COMPUTER NETWORKS LAB

Course Objectives

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

Course Outcomes

1. Implement data link layer farming methods
2. Analyze error detection and error correction codes.
3. Implement and analyze routing and congestion issues in network design.
4. Implement Encoding and Decoding techniques used in presentation layer
5. To be able to work with different network tools

LIST OF EXPERIMENTS

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

Aim : To implement the data link layer framing methods such as bit stuffing.

Software Used : Turbo c++/ Devc++/ Code blocks

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

---

1)b)Aim : To implement the data link layer framing methods such as character stuffing.

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
#include<string.h>
#include<process.h>
void main()
{
        inti=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

sk

Enter position

1

Enter the character

s

frame after stuffing:

dlestxdlesdleskdleetx

1)c)Aim : To implement the data link layer framing methods such as character.

Software Used : Turbo c++ / Devc++ / Code blocks

Program:

```
#include<stdio.h>
//#include<conio.h>
#include<string.h>
char data[20][20];
int n;
int main()
{
        //clrscr();
        inti,ch,j;
        chartmp[20][20];

        printf("Enter the number of frames:");
        scanf("%d",&n);
        for(i=0;i<=n;i++)
        {
                if(i!=0)
                {
                        printf("frame%d:",i);
                        fflush(stdin);
                        gets(data[i]);
```

```
                }
        }
/*saving frame with count and data*/
        for(i=0;i<=n;i++)
        {
                tmp[i][0]=49+strlen(data[i]);
                tmp[i][1]='\0';
                strcat(tmp[i],data[i]);
        }
        printf("\n\t\tAT THE SENDER:\n");
        printf("Data as frames:\n");
        for(i=1;i<=n;i++)
        {
                printf("Frame%d:",i);
                puts(tmp[i]);
        }
                printf("Data transmitted:");
                for(i=1;i<=n;i++)
                printf("%s",tmp[i]);
                printf("\n\t\tAT THE RECEIVER\n");
                printf("The data received:");
                for(i=1;i<=n;i++)
                {
                        ch=(int)(tmp[i][0]-49);
                        for(j=1;j<=ch;j++)
                        data[i][j-1]=tmp[i][j];
                        data[i][j-1]='\0';
                }
                printf("\n The data after removing count char:");
```

```
        for(i=1;i<=n;i++)
                printf("%s",data[i]);
        printf("\n The data in frame form:\n");
        for(i=1;i<=n;i++)
        {
                printf("Frame%d:",i);
                puts(data[i]);
        }
        getch();
        return 0;
}
```

Output :

Enter the number of frames:2

frame1:scient

frame2:engineering

AT THE SENDER:

Data as frames:

Frame1:7scient

Frame2:<engineering

Data transmitted:7scient<engineering

AT THE RECEIVER

The data received:

The data after removing count char:scientengineering

The data in frame form:

Frame1:scient

Frame2:engineering

Exp2 : Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP

Aim : To implement code for the polynomials CRC-12, CRC-16 and CRC CCIP

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
//#include<conio.h>
int main(void)
{
        int data[50],div[16],rem[16];
        intdatalen, divlen, i,j,k;
        intch;
        //clrscr();
        printf("Enter the data: ");
        i = 0;
        while((ch = fgetc(stdin)) != '\n')
        {
                if(ch == '1')
                data[i] = 1;
                else
```

```
                data[i] = 0;
                i++;
}
datalen = i;
printf("\nEnter the divisor: ");
i = 0;
while((ch = fgetc(stdin)) != '\n')
{
        if(ch == '1')
                div[i] = 1;
        else
                div[i] = 0;
        i++;
}
divlen = i;
for(i = datalen ; i<datalen + divlen - 1 ; i++)
data[i] = 0;
datalen = datalen + divlen - 1;
for(i = 0 ; i<divlen ; i++)
rem[i] = data[i];
k = divlen-1;
while(k <datalen)
if(rem[0] == 1)
{
        for(i = 0 ; i<divlen ; i++)
        rem[i] = rem[i] ^ div[i];
}
else
{
```

```
            if(k == datalen-1)
            break;
            for(i = 0 ; i< divlen-1 ; i++)
            {
                  rem[i] = rem[i+1];
                  printf("%d",rem[i]);
            }
            rem[i] = data[++k];
            printf("%d\n",rem[i]);
      }
      j=1;
      for(i = datalen - divlen + 1 ; i<datalen ; i++)
      {
            data[i] = rem[j++];
      }
      printf("\nThe data to be sent is\n");
      for(i = 0 ; i<datalen ; i++)
      printf("%d",data[i]);
      getch();
      return 0;
}
```

Output :

Enter the data: 10101111

Enter the divisor: 1010

0001

0011

0111

1111

1010

0000

0000

The data to be sent is

10101111000

Exp 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.

Aim : To implement simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
int main()
{
intw,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

Result :

Thus , the experiment is completed successfully

Exp 4: Take an example subnet of hosts and obtain a broadcast tree for the subnet.

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
int a[10][10],n;
int main()
{
        inti,j,root;
//clrscr();
        printf("Enter no.of nodes:");
        scanf("%d",&n);
        printf("Enter adjacent matrix\n");
        for(i=1;i<=n;i++)

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

```
                printf("Enter connecting of %d>%d::",i,j);
                scanf("%d",&a[i][j]);
        }
        printf("Enter root node:");
        scanf("%d",&root);
        adj(root);
}
    adj(int k)
    {
        inti,j;
        printf("Adjacent node of root node::\n");
        printf("%d\n",k);
        for(j=1;j<=n;j++)
        {
            if(a[k][j]==1 || a[j][k]==1)
            printf("%d\t",j);
        }
        printf("\n");
        for(i=1;i<=n;i++)
        {
            if((a[k][j]==0) && (a[i][k]==0) && (i!=k))
                printf("%d",i);
        }
        return 0;
}
```

Output :

Enter no.of nodes:5

Enter adjacent matrix

Enter connecting of 1–>1::0

Enter connecting of 1–>2::1

Enter connecting of 1–>3::1

Enter connecting of 1–>4::0

Enter connecting of 1–>5::0

Enter connecting of 2–>1::1

Enter connecting of 2–>2::0

Enter connecting of 2–>3::1

Enter connecting of 2–>4::1

Enter connecting of 2–>5::0

Enter connecting of 3–>1::1

Enter connecting of 3–>2::1

Enter connecting of 3–>3::0

Enter connecting of 3–>4::0

Enter connecting of 3–>5::0

Enter connecting of 4–>1::0

Enter connecting of 4–>2::1

Enter connecting of 4–>3::0

Enter connecting of 4–>4::0

Enter connecting of 4–>5::1

Enter connecting of 5–>1::0

Enter connecting of 5–>2::0

Enter connecting of 5–>3::0

Enter connecting of 5–>4::1

Enter connecting of 5–>5::0

Enter root node:2

Adjacent node of root node:: 2

1 3 4

5

Result : Thus , the experiment is successfully completed

Exp5 : Implement distance vector routing algorithm for obtaining routing tables at each node

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
#include<conio.h>
struct node
{
unsigned dist[20];
unsigned from[20];
}rt[10];
int main()
{
intcostmat[20][20];
```

```
intnodes,i,j,k,count=0;
printf("\nEnter the number of nodes : ");
scanf("%d",&nodes);//Enter the nodes
printf("\nEnter the cost matrix :\n");
for(i=0;i<nodes;i++)
   {
for(j=0;j<nodes;j++)
      {
scanf("%d",&costmat[i][j]);
costmat[i][i]=0;
rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix
rt[i].from[j]=j;
      }
   }
do
      {
count=0;
for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the
node i to k using the cost matrix
         //and add the distance from k to node j
for(j=0;j<nodes;j++)
for(k=0;k<nodes;k++)
if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])
            {//We calculate the minimum distance
rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j];
rt[i].from[j]=k;
count++;
            }
}while(count!=0);
```

```
for(i=0;i<nodes;i++)
      {
printf("\n\n For router %d\n",i+1);
for(j=0;j<nodes;j++)
         {
printf("\t\nnode %d via %d Distance %d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);
         }
      }
printf("\n\n");
getch();
}
```

Output:

Enter the number of nodes :

3

Enter the cost matrix :

0 2 7

2 0 1

7 1 0

For router 1

node 1 via 1 Distance 0

node 2 via 2 Distance 2

node 3 via 3 Distance 3

For router 2

node 1 via 1 Distance 2

node 2 via 2 Distance 0

node 3 via 3 Distance 1

For router 3

node 1 via 1 Distance 3

node 2 via 2 Distance 1

node 3 via 3 Distance 0

Result : Thus , the experiment is successfully completed

Exp 6: Implement data encryption and data decryption

Software Used : Turbo c++ / Devc++ / Code blocks

Program:

```
#include <stdio.h>
int main()
{
inti, x;
charstr[100];
printf("\nPlease enter a string:\t");
gets(str);
```

```
printf("\nPlease choose following options:\n");
printf("1 = Encrypt the string.\n");
printf("2 = Decrypt the string.\n");
scanf("%d", &x);
  //using switch case statements
switch(x)
  {
case 1:
for(i = 0; (i< 100 &&str[i] != '\0'); i++)
str[i] = str[i] + 3; //the key for encryption is 3 that is added to ASCII value
printf("\nEncrypted string: %s\n", str);
break;
case 2:
for(i = 0; (i< 100 &&str[i] != '\0'); i++)
str[i] = str[i] - 3; //the key for encryption is 3 that is subtracted to ASCII value
printf("\nDecrypted string: %s\n", str);
break;
default:
printf("\nError\n");
  }
return 0;
}
```

Output :

```
Please enter a string:  sceint

Please choose following options:
1 = Encrypt the string.
2 = Decrypt the string.
1
```

Encrypted string: vfhlqw

Please enter a string: sc

Please choose following options:

1 = Encrypt the string.

2 = Decrypt the string.

2

Decrypted string: p`

Result : Thus , the experiment is successfully completed

Exp 7: Write a C++ program for congestion control using Leaky bucket algorithm.

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<iostream>
#include<dos.h>
#include<stdlib.h>
#define bucketSize 512
using namespace std;
```

```
voidbktInput(inta,int b)
{
        if(a>bucketSize)
                cout<<"\n\t\tBucket overflow";
        else
        {

        while(a>b)
        {
                cout<<"\n\t\t"<<b<<" bytes outputted.";
                a-=b;

        }
        if(a>0)
                cout<<"\n\t\tLast "<<a<<" bytes sent\t";
        cout<<"\n\t\tBucket output successful";
        }
}
int main()
{
        int op, pktSize;
        randomize();
        cout<<"Enter output rate : ";
        cin>>op;
        for(inti=1;i<=5;i++)
        {
                delay(random(1000));
                pktSize=random(1000);
                cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
```

```
                bktInput(pktSize,op);
        }
return 0;
}
```

Output:

```
  Enter output rate : 100

  Packet no 0  Packet size = 3
                 Bucket output successful
                 Last 3 bytes sent
  Packet no 1  Packet size = 33
                 Bucket output successful
                 Last 33 bytes sent
  Packet no 2  Packet size = 117
                 Bucket output successful

               100 bytes outputted.
               Last 17 bytes sent
Packet no 3  Packet size = 95
               Bucket output successful
               Last 95 bytes sent
Packet no 4  Packet size = 949
               Bucket overflow
```

Result : Thus , the experiment is successfully completed

Exp 8 : Write a program for frame sorting technique used in buffers.

Software Used : Turbo c++ / Devc++ / Code blocks

Program :

```
#include<stdio.h>
#include<string.h>
#define FRAM_TXT_SIZ 3
```

```
#define MAX_NOF_FRAM 127
charstr[FRAM_TXT_SIZ*MAX_NOF_FRAM];
struct frame // structure maintained to hold frames
{
        char text[FRAM_TXT_SIZ];
        intseq_no;
}
fr[MAX_NOF_FRAM], shuf_ary[MAX_NOF_FRAM];
intassign_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, constint 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( constintno_frames ) // function shuffles the frames
{
        inti, 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(constintno_frames) // sorts the frames
{
        inti,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()
{
        intno_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: scient institute

After assigning sequence numbers:

0:sci 1:ent 2:ins 3:tit 4: ute

AFTER SHUFFLING:

 4: ute 1:ent 0:sci 2:ins 3:tit

AFTER SORTING

scient institute

Exp9 :

Wireshark

i. Packet Capture Using Wire shark

ii. Starting Wire shark

iii. Viewing Captured Traffic

iv. Analysis and Statistics & Filters.

Introduction to Wireshark:

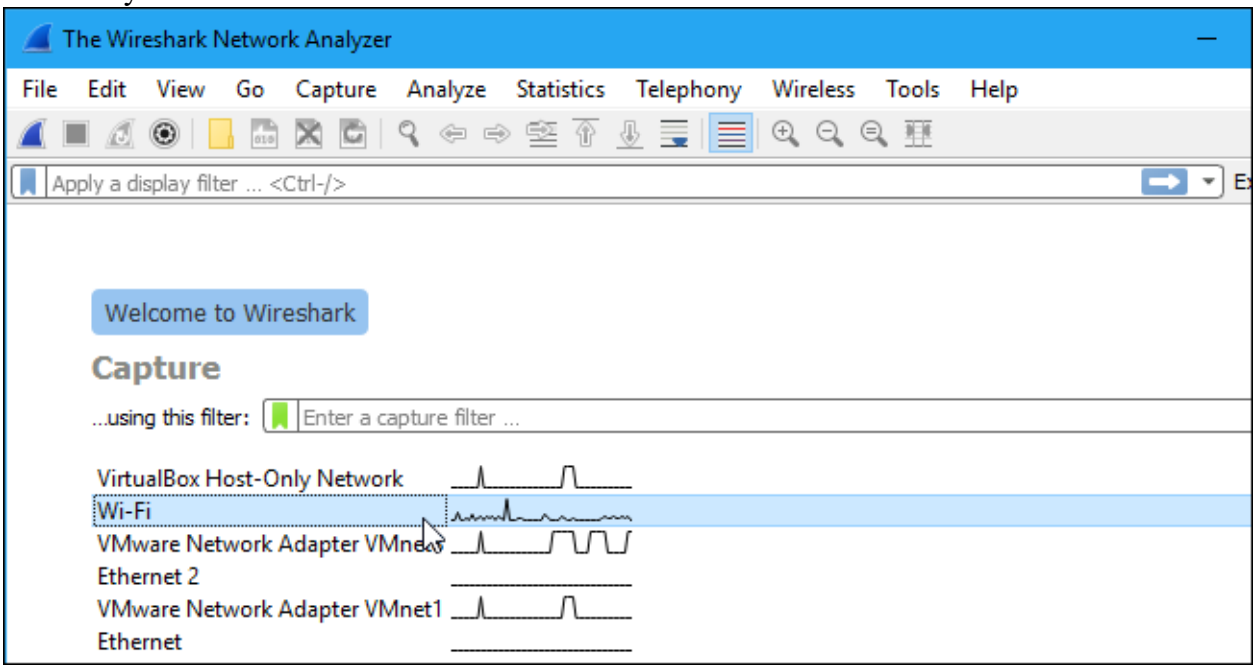
Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.You could think of a network packet analyzer as a measuring device for examining what's happening inside a network cable, just like an electrician uses a voltmeter for examining what's happening inside an electric cable .Wireshark is available for free, is open source, and is one of the best packet analyzers available today.

You can download Wireshark for Windows or macOS from its official website. If you're using Linux or another UNIX-like system, you'll probably find Wireshark in its package repositories. For example, if you're using Ubuntu, you'll find Wireshark in the Ubuntu Software Center.

Packet Capture Using Wire shark

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface.

You can configure advanced features by clicking Capture > Options, but this isn't necessary for now.



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

If you have promiscuous mode enabled—it's enabled by default—you'll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the "Enable promiscuous mode on all interfaces" checkbox is activated at the bottom of this window.

ii Starting Wireshark:

You can start Wireshark from the command line, but it can also be started from most Window managers as well.

Wireshark 3.5.0 (v3.5.0rc0-21-gce47866a4337)

Interactively dump and analyze network traffic.

See https://www.wireshark.org for more information.

Usage: wireshark [options] ... [<infile>]

Capture interface:

-i<interface>, --interface <interface>

name or idx of interface (def: first non-loopback)

-f <capture filter> packet filter in libpcap filter syntax

-s <snaplen>, --snapshot-length <snaplen>

packet snapshot length (def: appropriate maximum)

-p, --no-promiscuous-mode

don't capture in promiscuous mode

-k start capturing immediately (def: do nothing)

-S update packet display when new packets are captured

-l turn on automatic scrolling while -S is in use

-I, --monitor-mode capture in monitor mode, if available

-B <buffer size>, --buffer-size <buffer size>

size of kernel buffer (def: 2MB)

-y <link type>, --linktype<link type>

link layer type (def: first appropriate)

--time-stamp-type <type> timestamp method for interface

-D, --list-interfaces print list of interfaces and exit

-L, --list-data-link-types

print list of link-layer types of iface and exit

--list-time-stamp-types print list of timestamp types for iface and exit

Capture stop conditions:

-c <packet count> stop after n packets (def: infinite)

-a <autostop cond.> ..., --autostop<autostop cond.> ...

duration:NUM - stop after NUM seconds

filesize:NUM - stop this file after NUM KB

files:NUM - stop after NUM files

packets:NUM - stop after NUM packets

Capture output:

-b <ringbuffer opt.> ..., --ring-buffer <ringbuffer opt.>

duration:NUM - switch to next file after NUM secs

filesize:NUM - switch to next file after NUM KB

files:NUM - ringbuffer: replace after NUM files

packets:NUM - switch to next file after NUM packets

interval:NUM - switch to next file when the time is

an exact multiple of NUM secs

Input file:

-r <infile>, --read-file <infile>

set the filename to read from (no pipes or stdin!)

Processing:

-R <read filter>, --read-filter <read filter>

packet filter in Wireshark display filter syntax

-n disable all name resolutions (def: all enabled)

-N <name resolve flags> enable specific name resolution(s): "mnNtdv"

-d <layer_type>==<selector>,<decode_as_protocol> ...

"Decode As", see the man page for details

Example: tcp.port==8888,http

--enable-protocol <proto_name>

enable dissection of proto_name

--disable-protocol <proto_name>

disable dissection of proto_name

--enable-heuristic <short_name>

enable dissection of heuristic protocol

--disable-heuristic <short_name>

disable dissection of heuristic protocol

User interface:

-C <config profile> start with specified configuration profile

-H hide the capture info dialog during packet capture

-Y <display filter>, --display-filter <display filter>

start with the given display filter

-g <packet number> go to specified packet number after "-r"

-J <jump filter> jump to the first packet matching the (display)

filter

-j search backwards for a matching packet after "-J"

-t a|ad|adoy|d|dd|e|r|u|ud|udoy

format of time stamps (def: r: rel. to first)

-u s|hms output format of seconds (def: s: seconds)

-X <key>:<value>eXtension options, see man page for details

-z <statistics> show various statistics, see man page for details

Output:

-w <outfile|-> set the output filename (or '-' for stdout)

--capture-comment <comment>

set the capture file comment, if supported

Miscellaneous:

-h, --help display this help and exit

-v, --version display version info and exit

-P <key>:<path>persconf:path - personal configuration files

persdata:path - personal data files

-o <name>:<value> ... override preference or recent setting

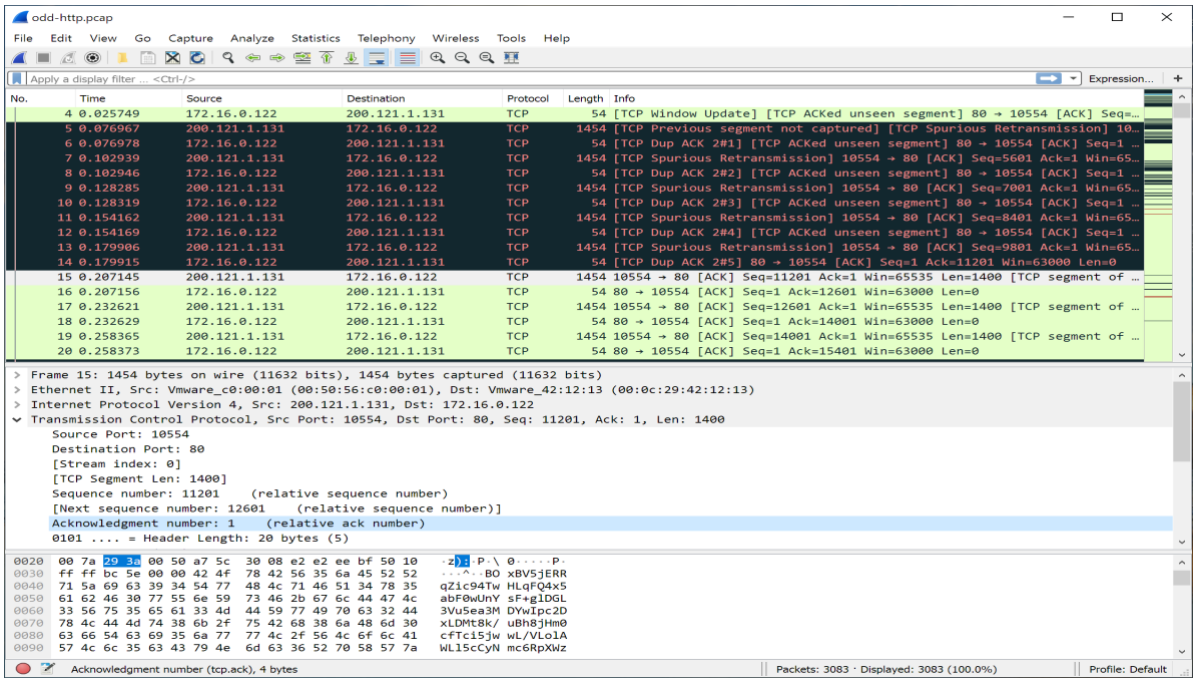
-K <keytab>keytab file to use for kerberos decryption

--display <X display> X display to use

--fullscreen start Wireshark in full screen

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.

iv Analysis and Statistics & filters

Whenever we type any commands in the filter command box, it turns green if your command is correct. It turns red if it is incorrect or the Wireshark does not recognize your command.

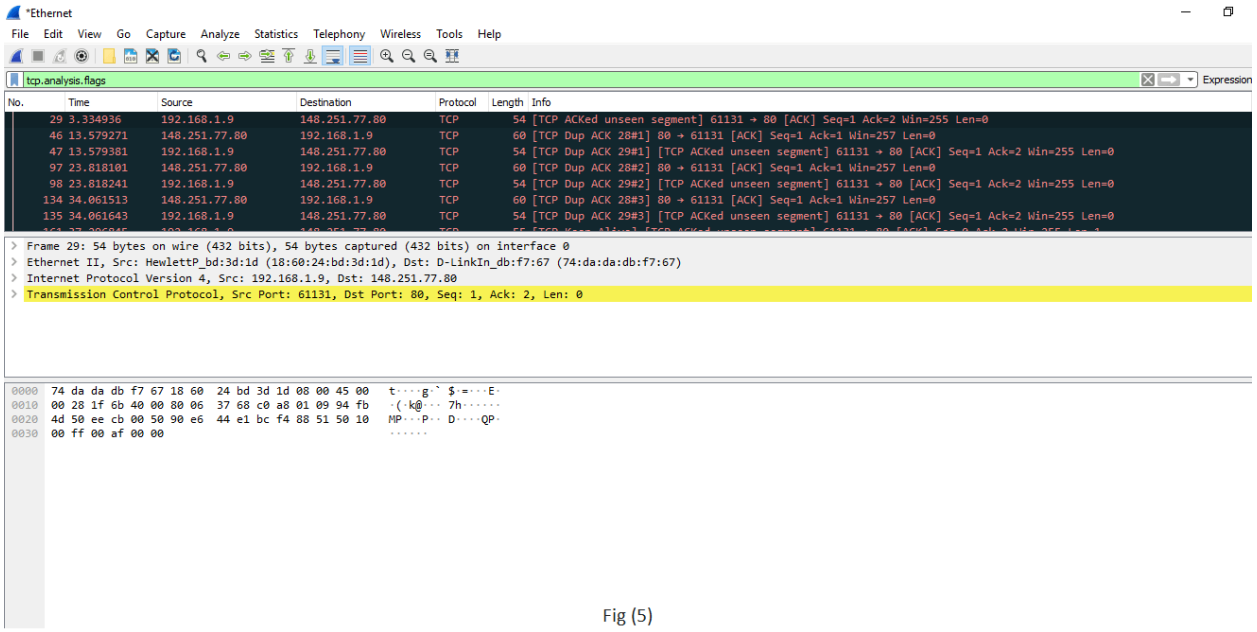

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.analysis.flags
No. Time Source Destination Protocol Length Info
29 3.334936 192.168.1.9 148.251.77.80 TCP 54 [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
46 13.579271 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#1] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
47 13.579381 192.168.1.9 148.251.77.80 TCP 54 [TCP Dup ACK 29#1] [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
97 23.818101 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#2] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
98 23.818241 192.168.1.9 148.251.77.80 TCP 54 [TCP Dup ACK 29#2] [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
134 34.061513 148.251.77.80 192.168.1.9 TCP 60 [TCP Dup ACK 28#3] 80 → 61131 [ACK] Seq=1 Ack=1 Win=257 Len=0
135 34.061643 192.168.1.9 148.251.77.80 TCP 54 [TCP Dup ACK 29#3] [TCP ACKed unseen segment] 61131 → 80 [ACK] Seq=1 Ack=2 Win=255 Len=0
Frame 29: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
Ethernet II, Src: HewlettP_bd:3d:1d (18:60:24:bd:3d:1d), Dst: D-LinkIn_db:f7:67 (74:da:da:db:f7:67)
Internet Protocol Version 4, Src: 192.168.1.9, Dst: 148.251.77.80
Transmission Control Protocol, Src Port: 61131, Dst Port: 80, Seq: 1, Ack: 2, Len: 0
0000 74 da da db f7 67 18 60 24 bd 3d 1d 08 00 45 00
0010 00 28 1f 6b 40 00 80 06 37 68 c0 a8 01 09 94 fb
0020 4d 50 ee cb 00 50 90 e6 44 e1 bc f4 88 51 50 10
0030 00 ff 00 af 00 00

Fig (5)

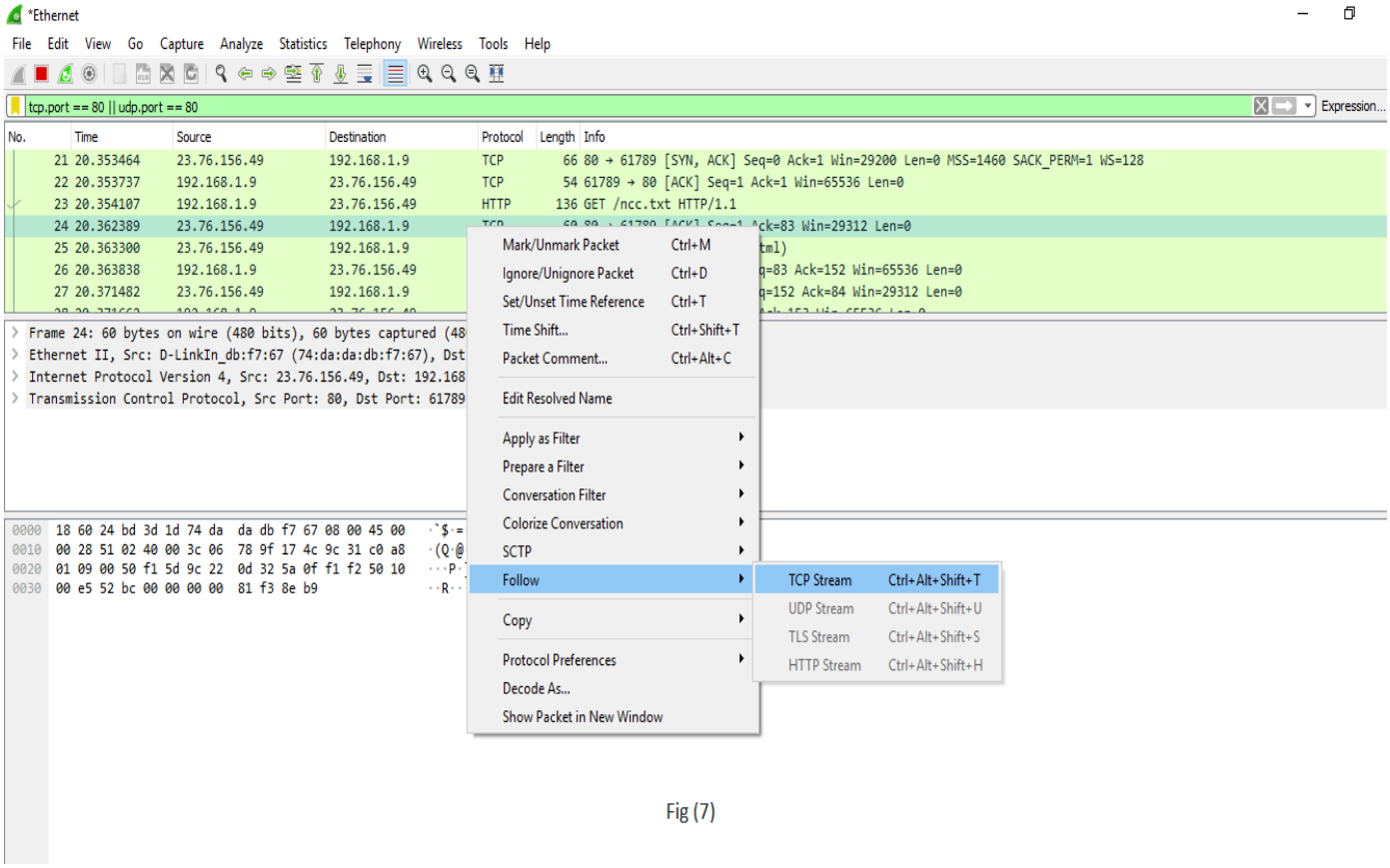

*Ethernet
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.port == 80 || udp.port == 80
No. Time Source Destination Protocol Length Info
21 20.353464 23.76.156.49 192.168.1.9 TCP 66 80 → 61789 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
22 20.353737 192.168.1.9 23.76.156.49 TCP 54 61789 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0
23 20.354107 192.168.1.9 23.76.156.49 HTTP 136 GET /ncc.txt HTTP/1.1
24 20.362389 23.76.156.49 192.168.1.9
25 20.363300 23.76.156.49 192.168.1.9
26 20.363838 192.168.1.9 23.76.156.49
27 20.371482 23.76.156.49 192.168.1.9
Frame 24: 60 bytes on wire (480 bits), 60 bytes captured (48
Ethernet II, Src: D-LinkIn_db:f7:67 (74:da:da:db:f7:67), Dst
Internet Protocol Version 4, Src: 23.76.156.49, Dst: 192.168
Transmission Control Protocol, Src Port: 80, Dst Port: 61789
0000 18 60 24 bd 3d 1d 74 da da db f7 67 08 00 45 00
0010 00 28 51 02 40 00 3c 06 78 9f 17 4c 9c 31 c0 a8
0020 01 09 00 50 f1 5d 9c 22 0d 32 5a 0f f1 f2 50 10
0030 00 e5 52 bc 00 00 00 00 81 f3 8e b9
Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
TCP Stream Ctrl+Alt+Shift+T
UDP Stream Ctrl+Alt+Shift+U
TLS Stream Ctrl+Alt+Shift+S
HTTP Stream Ctrl+Alt+Shift+H

Fig (7)

Exp 10 :How to run NMap scan

Description

Nmap is a free, open source and multi-platform network security scanner used for network discovery and security auditing. Amongst other things, it allows you to create a network inventory, managing service upgrade schedules, monitor host or service uptime and scan for open ports and services on a host. To get started, download and install Nmap from the nmap.org website and then launch a command prompt.

Executing Nmap on Windows

Nmap releases now include the Zenmap graphical user interface for Nmap. If you used the Nmap installer and left the Zenmap field checked, there should be a new Zenmap entry on your desktop and Start Menu.

Here are detailed instructions for users who are unfamiliar with command-line interfaces:

Make sure the user you are logged in as has administrative privileges on the computer (user should be a member of the administrators group).

Open a command/DOS Window. Though it can be found in the program menu tree, the simplest approach is to choose “Start” -> “Run” and type cmd<enter>. Opening a Cygwin window (if you installed it) by clicking on the Cygwin icon on the desktop works too, although the necessary commands differ slightly from those shown here.

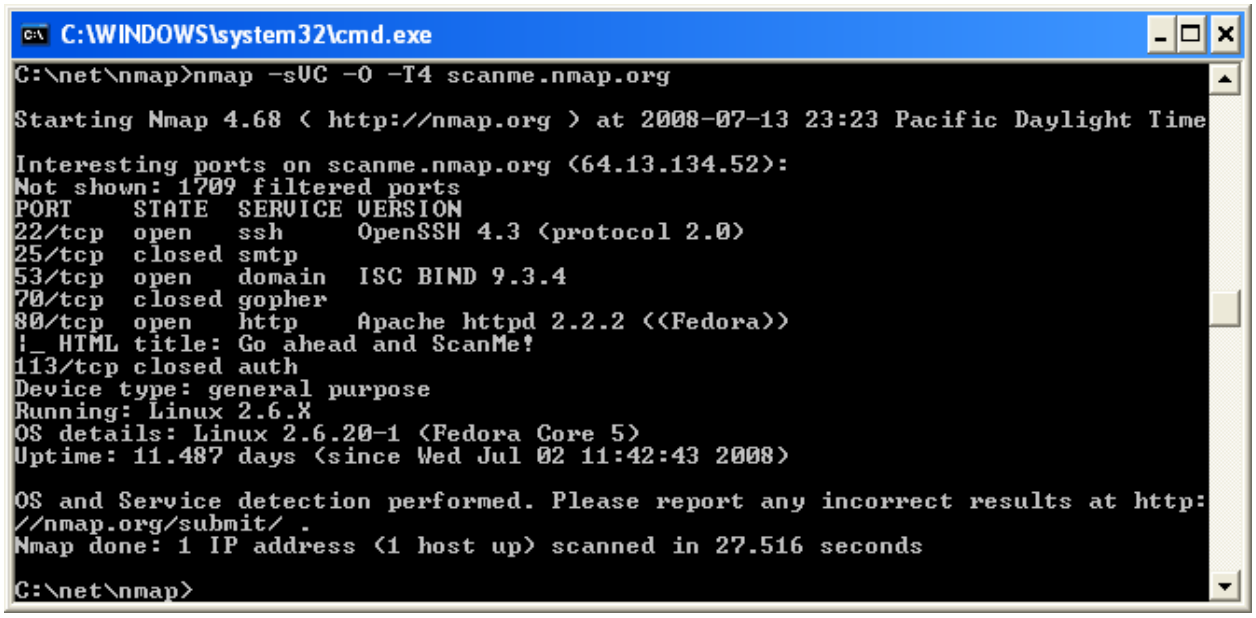
Change to the directory you installed Nmap into. You can skip this step if Nmap is already in your command path (the Zenmapisntaller adds it there by default). Otherwise, type the following commands.

c:

cd "\Program Files (x86)\Nmap"

In Windows releases prior to Windows 7, specify \Program Files\Nmap instead. The directory will also be different if you chose to install Nmap in a non-default location.

Figure 2.1. Executing Nmap from a Windows command shell



If you execute Nmap frequently, you can add the Nmap directory (c:\Program Files (x86)\Nmap by default on Windows 7) to your command execution path. The exact place to set this varies by Windows platform. On my Windows XP box, which installs Nmap in c:\Program Files\Nmap, I do the following:

- From the desktop, right click on My Computer and then click “properties”.
- In the System Properties window, click the “Advanced” tab.
- Click the “Environment Variables” button.
- Choose Path from the System variables section, then hit edit.
- Add a semi-colon and then your Nmap directory (e.g. c:\Program Files\Nmap) to the end of the value.

- Open a new DOS window and you should be able to execute a command such as nmap scanme.nmap.org from any directory.

Exp 11 : Operating System Detection using Nmap

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.

Exp12 :Implement Dijsktra's algorithm to compute the shortest path through a network

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

voiddijkstra(int G[MAX][MAX],intn,intstartnode);
```

```
int main()
{
        int G[MAX][MAX],i,j,n,u;
        printf("Enter no. of vertices:");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");

        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&G[i][j]);

        printf("\nEnter the starting node:");
        scanf("%d",&u);
        dijkstra(G,n,u);

        return 0;
}

voiddijkstra(int G[MAX][MAX],intn,intstartnode)
{

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

        //pred[] stores the predecessor of each node
        //count gives the number of nodes seen so far
        //create the cost matrix
        for(i=0;i<n;i++)
```

```
        for(j=0;j<n;j++)
                if(G[i][j]==0)
                        cost[i][j]=INFINITY;
                else
                        cost[i][j]=G[i][j];
        //initialize pred[],distance[] and visited[]
for(i=0;i<n;i++)
{
        distance[i]=cost[startnode][i];
        pred[i]=startnode;
        visited[i]=0;
}

distance[startnode]=0;
visited[startnode]=1;
count=1;

while(count<n-1)
{
        mindistance=INFINITY;

        //nextnode gives the node at minimum distance
        for(i=0;i<n;i++)
                if(distance[i]<mindistance&&!visited[i])
                {
                        mindistance=distance[i];
                        nextnode=i;
                }
```

```
                    //check if a better path exists through nextnode
                    visited[nextnode]=1;
                    for(i=0;i<n;i++)
                            if(!visited[i])
                                    if(mindistance+cost[nextnode][i]<distance[i])
                                    {
                                            distance[i]=mindistance+cost[nextnode][i];
                                            pred[i]=nextnode;
                                    }
            count++;
    }

    //print the path and distance of each node
    for(i=0;i<n;i++)
            if(i!=startnode)
            {
                    printf("\nDistance of node%d=%d",i,distance[i]);
                    printf("\nPath=%d",i);

                    j=i;
                    do
                    {
                            j=pred[j];
                            printf("<-%d",j);
                    }while(j!=startnode);
    }
}
```

Output :

Enter no. of vertices:5

Enter the adjacency matrix:

0 10 0 30 100

10 0 50 0 0

0 50 0 20 100

30 0 20 0 60

10 10 10 10

Enter the starting node:0

Distance of node1=10

Path=1<-0

Distance of node2=50

Path=2<-3<-0

Distance of node3=30

Path=3<-0

Distance of node4=90

Path=4<-3<-0

Exp 13 :NS2 SIMULATOR

1) Introduction to NS2 simulator

NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL

Features of NS2:

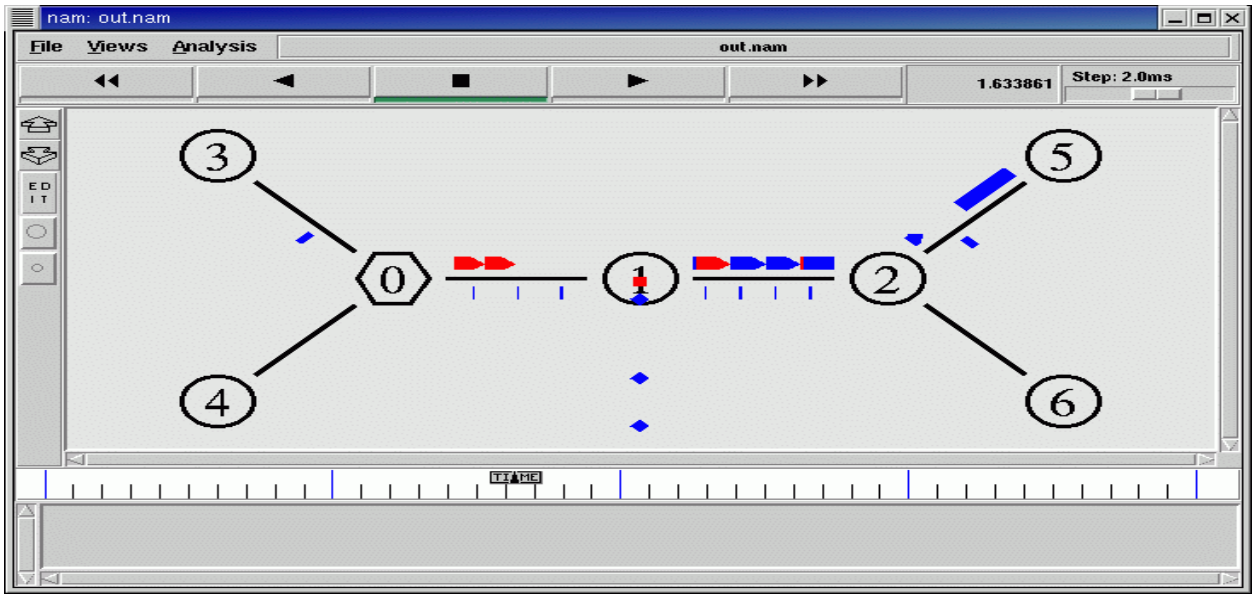
- It is a discrete event simulator for networking research.
- It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.
- It simulates wired and wireless network.
- It is primarily Unix based.
- Uses TCL as its scripting language.
- Otcl: Object oriented support
- Tclcl: C++ and otcl linkage
- Discrete event scheduler

2) Simulate to Find the Number of Packets Dropped

TCL script

```
set ns [ new Simulator ]
setnf [ open lab4.nam w ]
$ns namtrace-all $nf
settf [ open lab4.tr w ]
$ns trace-all $tf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
set p1 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001
set p2 [new Agent/Ping] # letters A and P should be capital $ns attach-agent $n1 $p2
set p3 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001
set p4 [new Agent/Ping] # letters A and P should be capital $ns attach-agent $n3 $p4
set p5 [new Agent/Ping] # letters A and P should be capital $ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$ns queue-limit $n4 $n5 2
Agent/Ping instprocrecv {from rtt} {
$self instvar node_
puts "node [$node_ id]received answer from $from with round trip time $rttmsec"
}
# please provide space between $node_ and id. No space between $ and from. No space
between and $ and rtt */
$ns connect $p1 $p5
$ns connect $p3 $p4
proc finish { } {
global ns nftf
$ns flush-trace
close $nf
```

```
close $tf
execnam lab4.nam &
exit 0
}
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "finish"
$ns run
```



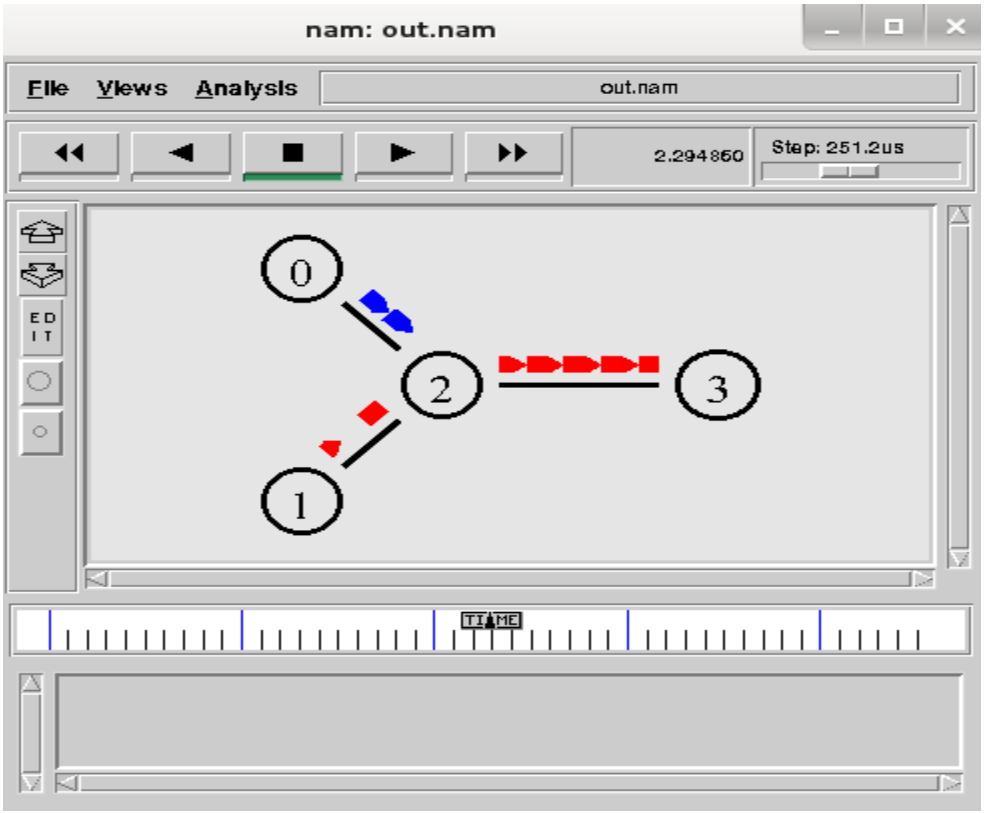
3) Simulate to Find the Number of Packets Dropped by TCP/UDP

TCL script

```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the NAM trace file
setnf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
global ns nf
      $ns flush-trace
      #Close the NAM trace file
close $nf
      #Execute NAM on the trace file
execnamout.nam&
exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
```

```
settcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
setudp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection
setcbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
#Detachtcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"
#Run the simulation
$ns run
```



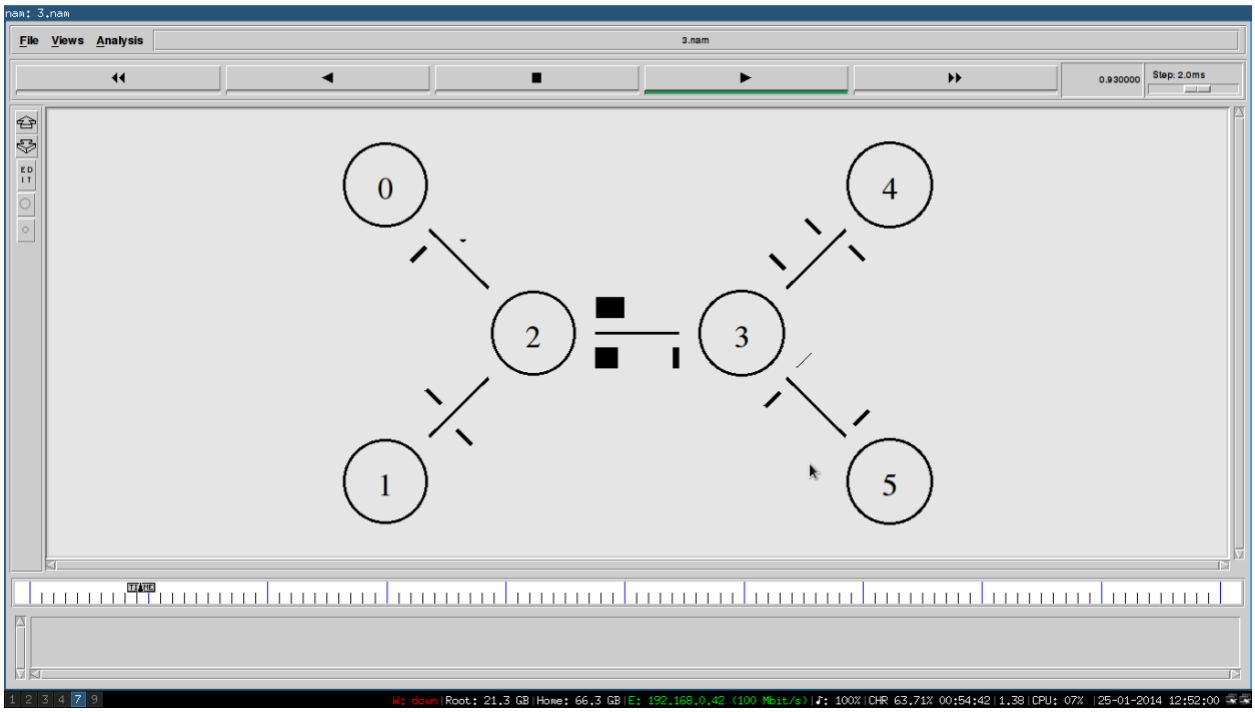
4) Simulate to Find the Number of Packets Dropped due to Congestion

TCL Script

```
#Create a simulator object
set ns [new Simulator]
#Open a nam trace file
setnf [open PING.nam w]
$ns namtrace-all $nf
#Open a trace file
setnt [open PING.tr w]
$ns trace-all $nt
#Define a finish procedure
proc finish {} {
```

```
global ns nfnt
        $ns flush-trace
close $nf
close $nt
execnamPING.nam&
exit 0
   }
#Create six  nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
# Connect the nodes with two links
  $ns duplex-link $n0 $n1 1Mb 10ms DropTail
   $ns duplex-link $n2 $n1 1Mb 10ms DropTail
   $ns duplex-link $n3 $n1 1Mb 10ms DropTail
   $ns duplex-link $n4 $n1 1Mb 10ms DropTail
   $ns duplex-link $n5 $n1 1Mb 10ms DropTail
#Set queue length
   $ns queue-limit $n0 $n1 5
   $ns queue-limit $n2 $n1 2
   $ns queue-limit $n3 $n1 5
   $ns queue-limit $n4 $n1 2
   $ns queue-limit $n5 $n1 2
#Label the nodes
   $n0 label "ping0"
   $n1 label "Router"
   $n2 label "ping2"
   $n3 label "ping3"
   $n4 label "ping4"
   $n5 label "ping5"
#Color the flow
   $ns color 2 Blue
   $ns color 3 Red
   $ns color 4 Yellow
   $ns color 5 Green
#Define s ‘recv’ function for the class ‘Agent/Ping’
   Agent/Ping instprocrecv {from rtt} {
     $self instvar node_
puts "node [$node_ id] received ping answer from \
     $from with round-trip-time $rttms."
```

```
  }
#Create ping agents and attach them to the nodes
set p0 [new Agent/Ping]
  $ns attach-agent $n0 $p0
  $p0 set class_ 1
set p2 [new Agent/Ping]
  $ns attach-agent $n2 $p2
  $p2 set class_ 2
set p3 [new Agent/Ping]
  $ns attach-agent $n3 $p3
  $p3 set class_ 3
set p4 [new Agent/Ping]
  $ns attach-agent $n4 $p4
  $p4 set class_ 4
set p5 [new Agent/Ping]
  $ns attach-agent $n5 $p5
  $p5 set class_ 5
#Connect the two agents
  $ns connect $p2 $p5
  $ns connect $p3 $p5
procSendPingPacket { } {
global ns p2 p3
setintervalTime 0.001
set now [$ns now]
    $ns at [expr $now+$intervalTime] "$p2 send"
    $ns at [expr $now+$intervalTime] "$p3 send"
    $ns at [expr $now+$intervalTime] "SendPingPacket"
  }
    $ns at 0.1 "SendPingPacket"
    $ns at 2.0 "finish"
    $ns run
```

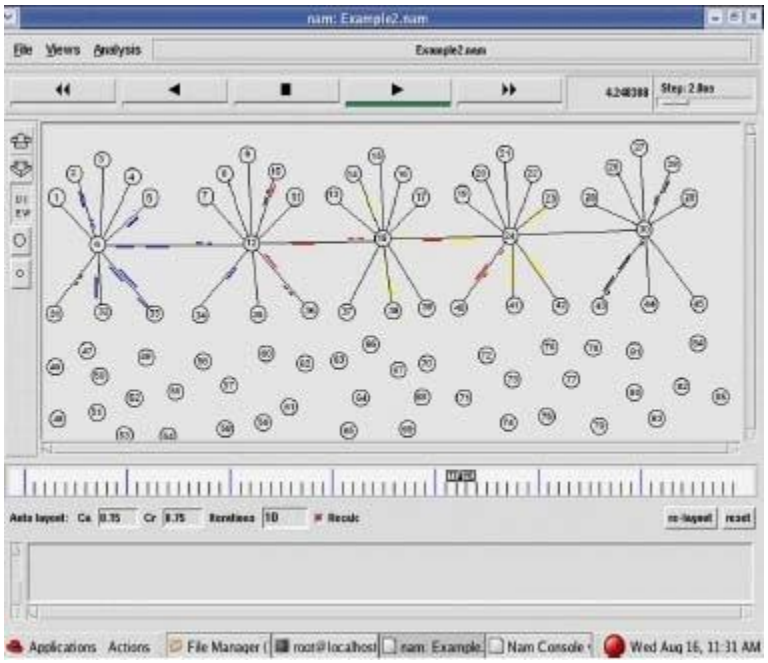


5) Simulate to Compare Data Rate& Throughput

TCL Script

```
# create a simulator object
set ns [new Simulator]
# create four nodes
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]
# create links between the nodes
$ns duplex-link $node1 $node3 2Mb 20ms DropTail
$ns duplex-link $node2 $node3 2Mb 20ms DropTail
$ns duplex-link $node3 $node4 1Mb 20ms DropTail
$ns queue-limit $node3 $node4 4
# set the display layout of nodes and links for nam
$ns duplex-link-op $node1 $node3 orient right-down
$ns duplex-link-op $node2 $node3 orient right-up
$ns duplex-link-op $node3 $node4 orient right
# define different colors for nam data flows
$ns color 0 Green
```

$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow
monitor the queue for the link between node 2 and node 3
$ns duplex-link-op $node3 $node4 queuePos 0.5



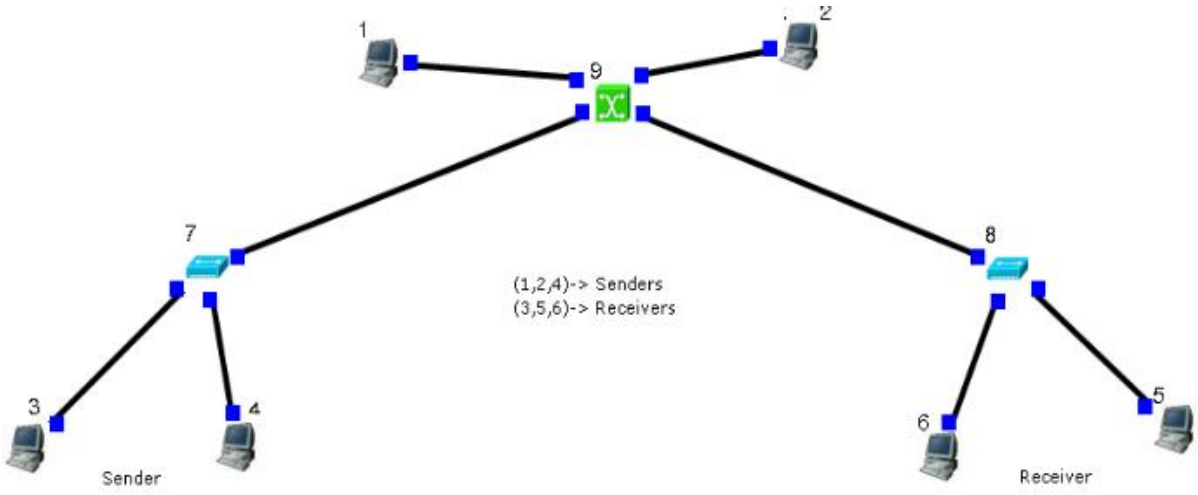
6) Simulate to Plot Congestion for Different Source/Destination

TCL script

Sender:-
stcp –p 2000 –l 1024 1.0.1.4
Receiver:-
rtcp –p 2000 –l 1024
Parameters:-
Receiver side Collision Packets and Drop Packets
Step1: Drawing topology
1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place HOST1 on the editor.
Repeat the above procedure and place 3 other hosts “HOST2”, “HOST3”, “HOST4”, “HOST5”, and “HOST6”on the editor.
2. Select/click the HUB icon on the toolbar and click the left mouse button on the editor, to place HUB1 on the editor.

Repeat the above procedure and place another host “HUB2” on the editor
3. Click on the LINK icon on the toolbar and connect HOST1, HOST2 and HOST3 to HUB1, HOST4, HOST5 and HOST6 to HUB2.
4. Select/click the SWITCH icon on the toolbar and click the left mouse button on the editor, to place SWITCH1 the editor.
5. Click on the LINK icon on the toolbar and connect HUB1 to SWITCH1 and HUB2 to SWITCH1.
6. Click on the “E” icon on the toolbar to save the current topology e.g: file7.tpl
(Look for the ******.tpl extension.)
NOTE: Changes cannot / (should not) be done after selecting the “E” icon.
Step2: Configuration
1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.
2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.
stcp –p 21 –l 1024 1.0.1.4
3. Click OK button on the command window to exit and once again click on the OK button on the HOST window to exit.
4. Repeat this step at HOST 2 and HOST3, but use different commands
stcp –p 23 –l 1024 1.0.1.5 at HOST2
stcp –p 25 –l 1024 1.0.1.6 at HOST3
5. Double click the left mouse button while cursor is on HOST4 to open the HOST window.
6. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.
rtcp –p 21 –l 1024
7. Click OK button on the command window to exit.
8. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.
9. Select LOG STATISTICS and select checkbox for Number of drop and collisions packets in the MAC window
10. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.
11. Repeat this step at HOST 5 and HOST6, but use different commands
rtcp –p 23 –l 1024 at HOST5
rtcp –p 25 –l 1024 at HOST6
12. Double click the left mouse button while cursor is on HOST5 to open the HOST window.
13. Click NODE EDITOR Button on the HOST5 window and select the MAC tab from the modal window that pops up.
14. Select LOG STATISTICS and select checkbox for Number of drop and collisions packets in the MAC window
15. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

16. Also select the drop and collisions at HOST6.
Step3: Simulate
i. Click "R" icon on the tool bar
ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
iii. To start playback select "►" icon located at the bottom right corner of the editor.
iv. To plot congestion window select Tools in the menu bar and select PLOT GRAPH in the drop down list.
v. In the Graph window, select File->OPEN, move to file7.results folder and the drop and collision log file.
vi. To open another Graph window, Select File->New tab on the drop down list to open up to a maximum of 6 windows
vii. To view results, Open up new TERMINAL window, move to file7.results folder and open input and output throughput log files in separate TERMINAL window.



7) Simulate to Determine the Performance with respect to Transmission of Packets

TCL Script

```
set ns [new Simulator]

#————-creating trace objects————————-#

setnt [open test2.tr w]
$ns trace-all $nt
#————-creating nam objects————————-#

setnf [open test2.nam w]
$ns namtrace-all $nf
#————-Setting color ID————————-#
$ns color 1 darkmagenta
```

```
$ns color 2 yellow
$ns color 3 blue
$ns color 4 green
$ns color 5 black
#———— Creating Network————————-#
settotalNodes 3
for {set i 0} {$i< $totalNodes} {incri} {
set node_($i) [$ns node]
}
set server 0
set router 1
set client 2
#———— Creating Duplex Link————————-#
$ns duplex-link $node_($server) $node_($router) 2Mb 50ms DropTail
$ns duplex-link $node_($router) $node_($client) 2Mb 50ms DropTail
$ns duplex-link-op $node_($server) $node_($router) orient right
$ns duplex-link-op $node_($router) $node_($client) orient right
#————Labelling————————-#
$ns at 0.0 "$node_($server) label Server"
$ns at 0.0 "$node_($router) label Router"
$ns at 0.0 "$node_($client) label Client"
$ns at 0.0 "$node_($server) color blue"
$ns at 0.0 "$node_($client) color blue"
$node_($server) shape hexagon
$node_($client) shape hexagon
#————Data Transfer between Nodes————————-#
# Defining a transport agent for sending
settcp [new Agent/TCP]
# Attaching transport agent to sender node
$ns attach-agent $node_($server) $tcp
# Defining a transport agent for receiving
set sink [new Agent/TCPSink]
# Attaching transport agent to receiver node
$ns attach-agent $node_($client) $sink
#Connecting sending and receiving transport agents
$ns connect $tcp $sink
#Defining Application instance
set ftp [new Application/FTP]
# Attaching transport agent to application agent
$ftp attach-agent $tcp
# Setting flow color
$tcp set fid_ 4
# data packet generation starting time
```

```
$ns at 1.0 "$ftp start"
# data packet generation ending time
$ns at 6.0 "$ftp stop"
#———finish procedure———#
proc finish {} {
global ns nfnt
$ns flush-trace
close $nf
close $nt
puts "running nam…"
execnam test2.nam &
exit 0
}
#Calling finish procedure
$ns at 10.0 "finish"
$ns run
```

***