

Aim: Study of following Network Devices in Detail and Connect the computers in Local Area Network.

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

Apparatus (Software): No software or hardware needed.

Procedure: Following should be done to understand this practical.

1. **Repeater:** Functioning at Physical Layer. A **repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be use to connect for more than two devices
2. **Hub:** An **Ethernet hub, active hub, network hub, repeater hub, hub** or **concentrator** is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.
3. **Switch:** A **network switch** or **switching hub** is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.
4. **Bridge:** A **network bridge** connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term *bridge* formally means a device that behaves according to the IEEE 802.1D standard. A bridge and switch are very much alike; a switch being a bridge with numerous ports. *Switch* or *Layer 2 switch* is often used interchangeably with *bridge*. Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.
5. **Router:** A **router** is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.
6. **Gate Way:** In a communications network, a network node equipped for interfacing with

Another network that uses different protocols.

- A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.
- A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.

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

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

When Internet Connection Sharing is enabled, your LAN adapter will be set to use IP address 192.168.0.1. Your computer may lose connectivity with other computers on your network. If these other computers have static IP addresses, it is a good idea to set them to obtain their IP addresses automatically. Are you sure you want to enable Internet Connection Sharing?

11. Click **Yes**.

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

The network adapter that is connected to the LAN is configured with a static IP address of 192.168.0.1 and a subnet mask of 255.255.255.0

On the client computer

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

1. Log on to the client computer as Administrator or as Owner.
2. Click **Start**, and then click **Control Panel**.
3. Click **Network and Internet Connections**.

4. Click **Network Connections**.
5. Right-click **Local Area Connection** and then click **Properties**.
6. Click the **General** tab, click **Internet Protocol (TCP/IP)** in the **connection uses the following items** list, and then click **Properties**.
7. In the **Internet Protocol (TCP/IP) Properties** dialog box, click **Obtain an IP address automatically** (if it is not already selected), and then click **OK**.

Note: You can also assign a unique static IP address in the range of 192.168.0.2 to 192.168.0.254. For example, you can assign the following static IP address, subnet mask, and default gateway:

8. IP Address 192.168.31.202
9. Subnet mask 255.255.255.0
10. Default gateway 192.168.31.1
11. In the **Local Area Connection Properties** dialog box, click **OK**.
12. Quit Control Panel.

1. AIM: WRITE A PROGRAM TO IMPLEMENT THE DATA LINK FRAMING METHODS SUCH AS CHARACTER STUFFING AND BIT STUFFING

BIT STUFFING ALGORITHM:

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 and 1's) i.e.

Step 3: initialize $I = 0$;

Step 4: read $a[i]$ and increment i Step 5: Initialize $i=0, j=0, \text{count} = 0$

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 $\text{count} < 5$) Step 10: Initialize $k=i+1$;

Step 11: Increment j and $b[j] = a[k]$; Step 12: Increment count ;

Step 13: if $\text{count} = 5$ 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

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: 8
Enter input frame(0's & 1's only): 1
                                   1
                                   1
                                   1
                                   1
                                   1
                                   1
                                   0
```

After stuffing the frame is: 111110110

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: 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

PROGRAM :

```
#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: saikumar
Enter position: 3
Enter the character: r
Frame after stuffing: dlestxsadlerdleikumardleetx

```

2. WRITE A PROGRAM TO IMPLEMENT DATA LINK LAYER FARMING METHOD CHECKSUM

PROGRAM:

```

#include<stdio.h>
#include<math.h>

int sender(int arr[10],int n)
{
    int checksum,sum=0,i;
    printf("\n***SENDER SIDE*\n");
    for(i=0;i<n;i++)

```

```

    sum+=arr[i];
    printf("SUM IS: %d",sum);
    checksum=~sum;    //1's complement of sum
    printf("\nCHECKSUM IS:%d",checksum);
    return checksum;
}

void receiver(int arr[10],int n,int sch)
{
    int checksum,sum=0,i;
    printf("\n\n***RECEIVER SIDE*\n");
    for(i=0;i<n;i++)
        sum+=arr[i];
    printf("SUM IS:%d",sum);
    sum=sum+sch;
    checksum=~sum;    //1's complement of sum
    printf("\nCHECKSUM IS:%d",checksum);
}

void main()
{
    int n,sch,rch;
    printf("\nENTER SIZE OF THE STRING:");
    scanf("%d",&n);
    int arr[n];
    printf("ENTER THE ELEMENTS OF THE ARRAY TO CALCULATE CHECKSUM:\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
    sch=sender(arr,n);
    receiver(arr,n,sch);
}

```

Output:

```

Enter size of the string: 2
Enter the elements of the array to calculate checksum: 4
                                                    6

SENDER SIDE
SUM IS : 10
CHECKSUM IS : -11

RECEIVER SIDE
SUM IS : 10
CHECKSUM IS : 0

```

4.WRITE A PROGRAM FOR HAMMING CODE GENERATION FOR ERROR DETECTION AND CORRECTION.

PROGRAM:

```

#include<stdio.h>
void main()

```

```

{
    int data[10];
    int dataatrec[10],c,c1,c2,c3,i;

    printf("Enter 4 bits of data one by one\n");
    scanf("%d",&data[0]);
    scanf("%d",&data[1]);
    scanf("%d",&data[2]);
    scanf("%d",&data[4]);

    //Calculation of even parity
    data[6]=data[0]^data[2]^data[4];
    data[5]=data[0]^data[1]^data[4];
    data[3]=data[0]^data[1]^data[2];

    printf("\nEncoded data is\n");
    for(i=0;i<7;i++)
        printf("%d",data[i]);

    printf("\n\nEnter received data bits one by one\n");
    for(i=0;i<7;i++)
        scanf("%d",&dataatrec[i]);

    c1=dataatrec[6]^dataatrec[4]^dataatrec[2]^dataatrec[0];
    c2=dataatrec[5]^dataatrec[4]^dataatrec[1]^dataatrec[0];
    c3=dataatrec[3]^dataatrec[2]^dataatrec[1]^dataatrec[0];
    c=c3*4+c2*2+c1 ;

    if(c==0) {
        printf("\nNo error while transmission of data\n");
    }
    else {
        printf("\nError on position %d",c);

        printf("\nData sent : ");
        for(i=0;i<7;i++)
            printf("%d",data[i]);

        printf("\nData received : ");
        for(i=0;i<7;i++)
            printf("%d",dataatrec[i]);
        printf("\nCorrect message is\n");

        //if errorneous bit is 0 we complement it else vice versa

        if(dataatrec[7-c]==0)
            dataatrec[7-c]=1;
        else
            dataatrec[7-c]=0;
        for (i=0;i<7;i++) {
            printf("%d",dataatrec[i]);
        }
    }
}

```


}

Output:

Enter 4 bits of data one by one

1
0
1
0

Encoded data is

1010010

Enter received data bits one by one

1
0
1
0
0
1
0

No error while transmission of data

5.AIM: IMPLEMENT ON A DATA SET OF CHARACTERS THE THREE CRC POLYNOMIALS-CRC-12,CRC-16,CRC-CCIP

ALGORITHM

Begin

Step 1: Declare I, j, fr[8], dupfr[11], recfr[11], tlen, flag, 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 $i < \text{genl}$ Step 8: $\text{rem}[i] = \text{fr}[j]$ exponential $\text{gen}[i]$ Step 9: increment I and j
 Step 10: initialize $I = 0$
 Step 11: repeat step(12 to 13) until $I < \text{genl}$ Step 12: $\text{fr}[k1] = \text{rem}[i]$
 Step 13: increment $k1$ and i Step 14: end if
 End

PROGRAM:

```

#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 1 1 1 1 1 1

Enter generator : 1 1 0 1

Frame received correctly

6.WRITE A PROGRAM TO IMPLEMENT SLIDING WINDOW PROTOCOL FOR GOBACK N.

PROGRAM:

```

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

```

Output:

enter frame size

8

Frame 0 has been transmitted.
Frame 1 has been transmitted.
Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

2

Frame 2 has been transmitted.
Frame 3 has been transmitted.
Frame 4 has been transmitted.
Frame 5 has been transmitted.
Frame 6 has been transmitted.
Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

8

7.WRITE A PROGRAM TO IMPLEMENT SLIDING WINDOW PROTOCOL FOR SELECTIVE REPEAT.

```
#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

8.WRITE A PROGRAM TO IMPLEMENT STOP & WAIT PROTOCOL.

PROGRAM:

```

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

```

Output:-

enter window size

8

Frame 0 has been transmitted.

Frame 1 has been transmitted.

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted.

Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

2

Frame 2 has been transmitted.

Frame 3 has been transmitted.

Frame 4 has been transmitted.

Frame 5 has been transmitted.

Frame 6 has been transmitted.

Frame 7 has been transmitted.

Please enter the last Acknowledgement received.

8

9.WRITE A PROGRAM FOR CONGESTION CONTROL USING LEAKY BUCKET ALGORITHM.

PROGRAM:

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

```

printf("Buffer size= %d out of bucket size=%d\n",
storage, bucket_size);
storage -= output_pkt_size;
}
return 0;
}

```

Output:

```

Buffer size= 4 out of bucket size=10
Buffer size= 7 out of bucket size=10
Buffer size= 10 out of bucket size=10
Packet loss = 4
Buffer size= 9 out of bucket size=10

```

10.AIM: IMPLEMENT DIJKSTRA'S ALGORITHM TO COMPUTE THE SHORTESTPATH IN A GRAPH

PROGRAM:

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int path[10][10],i,cost[10][10],minc,tc[10],N,np,j,stn,en,pindex,k;
    clrscr();
    printf("Enter the no.of nodes (N) =");
    scanf("%d",&N);
    printf("Enter the all possible paths into (paths(np)=");
    scanf("%d",&np);
    printf("Enter the all possible paths into (path[i][j])=\n");

    for(i=1;i<=np;i++)
    {
        for(j=1;j<=N;j++)
        {
            scanf("%d",&path[i][j]);
        }
    }
    printf("Enter the costs (distance) between nodes into (cost[i][j])=\n");

    for(i=1;i<=np;i++)
    {
        for(j=1;j<=N;j++)
        {

```

```

        scanf("%d",&cost[i][j]);
    }
}
for(i=1;i<=np;i++)
{
    stn=1;
    tc[i]=0;

    for(j=1;j<N;j++)
    {
        en=path[i][j+1];
        tc[i]=tc[i]+cost[stn][en];
        if(en==N)
            break;
        else
            stn=en;
    }
}

minc=tc[1];
pindex=1;

for(i=1;i<np;i++)
{
    if(tc[i]<minc);
    {
        minc=tc[i];
        pindex=i;
    }
}
for(k=1;k<=N;k++)
{
    printf("%d-->",path[pindex][k]);
}

printf("\n minimum cost =%d",minc);

getche();
}

```

Output


```

TC.EXE
Enter the no.of nodes (N) =5
Enter the all possible paths into (paths(np)=4
Enter the all possible paths into (path[i][j])=
1 2 4 5 0
1 3 4 5 0
1 2 5 0 0
1 4 5 0 0
Enter the costs (distance) between nodes into (cost[i][j])=
0 1 4 2 0
0 0 0 2 3
0 0 0 3 0
0 0 0 0 5
1-->2-->5-->0-->0-->
minimum cost =4

```

11. Now obtain Routing table for each node using distance vector routing algorithm

Algorithm:

Input: Graph and a given vertex src

Output: Shortest distance to all vertices from src. If there is a negative weight cycle, then shortest distances are not calculated, negative weight cycle is reported.

1) This step initializes distances from source to all vertices as infinite and distance to source itself as 0. Create an array dist[] of size |V| with all values as infinite except dist[src] where src is source vertex.

2) This step calculates shortest distances. Do following |V|-1 times where |V| is the number of vertices in given graph.

.....a) Do following for each edge v-u

.....If dist[v] > dist[u] + weight of edge uv, then update dist[v]

.....dist[v] = dist[u] + weight of edge uv

3) This step reports if there is a negative weight cycle in graph. Do following for each edge u-v

.....If dist[v] > dist[u] + weight of edge uv, then "Graph contains negative weight cycle"

The idea of step 3 is, step 2 guarantees shortest distances if graph doesn't contain negative weight cycle. If we iterate through all edges one more time and get a shorter path for any vertex, then there is a negative weight cycle

PROGRAM:

```

#include<stdio.h>
#include<math.h>
#include<conio.h> main()
{
int i,j,k,nv,sn,noadj,edel[20],tdel[20][20],min;
char sv,adver[20],ch;
clrscr();
printf("\n ENTER THE NO.OF VERTECES:");
scanf("%d",&nv);
printf("\n ENTER THE SOURCE VERTEX
NUM,BER AND NAME:");
scanf("%d",&sn);
flushall(); sv=getchar();
printf("\n NETER NO.OF ADJ VERTECES TO
VERTEX %c",sv);
scanf("%d",&noadj);

```

```

for(i=0;i<noadj;i++)
{
printf("\n ENTER TIME DELAY and NODE NAME:");
scanf("%d %c",&edel[i],&adver[i]);
}
for(i=0;i<noadj;i++)
{
printf("\n ENTER THE TIME DELAY FROM %c to ALL OTHER
NODES: ",adver[i]);
for(j=0;j<nv;j++)
scanf("%d",&tadel[i][j]);
}
printf("\n DELAY VIA--VERTEX \n ");
for(i=0;i<nv;i++)
{
min=1000; ch=0;
for(j=0;j<noadj;j++)
if(min>(tadel[j][i]+edel[j]))
{
min=tadel[j][i]+edel[j];
ch=adver[j];
}
if(i!=sn-1)
printf("\n%d %c",min,ch);
else
printf("\n0 -");
}
getch();
}

```

INPUT/OUTPUT:

```

ENTER THE NO.OF VERTECES:12
ENTER THE SOURCE VERTEX NUMBER AND NAME:10 J
ENTER NO.OF ADJ VERTECES TO
VERTEX 4 ENTER TIME DELAY and NODE
NAME:8 A ENTER TIME DELAY and NODE
NAME:10 I ENTER TIME DELAY and NODE
NAME:12 H ENTER TIME DELAY and
NODE NAME:6 K
ENTER THE TIME DELAY FROM A to ALL OTHER
NODES: 0 12 25 40 14 23 18 17 21 9 24 29
ENTER THE TIME DELAY FROM I to ALL OTHER
NODES: 24 36 18 27 7 20 31 20 0 11 22 33
ENTER THE TIME DELAY FROM H to ALL OTHER
NODES: 20 31 19 8 30 19 6 0 14 7 22 9
ENTER THE TIME DELAY FROM K to ALL OTHER
NODES: 21 28 36 24 22 40 31 19 22 10 0 9
DELAY VIA--VERTEX
8 a
20 a
28 i
20 h
17 i

```

30 i
18 h
12 h
10 i
0 -
6 k
15 k

12.write a program to implement broadcast tree by taking subnet of hosts.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
int edge[10][10],vnodes[10],s,nextn[10],k=-1,l=-1;
int i,j,n;
void getdata();
void operation();
void main()
{
int ch;
clrscr();
printf("\n\n 1.graph\n 2.exit\n");
printf("\n enter your choice=\n");
scanf("%d",&ch);
switch(ch)
{
    case 1: getdata();
            operation();
            break;
    case 2:exit(0);
}
getch();
}
void getdata()
{
    printf("\n enter the no of nodes exists in graph=\n");
    scanf("%d",&n);
    printf("\n Enter Edges(distance) matrix between each node=");
```

```

        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                scanf("%d",&edge[i][j]);
            }
        }
    }
void operation()
{
    for(i=1;i<=n;i++)
    {
        vnodes[i]=0;
    }
    printf("\n enter the starting node of your graph=/n");
    scanf("%d",&s);
    i=s;
    vnodes[i]=1;
    do
    {
        printf("\n%d-->",i);
        for(j=1;j<=n;j++)
        {
            if(vnodes[j]==0 && edge[i][j]==1)
            {
                printf("%d,",j);
                nextn[++k]=j;
                vnodes[j]=1;
            }
        }
        l++;
        i=nextn[l];
    }
    while((k+1)!=l);
}

```

Output:

- 1.graph
- 2.exit

Enter your choice = 1

Enter the no.of nodes exists in graph = 4

Enter edges(distance) matrix between each node :

```
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0
```

Enter the starting node of your graph = 1

```
1→2,4
2→3
4→
3→
```

13. WIRESHARK

(i)PACKET CAPTURE USING WIRE SHARK

(ii)STARTING WIRESHARK

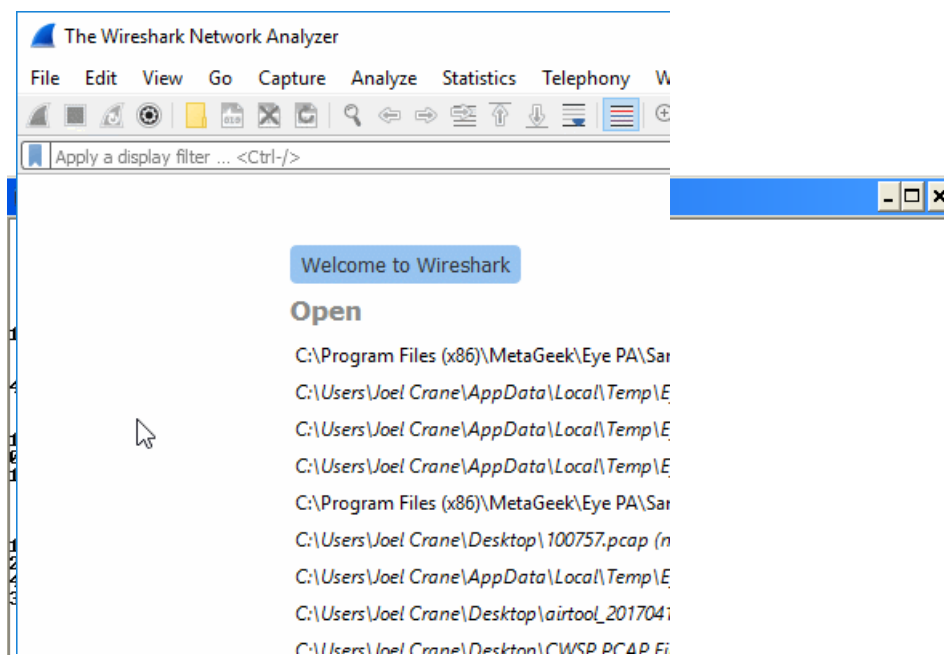
(iii)VIEWING CAPTURED TRAFFIC

(iv)ANALYSIS AND STATISTICS & FILTERS.

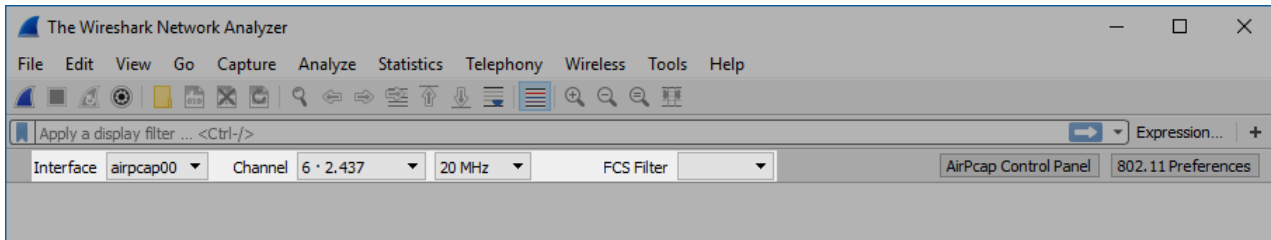
(i)PACKET CAPTURE USING WIRE SHARK

Set up the Packet Capture

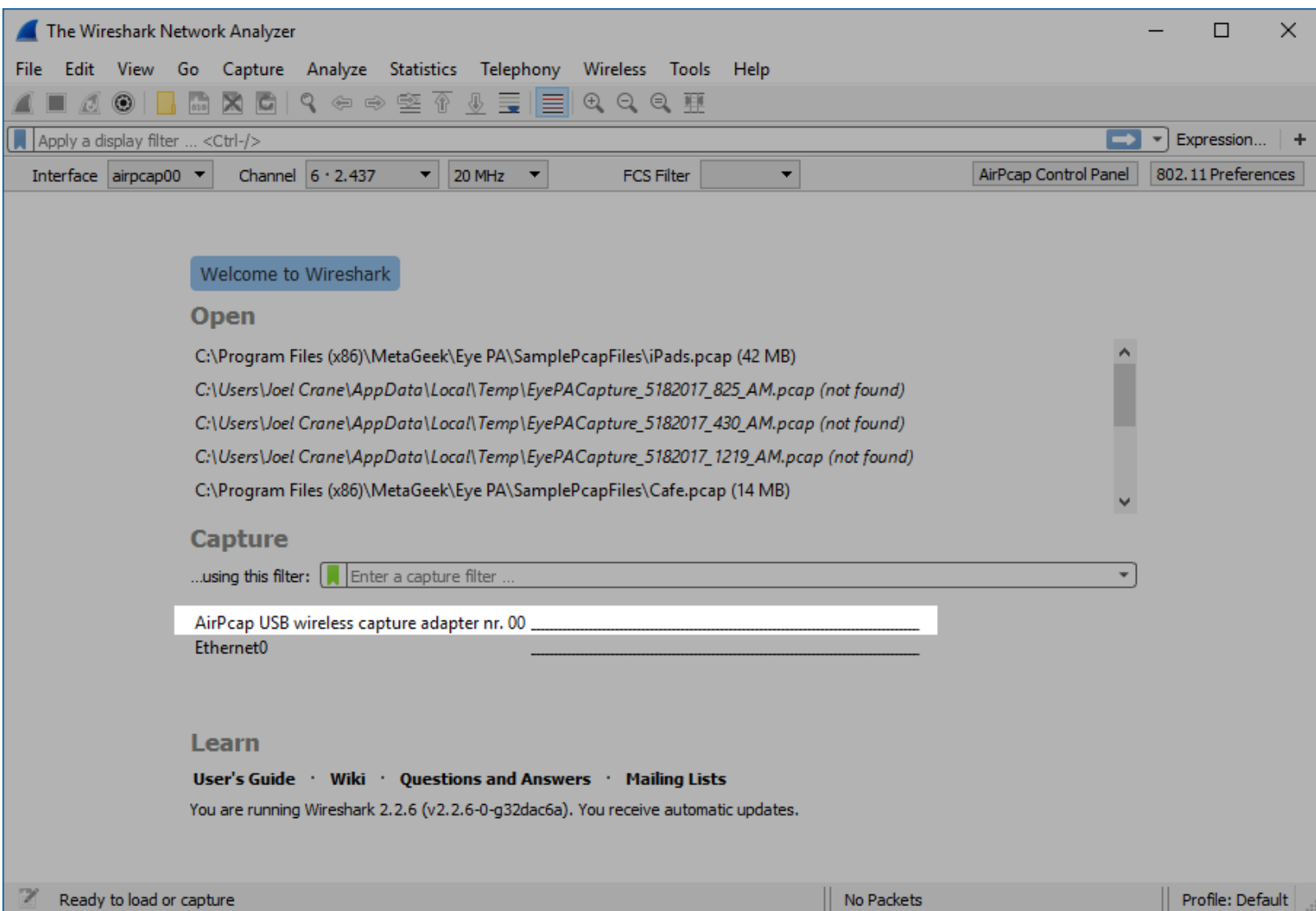
1. Click **View > Wireless Toolbar**. The Wireless Toolbar will appear just below the Main toolbar.



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



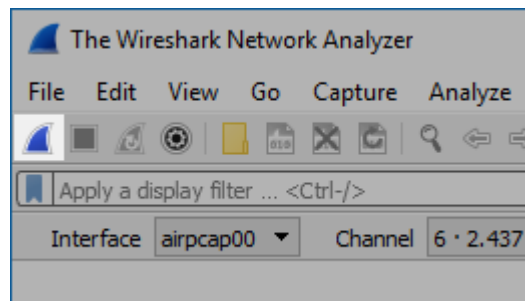
3. Under **Capture**, click on **AirPcap USB wireless capture adapter** to select the capture interface.



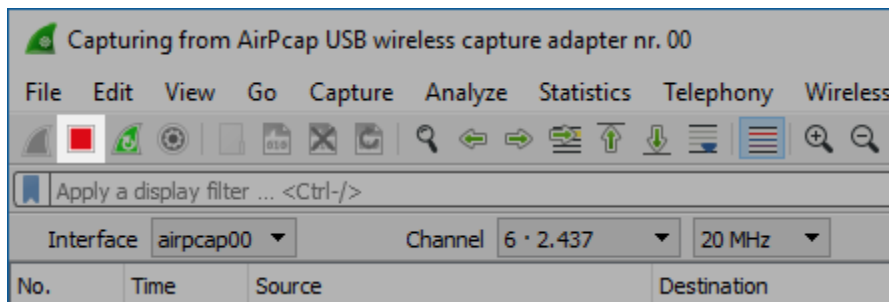
*Note: If the AirPcap isn't listed, press **F5** to refresh the list of available packet capture interfaces.*

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

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

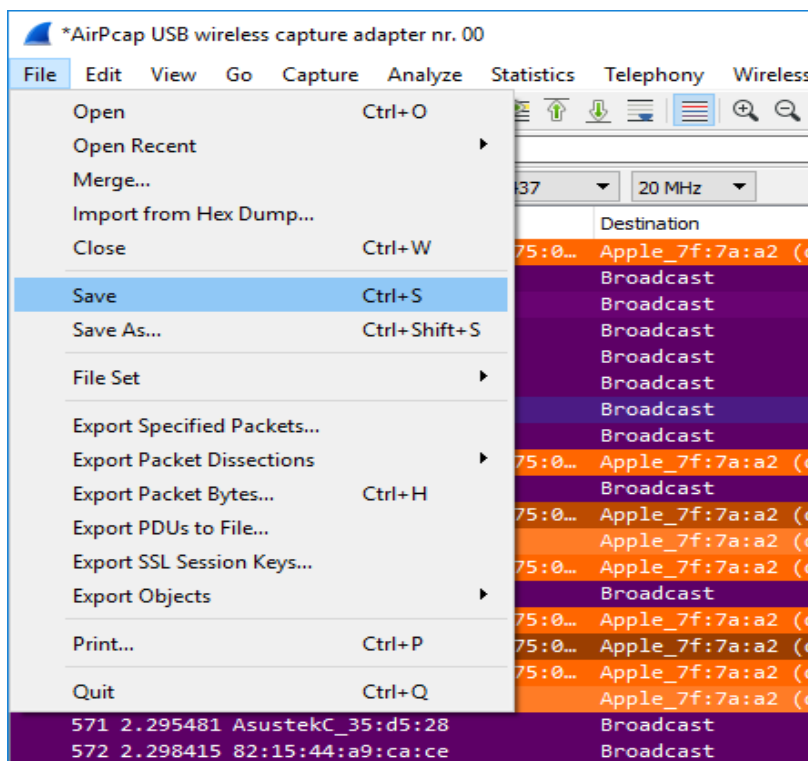


5. When you are finished capturing, click the **Stop** button.



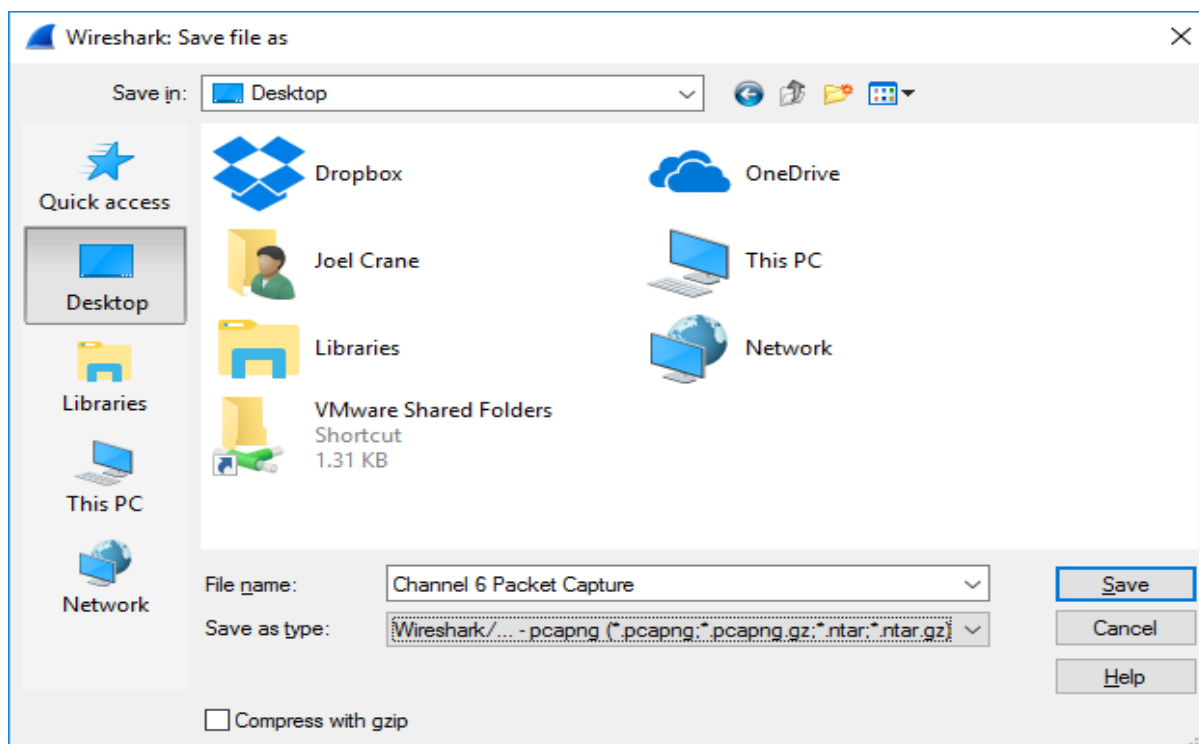
Saving the Capture

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.

(ii) STARTING WIRESHARK

Two different methods for starting Wireshark are available. These include the Start menu and the Run command box.

Method 1 - Start Menu[\[edit\]](#) [\[edit source\]](#)

To start Wireshark using the Start menu:

1. Open the **Start** menu.
2. Select **All Programs**.
3. Select **Wireshark**.

Method 2 - Run Command[\[edit\]](#) [\[edit source\]](#)

To start Wireshark using the Run command box:

1. Open the **Start** menu or press the **Windows key + R**.
2. Type **Wireshark** in the Run command box.
3. Press **Enter**.

Activity 2 - Open the Capture Interfaces Dialog Box[\[edit\]](#) [\[edit source\]](#)

Three different methods for opening the Capture Interfaces dialog box are available. These include the Capture menu, the Capture Interfaces toolbar button, and the Capture Interfaces keyboard shortcut.

Method 1 - Capture Menu[\[edit\]](#) [\[edit source\]](#)

To open the Capture Interfaces dialog box using the Capture menu:

1. Select the **Capture** menu.
2. Select **Interfaces**.

Method 2 - Capture Interfaces Toolbar Button[\[edit\]](#) [\[edit source\]](#)

To open the Capture Interfaces dialog box using the Capture interfaces Toolbar button:

1. Locate the toolbar button with the help text **List the available capture interfaces**. This should be the first toolbar button on the left.
2. Click the Capture Interfaces toolbar button.

Method 3 - Capture Interfaces Keyboard Shortcut[\[edit\]](#) | [edit source](#)

To open the Capture Interfaces dialog box using the Capture interfaces keyboard shortcut:

1. Press **<Ctrl> + I**.

Activity 3 - Start a Wireshark Capture[\[edit\]](#) | [edit source](#)

To start a Wireshark capture from the Capture Interfaces dialog box:

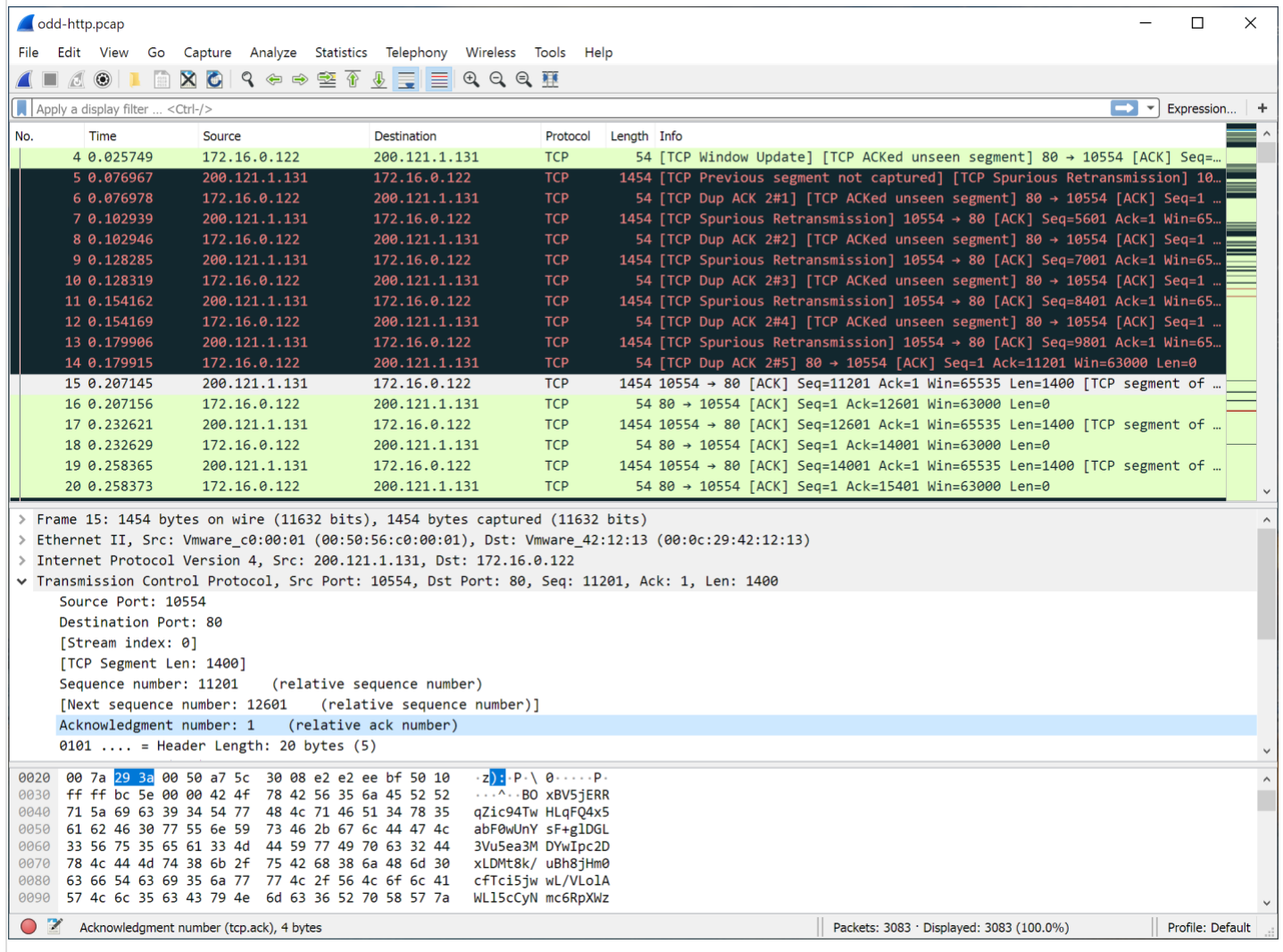
1. Observe the available interfaces. If you have multiple interfaces displayed, look for the interface with the highest packet count. This is your most active network interface.
2. Select the interface you want to use for the capture using the check box on the left.
3. Select **Start** to begin the capture.

(iii)VIEWING CAPTURED TRAFFIC

Once you have captured some packets or you have opened a previously saved capture file, you can view the packets that are displayed in the packet list pane by simply clicking on a packet in the packet list pane, which will bring up the selected packet in the tree view and byte view panes.

You can then expand any part of the tree to view detailed information about each protocol in each packet. Clicking on an item in the tree will highlight the corresponding bytes in the byte view. An example with a TCP packet selected is shown in [Figure 6.1, “Wireshark with a TCP packet selected for viewing”](#). It also has the Acknowledgment number in the TCP header selected, which shows up in the byte view as the selected bytes.

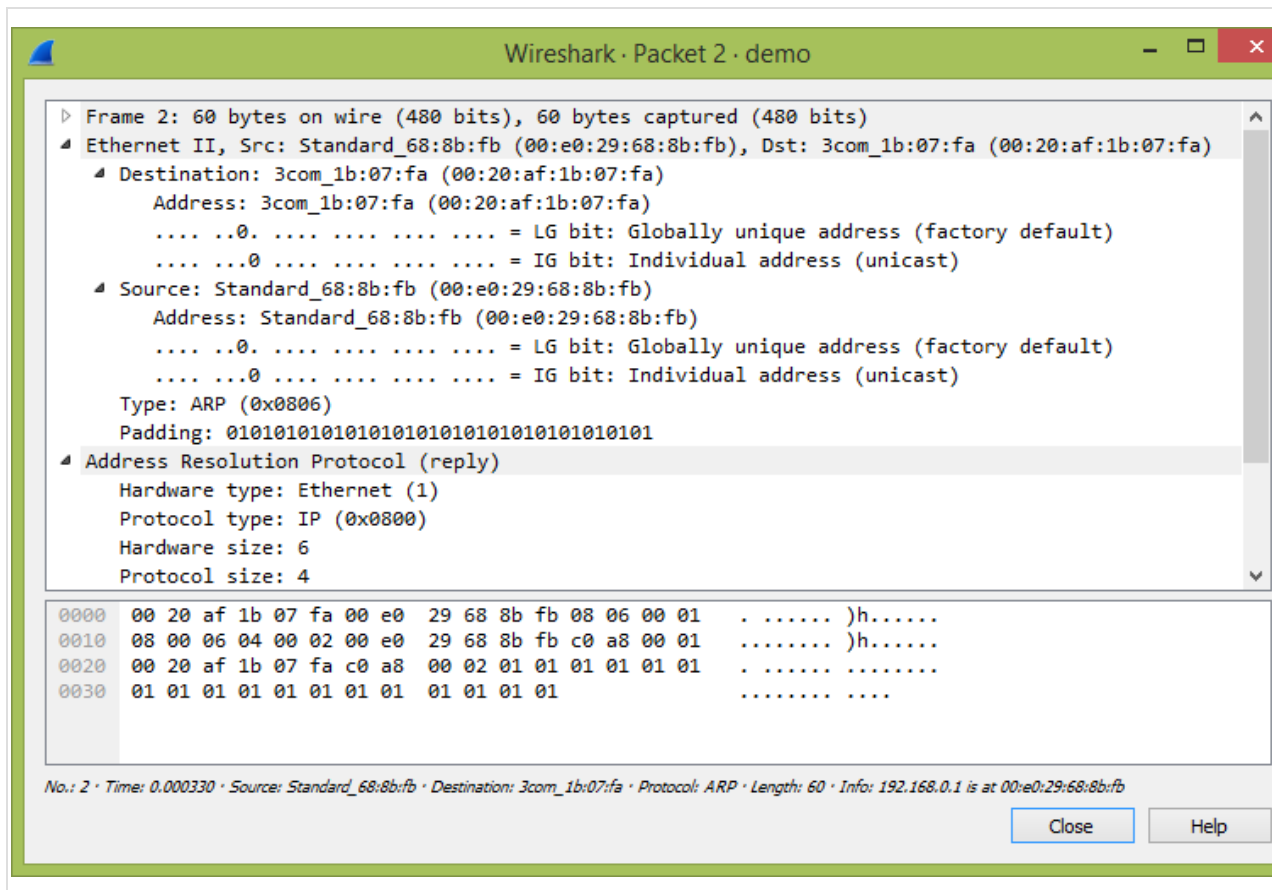
Figure 6.1. Wireshark with a TCP packet selected for viewing



You can also select and view packets the same way while Wireshark is capturing if you selected “Update list of packets in real time” in the “Capture Preferences” dialog box.

In addition you can view individual packets in a separate window as shown in [Figure 6.2, “Viewing a packet in a separate window”](#). You can do this by double-clicking on an item in the packet list or by selecting the packet in which you are interested in the packet list pane and selecting View → Show Packet in New Window. This allows you to easily compare two or more packets, even across multiple files.

Figure 6.2. Viewing a packet in a separate window



Along with double-clicking the packet list and using the main menu there are a number of other ways to open a new packet window:

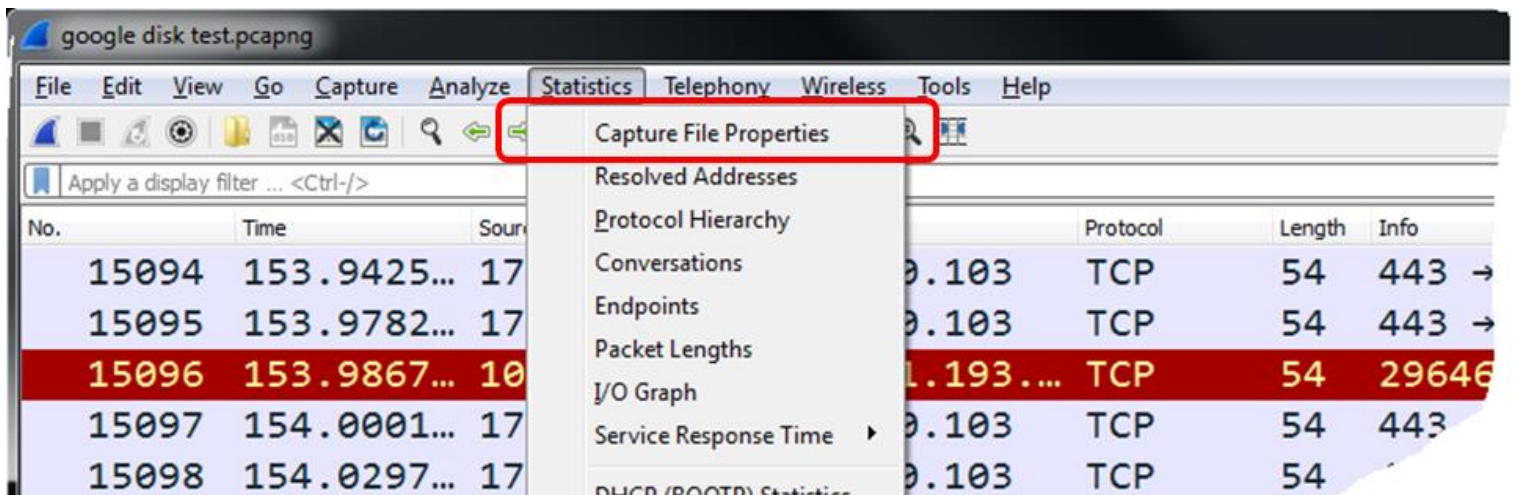
(iv) ANALYSIS AND STATISTICS & FILTERS.

Start Wireshark, click on Statistics.

How to do it...

1. From the Statistics menu, choose Capture File Properties:

What you will get is the Capture File Properties window (displayed in the following screenshot).



2. As you can see in the following screenshot, we have the following:

- File: Provides file data, such as filename and path, length, and so on
- Time: Start time, end time, and duration of capture
- Capture: Hardware information for the PC that Wireshark is installed on
- Interfaces: Interface information—the interface registry identifier on the left, if capture filter is turned on, interface type and packet size limit
- Statistics: General capture statistics, including captured and displayed packets:

The image shows the 'Wireshark - Capture File Properties - google disk test' window. The window is divided into several sections, each highlighted with a red border and a red label to its right:

- File**: Contains file information such as Name, Length, Format, and Encapsulation.
- Time**: Contains time information such as First packet, Last packet, and Elapsed time.
- Capture**: Contains hardware information such as Hardware, OS, and Application.
- Interfaces**: Contains interface information such as Interface, Dropped packets, Capture filter, Link type, and Packet size limit.
- Statistics**: Contains general capture statistics such as Measurement, Captured, Displayed, and Marked.

At the bottom of the window, there is a 'Capture file comments' section and a row of buttons: Refresh, Save Comments, Close, Copy To Clipboard, and Help.

Details				
File				
Name:	C:\Technical\Wireshark\CAP-PCAP Customers\google disk test.pcapng			
Length:	52 MB			
Format:	Wireshark/... - pcapng			
Encapsulation:	Ethernet			
Time				
First packet:	2013-08-18 17:52:47			
Last packet:	2013-08-18 18:00:50			
Elapsed:	00:08:02			
Capture				
Hardware:	Unknown			
OS:	64-bit Windows 7 Service Pack 1, build 7601			
Application:	Dumpcap 1.8.4 (SVN Rev 46250 from /trunk-1.8)			
Interfaces				
<u>Interface</u>	<u>Dropped packets</u>	<u>Capture filter</u>	<u>Link type</u>	<u>Packet size limit</u>
\Device \NPF_{55DFE1F7-0FDB-46E3-8D48- C5804C455B8A}	0 (0 %)	none	Ethernet	
Statistics				
<u>Measurement</u>	<u>Captured</u>	<u>Displayed</u>	<u>Marked</u>	
Packets	63603	63603 (100.0%)	N/A	
Time span, s	482.808	482.808	N/A	
Average pps	131.7	131.7	N/A	
Average packet size, B	794.5	794.5	N/A	
Bytes	50540636	50540636 (100.0%)	n	
Average bytes/s	104 k	104 k		
Average bits/s	837 k	837 k		
Capture file comments				
<input type="button" value="Refresh"/> <input type="button" value="Save Comments"/> <input type="button" value="Close"/> <input type="button" value="Copy To Clipboard"/> <input type="button" value="Help"/>				

How it works...

This menu simply gives a summary of the filtered data properties and the capture statistics (average packets or bytes per second) when someone wants to learn the capture statistics.

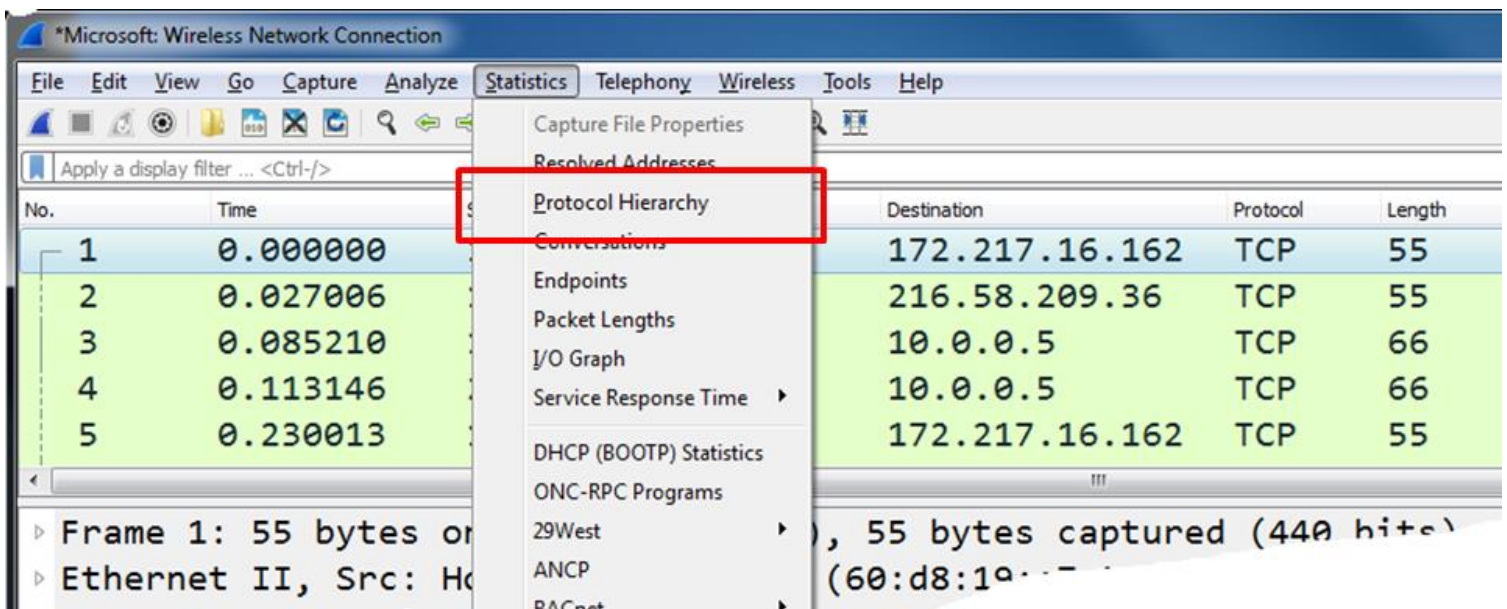
Using the statistics for protocol hierarchy menu

In this recipe, we will learn how to get protocol hierarchy information of the data that runs over the network.

Start Wireshark, click on Statistics.

How to do it...

1. From the Statistics menu, choose Protocol Hierarchy:



What you will get is data about the protocol distribution in the captured file. You will get the protocol distribution of the captured data.

2. The partial screenshot displayed here depicts the statistics of packets captured on a per-protocol basis:

Wireshark - Protocol Hierarchy Statistics - Neomim_12_04_16_1300_00001_20160412130115

Protocol	Percent Packets	Packets	Percent Bytes
Frame	100.0	280091	100.0
Ethernet	100.0	280091	100.0
TDMoP protocol	0.0	8	0.0
Logical-Link Control	0.5	1484	0.4
Link Layer Discovery Protocol	0.0	48	0.0
Internet Protocol Version 6	1.0	2725	1.1
User Datagram Protocol	0.9	2544	1.1
Service Location Protocol	0.0	12	0.0
Link-local Multicast Name Resolution	0.2	512	0.1
Hypertext Transfer Protocol	0.1	150	0.2
DHCPv6	0.7	1842	0.7
Internet Control Message Protocol v6	0.1	181	0.0
Internet Protocol Version 4	88.8	248799	94.5
User Datagram Protocol	79.3	222244	46.1
Simple Network Management Protocol	0.0	57	0.0
Service Location Protocol	0.0	72	0.0
Network Time Protocol	0.0	2	0.0
NetBIOS Name Service	0.9	2647	0.6
NetBIOS Datagram Service	0.1	196	0.1
Multicast Domain Name System	0.0	18	0.0
Link-local Multicast Name Resolution	0.2	608	0.1
Hypertext Transfer Protocol	0.7	1850	1.6
Dropbox LAN sync Discovery Protocol	0.0	94	0.0
Domain Name System	0.0	88	0.0
Data	2.6	7286	1.5
Connectionless Lightweight Directory Access Protocol	0.0	8	0.0
Check Point High Availability Protocol	74.7	209247	42.1
Bootstrap Protocol	0.0	61	0.1
ADwin configuration protocol	0.0	10	0.0
Transmission Control Protocol	9.2	25716	48.2
TCP Encapsulation of IPsec Packets	0.0	1	0.0
Encapsulating Security Payload	0.0	1	0.0
Tabular Data Stream	1.3	3502	6.9
Malformed Packet	0.1	380	0.2
Secure Sockets Layer	0.1	319	0.5
NetBIOS Session Service	0.7	2024	1.3
SMB2 (Server Message Block Protocol version 2)	0.3	837	0.8
SMB (Server Message Block Protocol)	0.4	1201	0.5
SMB Pipe Protocol	0.0	4	0.0
Lightweight Directory Access Protocol	0.0	46	0.1
Kerberos	0.0	2	0.0

No display filter.

Close Copy Help

What you will get is the Protocol Hierarchy window:

- Protocol: The protocol name
- Percent Packets: The percentage of protocol packets from the total captured packets
- Packets: The number of protocol packets from the total captured packets
- Percent Bytes: The percentage of protocol bytes from the total captured packets
- Bytes: The number of protocol bytes from the total captured packets
- Bit/s: The bandwidth of this protocol, in relation to the capture time
- End Packets: The absolute number of packets of this protocol (for the highest protocol in the decode file)

- End Bytes: The absolute number of bytes of this protocol (for the highest protocol in the decode file)
- End Bit/s: The bandwidth of this protocol, relative to the capture packets and time (for the highest protocol in the decode file)

The end columns counts when the protocol is the last protocol in the packet (that is, when the protocol comes at the end of the frame). These can be TCP packets with no payload (for example, SYN packets) which carry upper layer protocols. That is why you see a zero count for Ethernet, IPv4, and UDP end packets; there are no frames where those protocols are the last protocol in the frame.

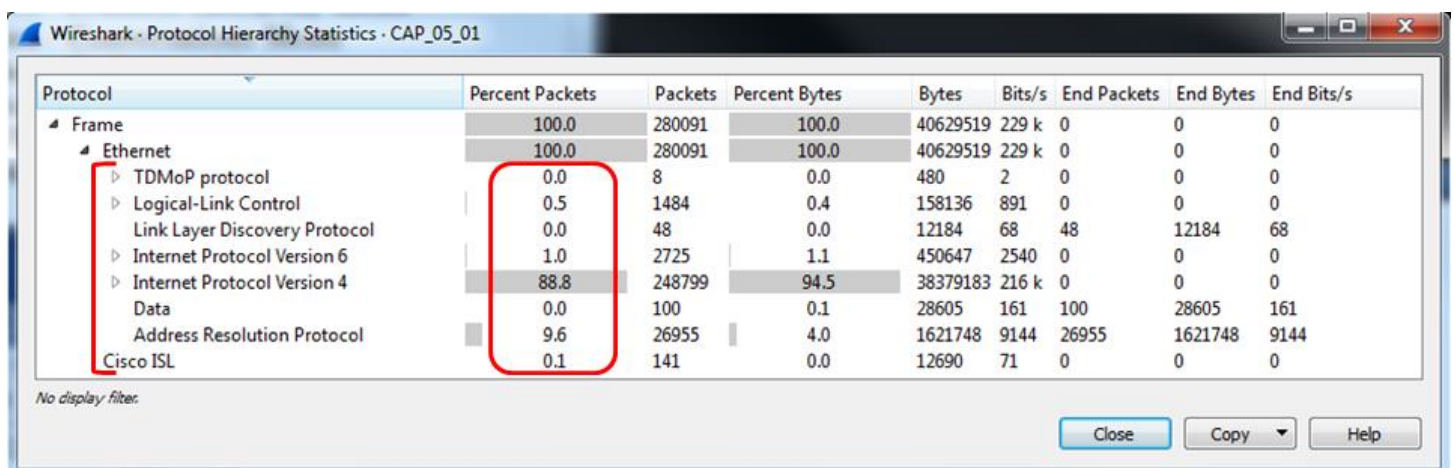
In this file example, we can see two interesting issues:

- We can see 1,842 packets of DHCPv6. If IPv6 and DHCPv6 are not required, disable it.
- We see more than 200,000 **checkpoint high availability (CPHA)** packets, 74.7% of which are sent over the network we monitored. These are synchronization packets that are sent between two firewalls working in a cluster, updating session tables between the firewalls. Such an amount of packets can severely influence performance. The solution for this problem is to configure a dedicated link between the firewalls so that session tables will not influence the network.

How it works...

Simply, it calculates statistics over the captured data. Some important things to notice:

- The percentage always refers to the same layer protocols. For example, in the following screenshot, we see that logical link control has 0.5% of the packets that run over Ethernet, IPv6 has 1.0%, IPv4 has 88.8% of the packets, ARP has 9.6% of the packets and even the old Cisco ISK has 0.1 %—a total of 100 % of the protocols over layer 2 Ethernet.
- On the other hand, we see that TCP has 75.70% of the data, and inside TCP, only 12.74% of the packets are HTTP, and that is almost it. This is because Wireshark counts only the packets with the HTTP headers. It doesn't count, for example, the ACK packets, data packets, and so on:



Wireshark - Protocol Hierarchy Statistics - CAP_05_01

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	280091	100.0	40629519	229 k	0	0	0
Ethernet	100.0	280091	100.0	40629519	229 k	0	0	0
TDMoP protocol	0.0	8	0.0	480	2	0	0	0
Logical-Link Control	0.5	1484	0.4	158136	891	0	0	0
Link Layer Discovery Protocol	0.0	48	0.0	12184	68	48	12184	68
Internet Protocol Version 6	1.0	2725	1.1	450647	2540	0	0	0
Internet Protocol Version 4	88.8	248799	94.5	38379183	216 k	0	0	0
Data	0.0	100	0.1	28605	161	100	28605	161
Address Resolution Protocol	9.6	26955	4.0	1621748	9144	26955	1621748	9144
Cisco ISL	0.1	141	0.0	12690	71	0	0	0

No display filter.

Close Copy Help

Using the statistics for conversations menu

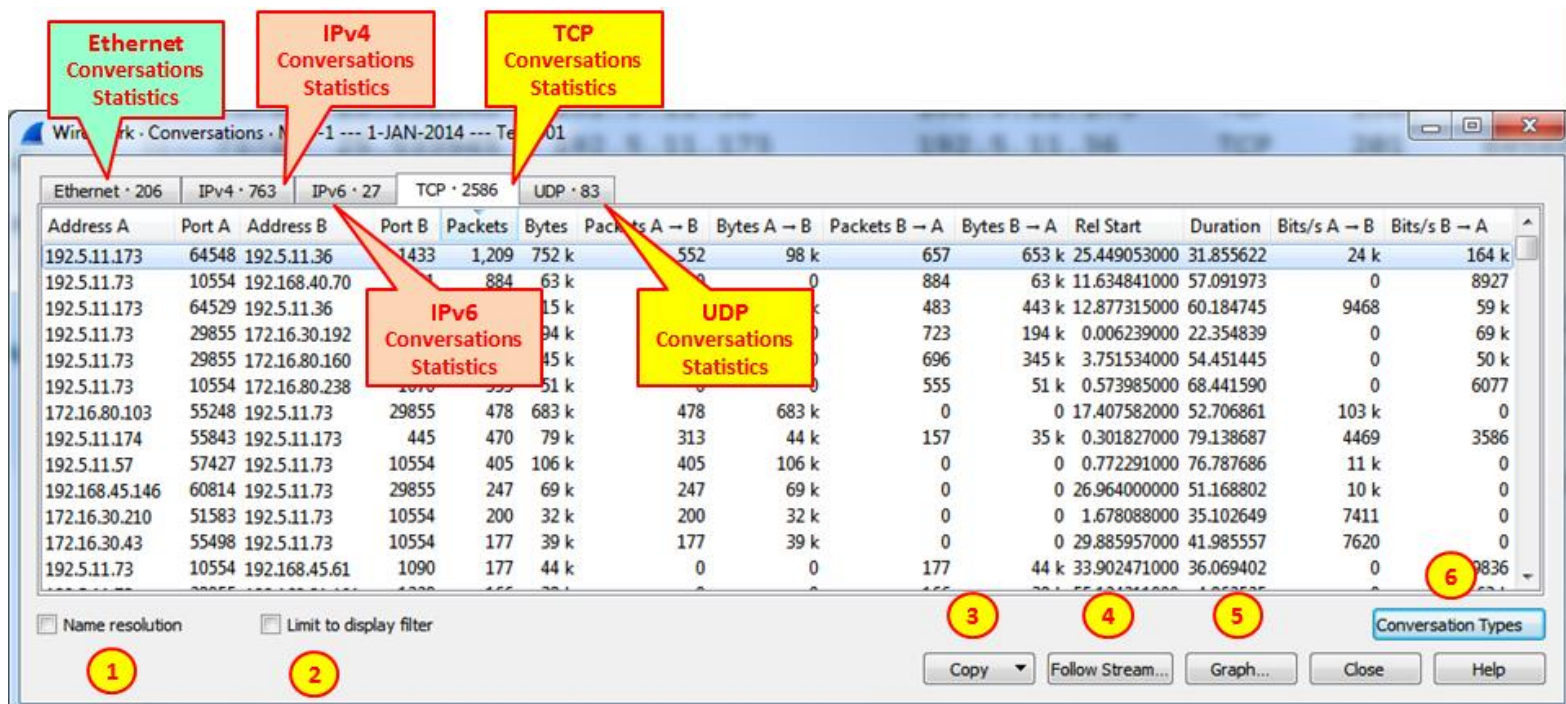
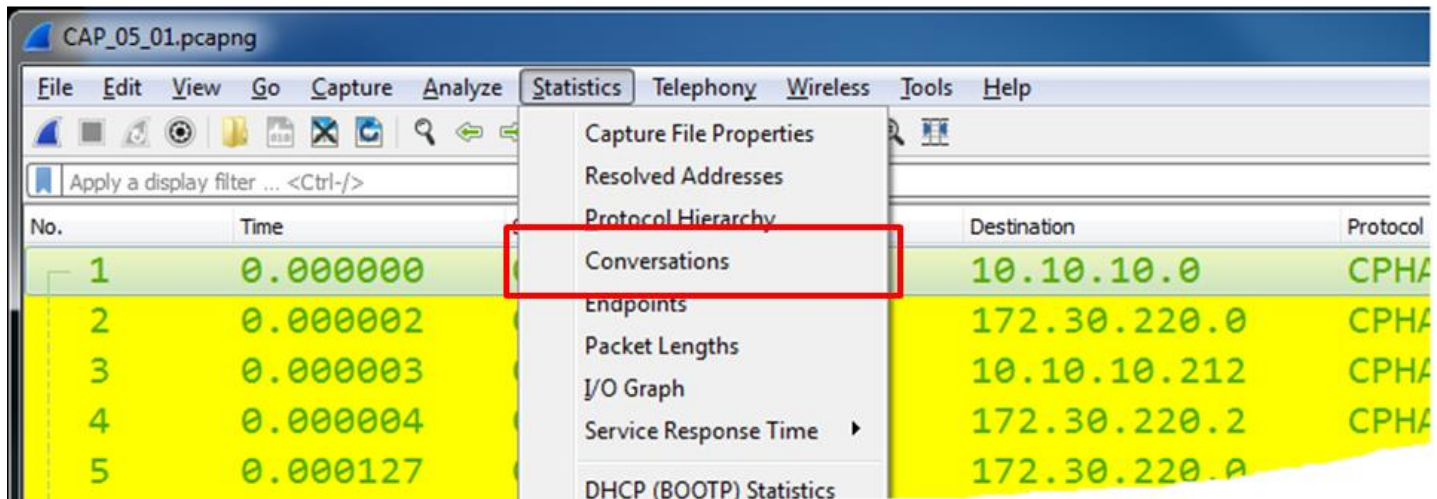
In this recipe, we will learn how to get conversation information of the data that runs over the network.

Start Wireshark, click on Statistics.

How to do it...

From the Statistics menu, choose Conversations:

The following window will come up:



You can choose between layer 2 Ethernet statistics, layer 3 IP statistics, or layer 4 TCP or UDP statistics.

You can use this statistics tools for:

- **On layer 2 (Ethernet):** To find and isolate broadcast storms
- **On layer 3/layer 4 (TCP/IP):** To connect in parallel to the internet router port, and check who is loading the line to the ISP

If you see that there is a lot of traffic going out to port 80 (HTTP) on a specific IP address on the internet, you just have to copy the address to your browser and find the website that is most popular with your users.

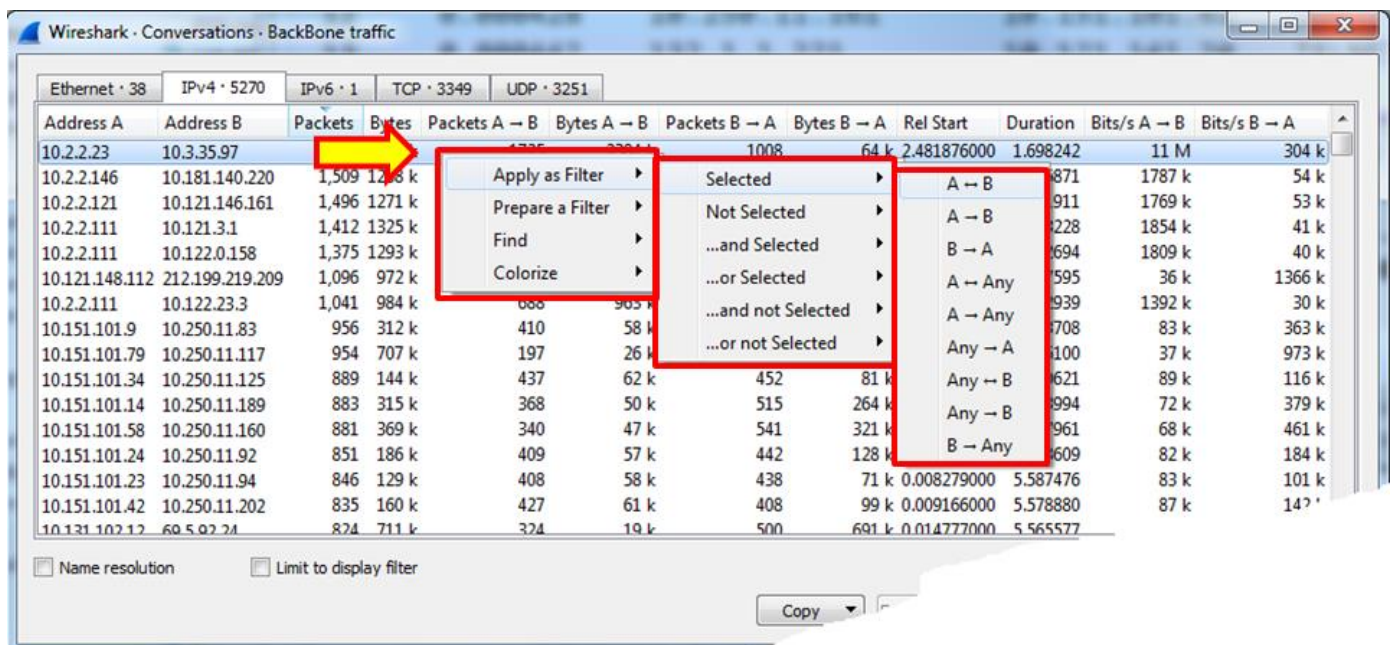
If you don't get anything, simply go to a standard DNS resolution website (search Google for DNS lookup) and find out what is loading your internet line.

For viewing IP addresses as names, you can check the Name resolution checkbox for name resolution (1 in the previous screenshot). For seeing the name resolution, you will first have to enable it by choosing View | Name Resolution | Enable for Network layer.

You can also limit the conversations statistics to a display filter by checking the Limit to display filter checkbox (2). In this way, statistics will be presented on all the packets passing the display filter.

A new feature in Wireshark version 2 is the graph feature, marked as (5) in the previous screenshot. When you choose a specific line in the TCP conversations statistics and click Graph..., it brings you to the TCP time/sequence (tcptrace) stream graph.

To copy table data, click on the **Copy** button (3). In TCP or UDP, you can mark a specific line, and then click on the Follow Stream... button (4). This will define a display filter that will show you the specific stream of data. As you can see in the following screenshot, you can also right-click a line and choose to prepare or apply a filter, or to colorize a data stream:



We also see that, unlike the previous Wireshark version, in which we saw all types of protocols in the upper tabs, here we can choose which protocols to see when only the identified protocols are presented by default.

How it works...

A network conversation is the traffic between two specific endpoints. For example, an IP conversation is all the traffic between two IP addresses, and TCP conversations present all TCP connections.

Using the statistics for endpoints menu

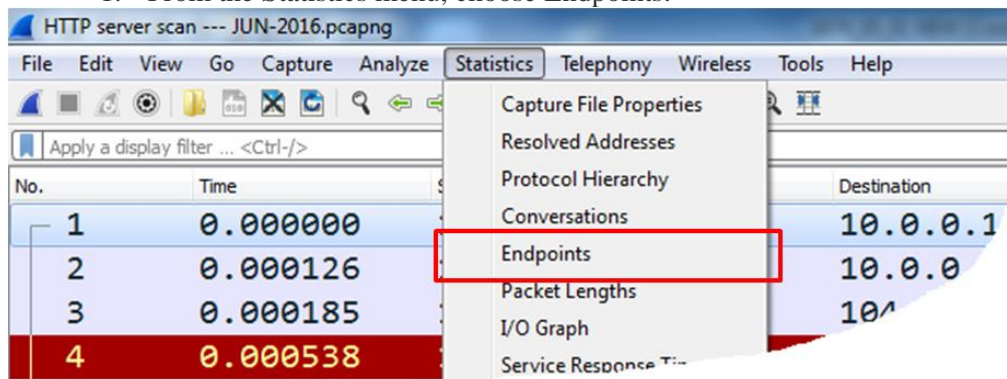
In this recipe, we will learn how to get endpoint statistics information of the captured data.

Start Wireshark and click on Statistics.

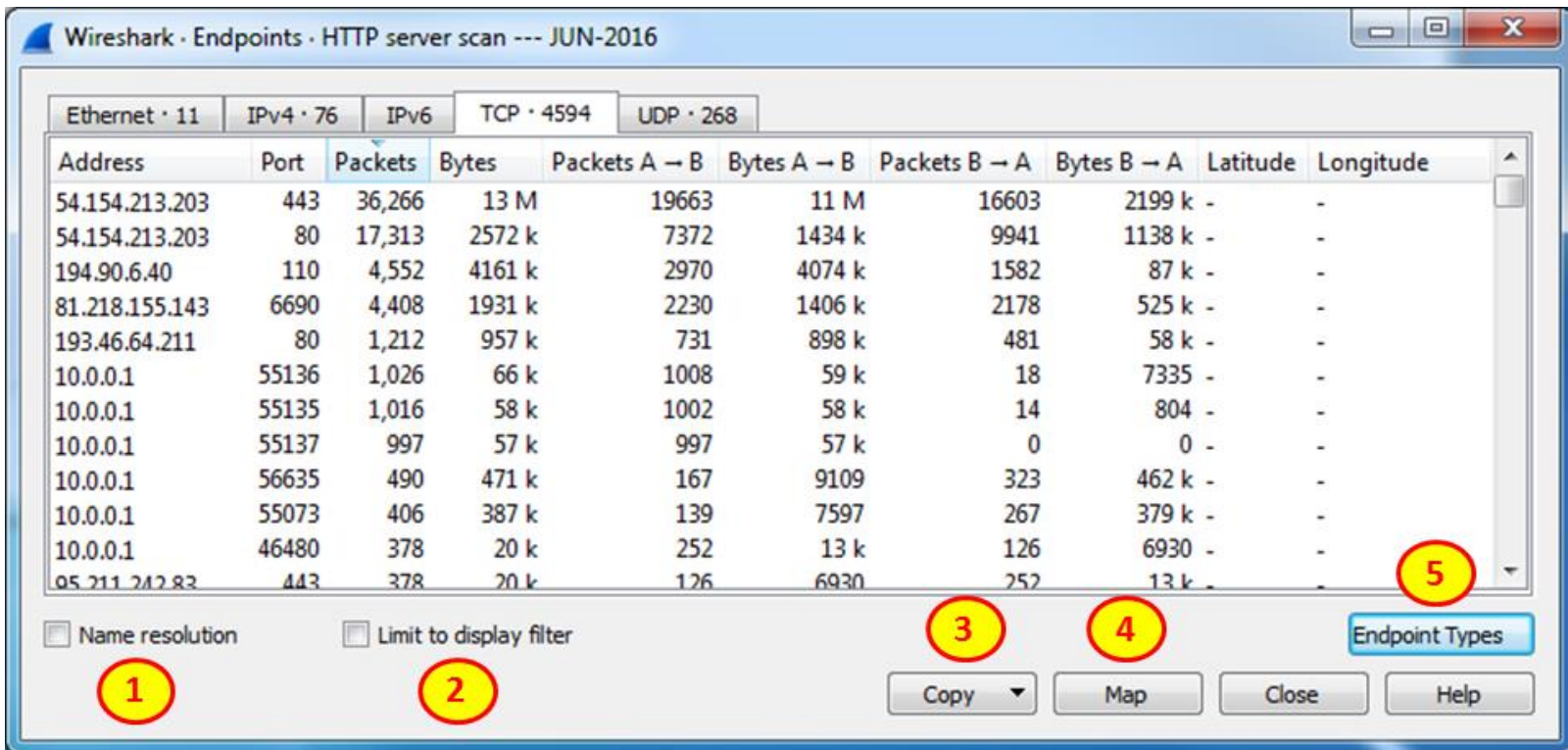
How to do it...

To view the endpoint statistics, follow these steps:

1. From the Statistics menu, choose Endpoints:



2. The following window will come up:



In this window, you will be able to see layer 2, 3, and 4 endpoints, which is Ethernet, IP, and TCP or UDP.

From the left-hand side of the window you can see (here is an example for the TCP tab):

- Endpoint IP address and port number on this host
- Total packets sent, and bytes received from and to this host
- Packets to the host (Packets A → B) and bytes to host (Bytes A → B)
- Packets to the host (Packets B → A) and bytes to host (Bytes B → A)
- The Latitude and Longitude columns applicable with the GeoIP configured

At the bottom of the window we have the following checkboxes:

- Name resolution: Provide name resolution in cases where it is configured in the name resolution under the view menu.
- Limit to display filter: To show statistics only for the display filter configured on the main window.
- Copy: Copy the list values to the clipboard in CSV or YAML format.
- Map: In cases where GeoIP is configured, shows the geographic information on the geographical map.

How it works...

Quite simply, it gives statistics on all the endpoints Wireshark has discovered. It can be any situation, such as the following:

- Few Ethernet (even on) end nodes (that is, MAC addresses), with many IP end nodes (that is, IP addresses)—this will be the case where, for example, we have a router that sends/receives packets from many remote devices.
- Few IP end nodes with many TCP end nodes—this will be the case for many TCP connections per host. Can be a regular operation of a server with many connections, and it could also be a kind of attack that comes through the network (SYN attack).

14.HOW TO RUN NMAP SCAN.

Are you worried about the security of your network or the security of someone else's?

Ensuring that your router is protected from unwanted intruders is one of the foundations of a secure network. One of the basic tools for this job is Nmap, or Network Mapper. This program will scan a target and report which ports are open and which are closed, among other things. Security specialists use this program to test the security of a network. To learn how to use it yourself, see Step 1 below.

Method 1

Using Zenmap [Download Article](#)

1

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

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

2

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

3

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

4

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

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

5

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

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

6

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

7

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

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

Method 2

Using the Command Line [Download Article](#)

1

Install Nmap. Before using Nmap, you will need to install it so that you can run it from the command line of your operating system. Nmap is small and available for free from the developer. Follow the instructions below for your operating system:

- **Linux** - Download and install Nmap from your repository. Nmap is available through most of the major Linux repositories. Enter in the command below based on your distribution:
 - Red Hat, Fedora, SUSE

```
rpm -vhU http://nmap.org/dist/nmap-6.40-1.i386.rpm (32-bit) OR  
rpm -vhU http://nmap.org/dist/nmap-6.40-1.x86_64.rpm (64-bit)
```
 - Debian, Ubuntu

```
sudo apt-get install nmap
```
- **Windows** - Download the Nmap installer. This can be found for free from the developer's website. It is highly recommended that you download directly from the developer to avoid any potential viruses or fake files. Using the installer allows you to quickly install the command line Nmap tools without having to worry about extracting to the right folder.

- If you don't want the Zenmap graphical user interface, you can uncheck it during the installation process.
- **Mac OS X** – Download the Nmap disk image. This can be found for free from the developer's website. It is highly recommended that you download directly from the developer to avoid any potential viruses or fake files. Use the included installer to install Nmap on your system. Nmap requires OS X 10.6 or later.

2

Open your command line. Nmap commands are run from the command line, and the results are displayed beneath the command. You can use variables to modify the scan. You can run the scan from any directory on the command line.

- **Linux** - Open the terminal if you are using a GUI for your Linux distribution. The location of the terminal varies by distribution
- **Windows** - This can be accessed by pressing the Windows key + R and then typing "cmd" into the Run field. Windows 8 users can press Windows key + X and select Command Prompt from the menu. You can run an Nmap scan from any directory.
- **Mac OS X** - Open the Terminal application located in the Utility subfolder of your Applications folder.

3

Run a scan of your target's ports. To start a basic scan, type `nmap <target>`. This will ping the target and scan the ports. This is an easily-detected scan. The results will be displayed on your screen. You may need to scroll back up to see all of the results.

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

4

Run a modified scan. You can use command line variables to change the parameters of the scan, resulting in more detailed or less detailed results. Changing the scan variables will change the intrusiveness of the scan. You can add multiple variables by placing a space between each one. Variables come before the target: `nmap <variable> <variable> <target>`^[3]

- **-sS** - This is a SYN stealth scan. It is less detectable than a standard scan, but may take longer. Many modern firewalls can detect an -sS scan.
- **-sn** - This is a ping scan. This will disable port scanning, and will only check to see if the host is online.
- **-O** - This is an operating system scan. The scan will attempt to determine the operating system of the target.
- **-A** - This variable enables several of the most commonly used scans: OS detection, version detection, script scanning, and traceroute.
- **-F** - This enables fast mode, and will reduce the number of ports scanned.
- **-v** - This will show more information in your results, making them easier to read.

5

Output the scan to an XML file. You can set your scan results to be outputted as an XML file so that you can easily read them in any web browser. To do this, you will need to use the **-oX** variable, as well as set a filename for the new XML file. A completed command would look similar to `nmap -oX Scan_Results.xml <target>`.

- The XML file will be saved to whatever your current working.

15. Operation system Detection using Nmap

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.

