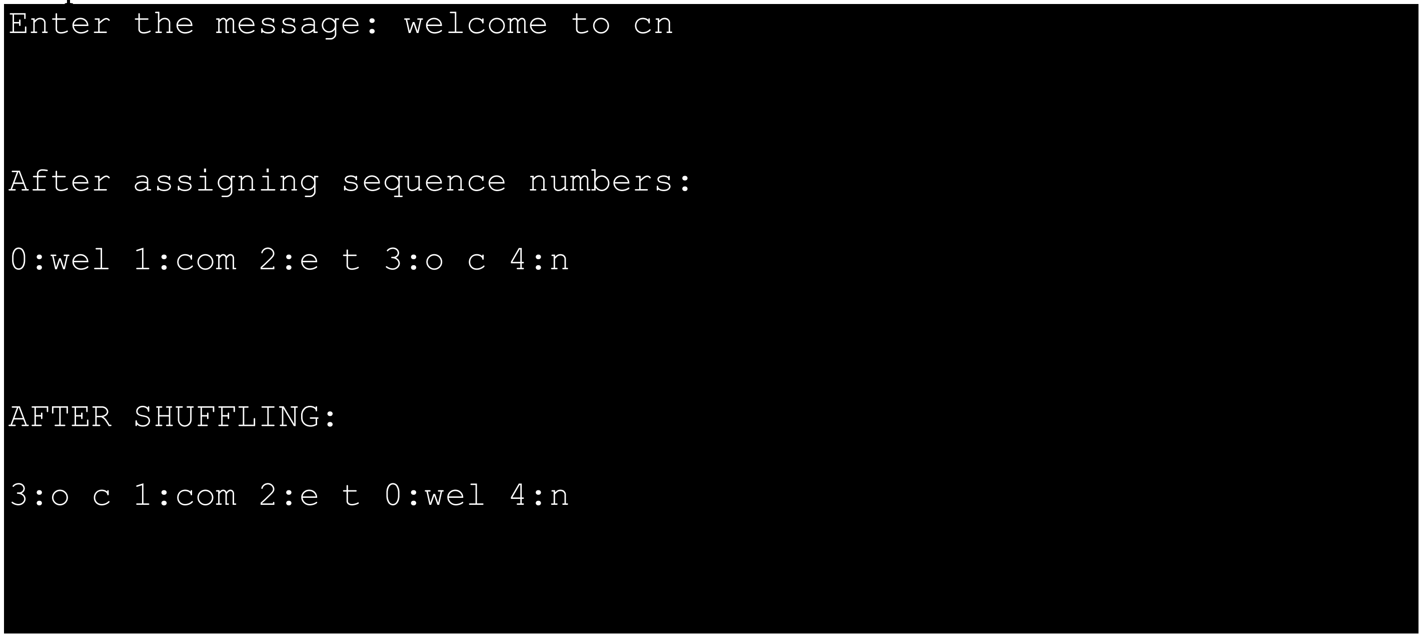
HARDWARE REQUIREMENTS: Intel based Desktop PC**:-** RAM of 512 MB

SOFTWARE REQUIREMENTS: Turbo C / Borland C.

```
#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");
}
```

Output:



```
Enter the message: welcome to cn



After assigning sequence numbers:

0:wel 1:com 2:e t 3:o c 4:n



AFTER SHUFFLING:

3:o c 1:com 2:e t 0:wel 4:n
```

EXPERIMENT NO: 10

NAME OF THE EXPERIMENT: Wireshark

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

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.

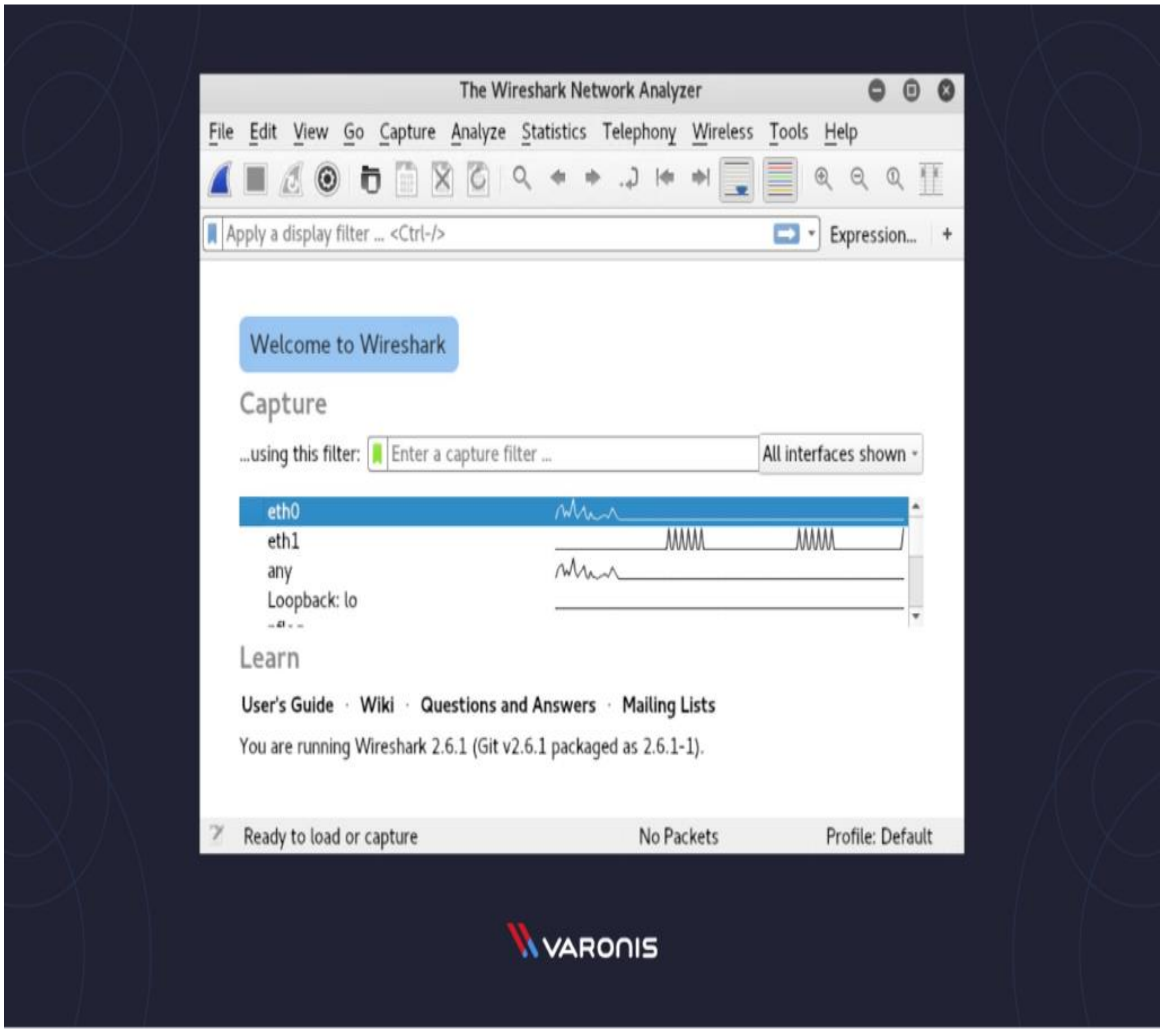
Ed. Note: A "packet" is a single message from any network protocol (i.e., TCP, DNS, etc.)
Ed. Note 2: LAN traffic is in broadcast mode, meaning a single computer with Wireshark can see traffic between two other computers. If you want to see traffic to an external site, you need to capture the packets 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.

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.

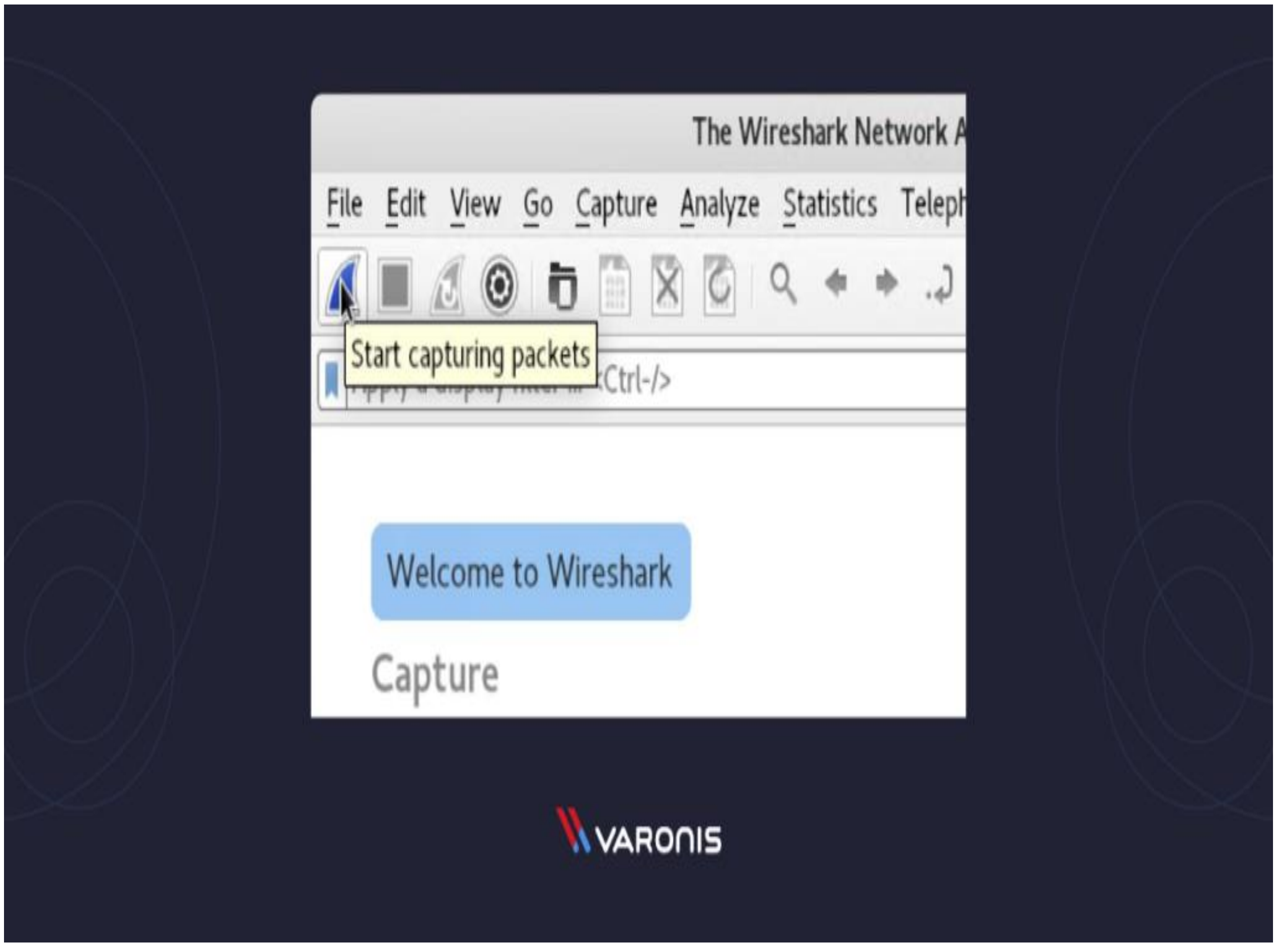
Capturing Data Packets on Wireshark

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.

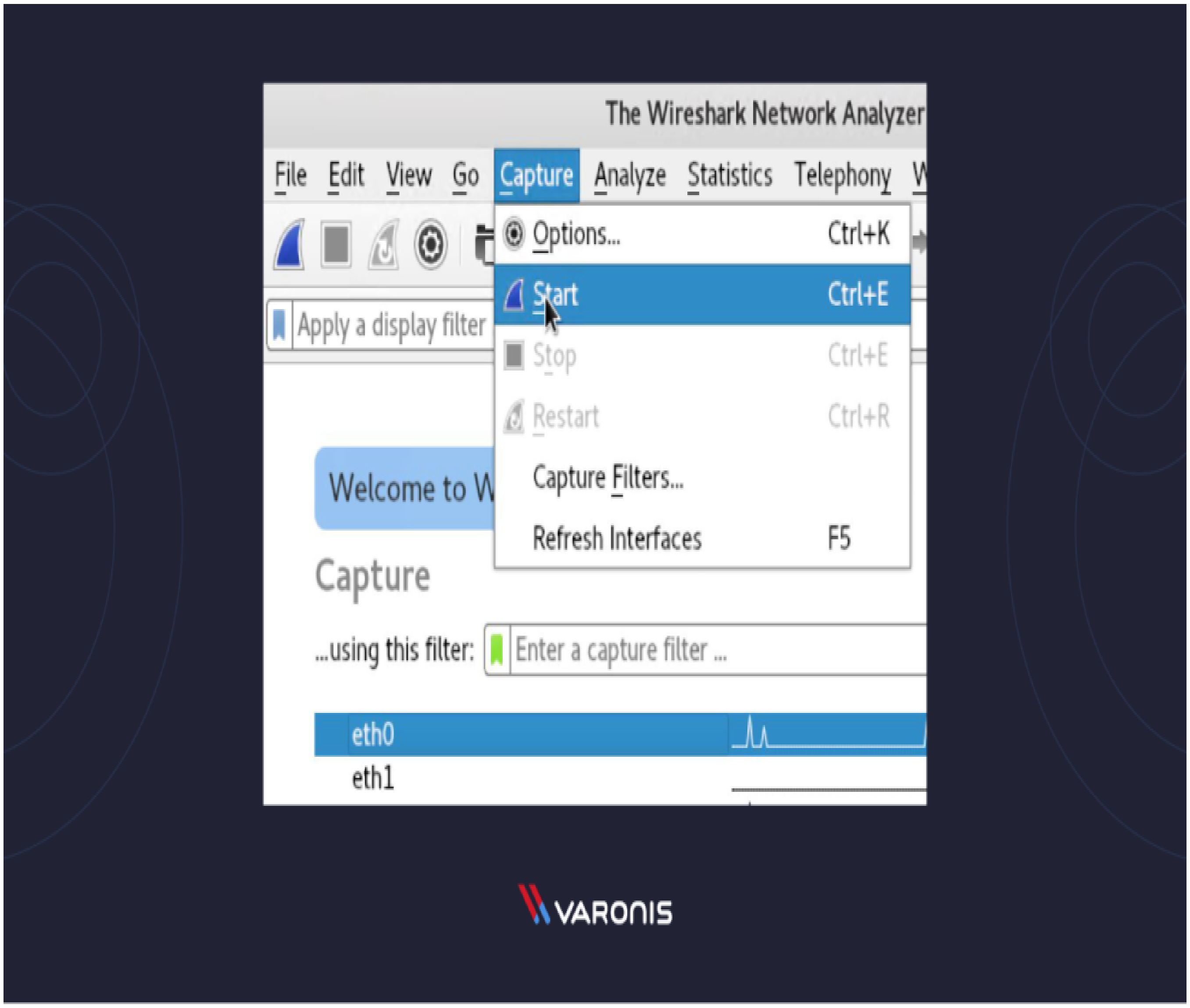

The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Welcome to Wireshark
Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
eth0
eth1
any
Loopback: lo
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists
You are running Wireshark 2.6.1 (Git v2.6.1 packaged as 2.6.1-1).
Ready to load or capture
No Packets
Profile: Default
VARONIS

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.”

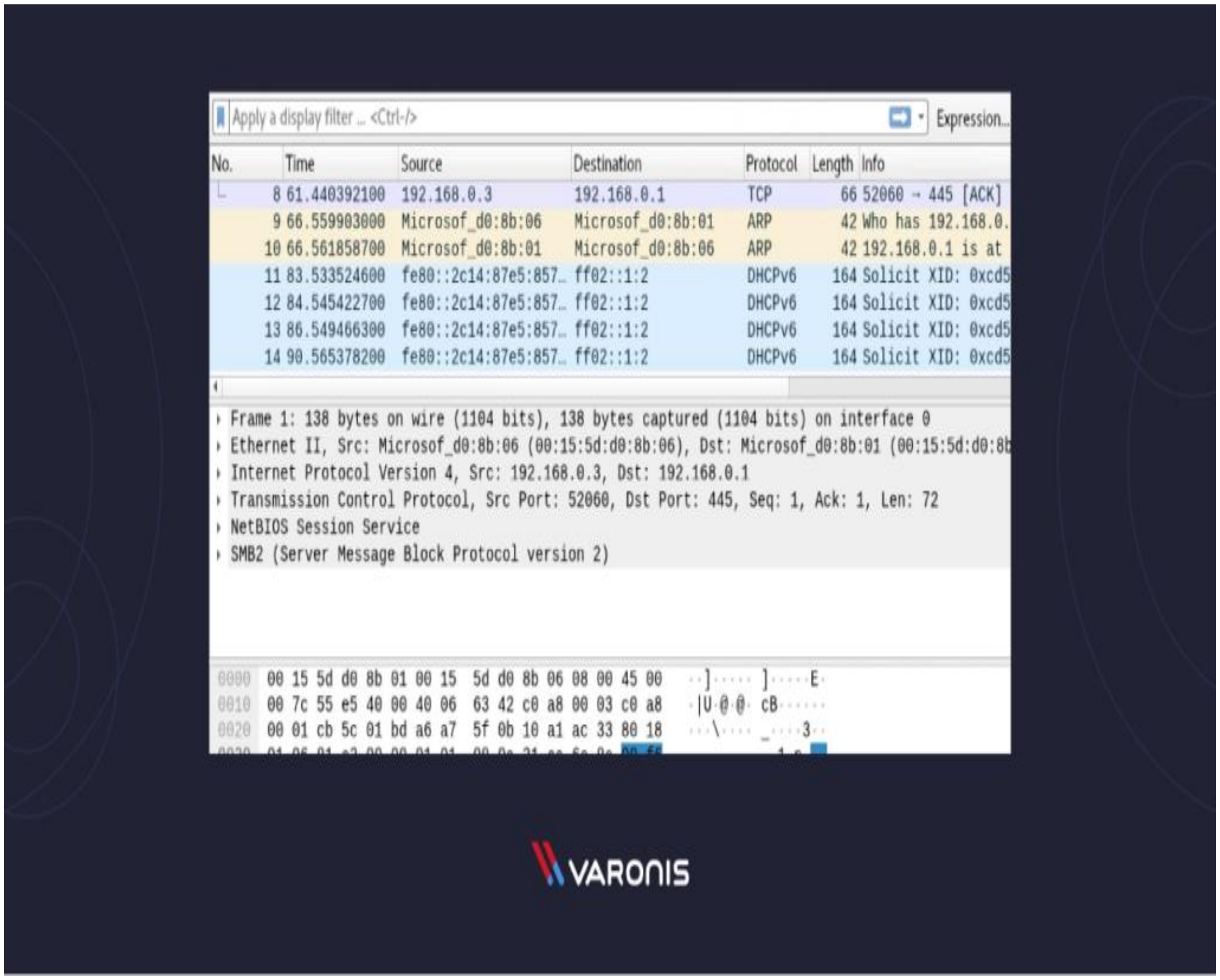


You can select the menu item Capture -> Start.



Or you could use the keystroke Control – E.

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



```
No.   Time          Source                 Destination         Protocol Length Info
   8 61.440392100  192.168.0.3            192.168.0.1         TCP        66 52060 → 445 [ACK]
   9 66.559903000  Microsof_d0:8b:06      Microsof_d0:8b:01   ARP        42 Who has 192.168.0.
  10 66.561858700  Microsof_d0:8b:01      Microsof_d0:8b:06   ARP        42 192.168.0.1 is at
  11 83.533524600  fe80::2c14:87e5:857…   ff02::1:2           DHCPv6    164 Solicit XID: 0xcd5
  12 84.545422700  fe80::2c14:87e5:857…   ff02::1:2           DHCPv6    164 Solicit XID: 0xcd5
  13 86.549466300  fe80::2c14:87e5:857…   ff02::1:2           DHCPv6    164 Solicit XID: 0xcd5
  14 90.565378200  fe80::2c14:87e5:857…   ff02::1:2           DHCPv6    164 Solicit XID: 0xcd5
```

```
› Frame 1: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface 0
› Ethernet II, Src: Microsof_d0:8b:06 (00:15:5d:d0:8b:06), Dst: Microsof_d0:8b:01 (00:15:5d:d0:8b
› Internet Protocol Version 4, Src: 192.168.0.3, Dst: 192.168.0.1
› Transmission Control Protocol, Src Port: 52060, Dst Port: 445, Seq: 1, Ack: 1, Len: 72
› NetBIOS Session Service
› SMB2 (Server Message Block Protocol version 2)
```

```
0000  00 15 5d d0 8b 01 00 15  5d d0 8b 06 08 00 45 00   ··]····· ]·····E·
0010  00 7c 55 e5 40 00 40 06  63 42 c0 a8 00 03 c0 a8   ·|U·@·@· cB······
0020  00 01 cb 5c 01 bd a6 a7  5f 0b 10 a1 ac 33 80 18   ···\···· _····3··
```

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

EXPERIMENT NO: 11