Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

COMPUTER NETWORKS LAB MANUAL

III-AI&DS -Semester I

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (AI&DS)

ACADEMIC YEAR 2025-26

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510

DEPARTMENT OF

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

LAB MANUAL

Branch: AI&DS
Subject: Computer Networks Lab
Academic Year: 2025-26

Class: B.Tech- III Year-I sem
Code: R22CSE3126
Regulation: R22

Core/Elective/H&S: Core

Credits: 1

Prepared By
Name: Mrs . T.TEJASWI

Verified By
Head of the Department:

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
B. TECH – ARTIFICIAL INTELLIGENCE AND DATASCIENCE

INSTITUTION VISION.

To be a premier institution in engineering & technology and management for competency, values and social consciousness

INSTITUTION MISSION

IM_1 Provide high quality academic programs, training activities and research facilities.

IM_2 Promote Continuous Industry-Institute interaction for employability, Entrepreneurship, leadership and research aptitude among stakeholders.

IM_3 Contribute to the economical and technological development of the region, state and nation.

DEPARTMENT VISION

To produce competent professionals recognized for excellence, innovation and societal relevance by impacting their knowledge of Artificial Intelligence and Data Science.

DEPARTMENT MISSION

The Department has following Missions:

DM_1 To produce industry-ready professionals and leverage Artificial Intelligence and Data science innovative models for automation, effective decision-making, and competitive advantage.

DM_2 To develop state-the-art of academic and infrastructural services with modern learning Resources to produce self- sustainable professionals..

DM_3 To inculcate the prominence of higher studies, research and entrepreneurship to pursue global standards.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: Comply with the contemporary trends and best practices of industry and research standards of Artificial Intelligence and Data Science.

PEO2: Develop Artificial Intelligence and Data Science based solutions to address diverse needs of the community for improving the quality of life and environment

PEO3: To produce creative and technically strong engineers with research pioneering solutions to meet global challenges

PEO4: Inculcate values of professional ethics, social concerns, environment protection

and life-long learning.

PROGRAMOUTCOMES(POs)

PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
PO3	**Design / Development of Solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
PO4	**Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
PO7	**Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one‟s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

COURSE OUTCOMES

C216.1	Implement data link layer farming methods and Analyze error detection and error correction codes.
C216.2	Implement and analyze routing and congestion issues in network design.
C216.3	Implement Encoding and Decoding techniques used in presentation layer.

COsMAPPINGWITHPOs&PSOs

Course Outcome	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
C216.1	2	2	3	2	2	-	-	-	-	1	-	-	1	2	1
C216.2	2	1	2	1	2	-	-	-	-	-	2	-	1	1	1
C216.3	1	2	1	2	1	-	-	--	-	-	-	-	2	1	1
C216	**1.6**	**1.6**	**2.0**	**1.6**	**1.6**	**-**	**-**	**-**	**-**	**0.3**	**0.6**	**-**	**1.3**	**1.3**	**1.16**

List of Experiments

SNO	PROGRAM
1	**Week 1-:**Implement the data link layer framing methods such as character, character-stuffing and bit stuffing.
2	**Week 2 -:**Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP
3	**Week 3 - :**Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.
4	**Week 4 - :**Implement Dijsktra's algorithm to compute the shortest path through a network
5	**Week 5- :** Take an example subnet of hosts and obtain a broadcast tree for the subnet.
6	**Week 6- :** Implement distance vector routing algorithm for obtaining routing tables at each node.
7	**Week 7- :** Implement data encryption and data decryption.
8	**Week 8- :** Write a program for congestion control using Leaky bucket algorithm.
9	**Week 9- :** Write a program for frame sorting technique used in buffers.
10	**Week 10- :** Wireshark i. Packet Capture Using Wire shark ii. Starting Wire shark iii. Viewing Captured Traffic iv. Analysis and Statistics & Filters.
11	**Week 11- :** How to run Nmap scan
12	**Week 12- :** Operating System Detection using Nmap
13	**Week 13- :** Do the following using NS2 Simulator i. NS2 Simulator-Introduction ii. Simulate to Find the Number of Packets Dropped iii. Simulate to Find the Number of Packets Dropped by TCP/UDP iv. Simulate to Find the Number of Packets Dropped due to Congestion v. Simulate to Compare Data Rate& Throughput. vi. Simulate to Plot Congestion for Different Source/Destination vii. Simulate to Determine the Performance with respect to Transmission of Packets

EXPERIMENT NO: 1. (a)

NAME OF THE EXPERIMENT: Bit Stuffing.
AIM: Implement the data link layer framing methods such as and bit stuffing. HARDWARE **REQUIREMENTS**: Intel based Desktop PC:- RAM of 512 MB SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
The new technique allows data frames to contain an arbitrary number if bits and allows character codes with an arbitrary no of bits per character. Each frame begins and ends with special bit pattern, 01111110, called a flag byte. Whenever the sender's data link layer encounters five consecutive one's in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data ALGORITHM:
Begin
Step 1: Read frame length n
Step 2: Repeat step (3 to 4) until i<n(: Read values in to the input frame (0's and 1's) i.e.
Step 3: initialize I i=0;
Step 4: read a[i] and increment i
Step 5: Initialize i=0, j=0,count =0
Step 6: repeat step (7 to 22) until i<n
Step 7: If a[i] == 1 then
Step 8: b[j] = a[i]
Step 9: Repeat step (10 to 18) until (a[k] =1 and k<n and count <5)
Step 10: Initialize k=i+1;
Step 11: Increment j and b[j]= a[k];
Step 12: Increment count ; Step 13: if count =5 then Step 14: increment j,
Step 15: b[j] =0
Step 16: end if
Step 17: i=k;
Step 18: increment k
Step 19: else
Step 20: b[j] = a[i]
Step 21: end if
Step 22: increment I and j
Step 23: print the frame after bit stuffing Step 24: repeat step (25 to 26) until i< j Step 25: print b[i]
Step 26: increment i
End

SOURCE CODE:

```
#include <stdio.h>
void main() {
   int a[20], b[30], i, j = 0, 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;
    while(i < n) {
        if(a[i] == 1) {
            count = 1;
            b[j] = a[i];
            for(k = i + 1; k < n && a[k] == 1 && count < 5; k++) {
              j++;
               b[j] = a[k];
               count++;

            if(count == 5) {
               j++;
               b[j] = 0;  // Bit stuffing
            }

            i = k;  // Move i to next bit after sequence
            count=1;
        }
        }
        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: 7
Enter input frame (0's & 1's only): 1
1
1
1
1
 0
 1
After stuffing the frame is: 11111001
```

VIVA QUESTIONS:

1. What is bit stuffing?
2. What is the use of bitstuffing?
3. with bit stuffing the boundary b/w 2 frames can be unambiguously recognized by ..
4. ----------is analogous to character stuffing
5. Each frame begins and ends with a special bit pattern 01111110 called ---
---
6. The senders data link layer encounters no of 1'sconsecutively

EXPERIMENT NO: 1. (b)

NAME OF THE EXPERIMENT: Character Stuffing.
AIM: Implement the data link layer framing methods such as character, character stuffing. HARDWARE **REQUIREMENTS**: Intel based Desktop PC:-RAM of 512 MB SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
The framing method gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and the sequence DLE ETX. If the destination ever losses the track of the frame boundaries all it has to do is look for DLE STX or DLE ETX characters to figure out. The data link layer on the receiving end removes the DLE before the data are given to the network layer. This technique is called characterstuffing
ALGORITHM:
Begin
Step 1: Initialize I and j as 0
Step 2: Declare n and pos as integer and a[20],b[50],ch as character
Step 3: read the string a
Step 4: find the length of the string n, i.e n-strlen(a)
Step 5: read the position, pos
Step 6: if pos > n then
Step 7: print invalid position and read again the position, pos
Step 8: end if
Step 9: read the character, ch
Step 10: Initialize the array b , b[0…5] as 'd', 'l', 'e', 's' ,'t' ,'x' respectively
Step 11: j=6;
Step 12: Repeat step[(13to22) until i<n
Step 13: if i==pos-1 then
Step 14: initialize b array,b[j],b[j+1]…b[j+6] as'd', 'l', 'e' ,'ch, 'd', 'l','e' respectively
Step 15: increment j by 7, i.e j=j+7
Step 16: end if
Step 17: if a[i]=='d' and a[i+1]=='l' and a[i+2]=='e' then Step 18: initialize array b, b[13…15]='d', 'l', 'e' respectively Step 19: increment j by 3, i.e j=j+3
Step 20: end if
Step 21: b[j]=a[i]
Step 22: increment I and j;
Step 23: initialize b array,b[j],b[j+1]…b[j+6] as'd', 'l','e' ,'e','t', 'x','\0' respectively
Step 24: print frame after stiuffing
Step 25: print b
End

SOURCE CODE:
//PROGRAM FOR CHARACTER STUFFING

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>  // for exit()
```

```
int main() {
    int i = 0, j = 0, n, pos;
    char a[20], b[100], ch;

    printf("Enter string:\n");
    scanf("%s", a);
    n = strlen(a);

    printf("Enter position:\n");
    scanf("%d", &pos);

    if (pos > n || pos < 1) {
        printf("Invalid position, exiting.\n");
        exit(1);
    }

    printf("Enter the character:\n");
    // Use space before %c to consume leftover newline from previous scanf
    scanf(" %c", &ch);  // REPLACES getche()

    // Frame starts with 'dlestx'
    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) {
            // Insert dle + ch + dle
            b[j++] = 'd';
            b[j++] = 'l';
            b[j++] = 'e';
            b[j++] = ch;
            b[j++] = 'd';
            b[j++] = 'l';
            b[j++] = 'e';
        }

        // Byte stuffing for 'dle' in input string
        if (a[i] == 'd' && a[i+1] == 'l' && a[i+2] == 'e') {
            b[j++] = 'd';
            b[j++] = 'l';
            b[j++] = 'e';
        }

        b[j++] = a[i];
```

```
        i++;
    }

    // Frame ends with 'dleetx'
    b[j++] = 'd';
    b[j++] = 'l';
    b[j++] = 'e';
    b[j++] = 'e';
    b[j++] = 't';
    b[j++] = 'x';
    b[j] = '\0';

    printf("\nFrame after stuffing:\n");
    printf("%s\n", b);

    return 0;
}
```

OUTPUT:

```
Enter string:
abcde
Enter position:
3
Enter the character:
t

Frame after stuffing:
Dlestxabdletdlecdedleetx
```

VIVA QUESTIONS:

What is character stuffing?
What is the use of characterstuffing?
_____is analogous to bit stuffing.
_____are the delimiters for character stuffing
Expand DLESTX____
Expand DLE ETX___

EXPERIMENT NO: 2.
NAME OF THE EXPERIMENT: Cyclic Redundancy Check.
AIM: Implement on a data set of characters the three CRC polynomials – CRC 12, CRC 16 and CRC CCIP.
HARDWARE REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
CRC method can detect a single burst of length n, since only one bit per column will be changed, a burst of length n+1 will pass undetected, if the first bit is inverted, the last bit is inverted and all other bits are correct. If the block is badly garbled by a long burst or by multiple shorter burst, the probability that any of the n columns will have the correct parity that is 0.5. so the probability of a bad block being expected when it should not be 2 power(-n). This scheme some times known as Cyclic Redundancy Code

ALGORITHM/FLOWCHART:

Begin

Step 1: Declare `i`, `j`, `data[20]`, `div[20]`, `temp[4]`, `total[100]`, `datalen`, `divlen`, `len`, `flag` as integers and character arrays
Step 2: Set `flag = 1`
Step 3: Prompt user and read `datalen` (length of data)
Step 4: Prompt user and read `divlen` (length of divisor)
Step 5: Compute `len = datalen + divlen - 1`
Step 6: Prompt user and read `data[]`
Step 7: Prompt user and read `div[]`

Step 8: Initialize `i = 0`
Step 9: Repeat Steps 10 to 11 until `i < datalen`
Step 10: Set `total[i] = data[i]`
Step 11: Increment `i`
Step 12: Repeat Steps 13 to 14 until `i < len`
Step 13: Set `total[i] = '0'`
Step 14: Increment `i`

Step 15: Call `check()` function to compute remainder
Step 16: Initialize `i = 0`
Step 17: Repeat Steps 18 to 19 until `i < divlen`
Step 18: Set `temp[i + datalen] = data[i]`
Step 19: Increment `i`
Step 20: Display `temp[]` as the transmitted code word

Step 21: Prompt user to enter received code word → store in `total[]`
Step 22: Call `check()` function again using received code

Step 23: Initialize `i = 0`
Step 24: Repeat Steps 25 to 27 until `i < divlen - 1`
Step 25: If `data[i] != '0'`, then
Step 26: Set `flag = 0` and break
Step 27: Increment `i`

Step 28: If `flag == 1`, then print `"successful!!"`
Step 29: Else, print `"received code word contains errors"`
End

Function: check()

Begin

Step 1: Initialize `j = 0`
Step 2: Repeat Steps 3 to 4 until `j < divlen`
Step 3: Set `data[j] = total[j]`
Step 4: Increment `j`

Step 5: While `j <= len`, do

Step 6: If `data[0] == '1'` then

Step 7: For `i = 1` to `divlen - 1`, do

```
data[i] = (data[i] == div[i]) ? '0' : '1'
```

Step 8: For `i = 0` to `divlen - 2`, do

```
data[i] = data[i+1]
```

Step 9: Set `data[divlen-1] = total[j]`

Step 10: Increment `j`
End While

End Function

SOURCE CODE:
//PROGRAM FOR CYCLIC REDUNDENCY CHECK

```
#include<stdio.h>
char data[20],div[20],temp[4],total[100];
int i,j,datalen,divlen,len,flag=1;
void check();
int main()
{
        printf("Enter the total bit of data:");
        scanf("%d",&datalen);
        printf("\nEnter the total bit of divisor");
        scanf("%d",&divlen);
        len=datalen+divlen-1;
        printf("\nEnter the data:");
        scanf("%s",&data);
```

```
   printf("\nEnter the divisor");
   scanf("%s",div);

        for(i=0;i<datalen;i++)
        {
                total[i]=data[i];
                temp[i]=data[i];
        }
        for(i=datalen;i<len;i++)
      total[i]='0';
        check();
        for(i=0;i<divlen;i++)
                temp[i+datalen]=data[i];
        printf("\ntransmitted Code Word:%s",temp);
        printf("\n\nEnter the received code word:");
   scanf("%s",total);
        check();
        for(i=0;i<divlen-1;i++)
                if(data[i]=='1')
                {
                        flag=0;
                        break;
                }
        if(flag==1)
        printf("\nsuccessful!!");
        else
        printf("\nreceived code word contains errors...\n");
}
void check()
{
   for(j=0;j<divlen;j++)
      data[j]=total[j];
   while(j<=len)
        {
      if(data[0]=='1')
         for(i = 1;i <divlen ; i++)
                                data[i] = (( data[i] == div[i])?'0':'1');
      for(i=0;i<divlen-1;i++)
         data[i]=data[i+1];
      data[i]=total[j++];
   }
}
```

OUTPUT:

Enter the total bit of data:7

Enter the total bit of divisor4

Enter the data:1001010

Enter the divisor1011

transmitted Code Word:1001010111

Enter the received code word:1001010111

successful!!

Enter the total bit of data:7

Enter the total bit of divisor4

Enter the data:1001010

Enter the divisor1011

transmitted Code Word:1001010111

Enter the received code word:1001010110

received code word contains errors...

VIVA QUESTIONS:
What is CRC?
What is the use ofCRC?
Name the CRC standards
Define checksum?
Define generator polynomial?
Polynomial arithmetic is doneby____

EXPERIMENT NO: 3

NAME OF THE EXPERIMENT: flow control using the sliding window protocol
AIM:Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism.

ALGORITHM/FLOWCHART:
Begin
Step 1: Declare w, i, f, frames[50] as integers
Step 2: Prompt and read w → window size
Step 3: Prompt and read f → number of frames
Step 4: If w > f then
Step 5: Print "Window size should be less than frame count"
Step 6: Prompt and re-read f
Step 7: Prompt user to input f frame values
Step 8: Initialize i = 1
Step 9: Repeat Steps 10–11 until i <= f
Step 10: Read frames[i]
Step 11: Increment i
Step 12: Print transmission description:
"Frames will be sent assuming no corruption... Sender waits for ACK after sending w frames."
Step 13: Initialize i = 1
Step 14: Repeat Steps 15–20 until i <= f
Step 15: If i % w == 0 then
Step 16: Print frames[i]
Step 17: Print "Acknowledgement of above frames received"
Step 18: Else
Step 19: Print frames[i] (in same line)
Step 20: Increment i
Step 21: If f % w != 0 then
Step 22: Print "Acknowledgement of above frames sent is received"
End

Source code:

```
#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);
if(w>f)
{
   printf("\n window size should be less than frames count\n\n Re-Enter 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: 2

 window size should be less than frames count

 Re-Enter number of frames to transmit: 4

Enter 4 frames: 1 2 3 4

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

1 2 3
Acknowledgement of above frames sent is received by sender

4
Acknowledgement of above frames sent is received by sender


=== Code Execution Successful ===
```

VIVA QUESTIONS:

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

EXPERIMENT NO: 4

NAME OF THE EXPERIMENT: Shortest Path.
AIM: Implement Dijkstra's algorithm to compute the Shortest path thru a given graph.
HARDWARE REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.

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

SOURCE CODE:

```
//**********************************
// .PROGRAM FOR FINDING SHORTEST //PATH FOR A GIVEN GRAPH
//**********************************
#include<stdio.h>

#define INFINITY 9999
#define MAX 50
```

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

int main() {
   int G[MAX][MAX], i, j, n, u, flag = 0;

   printf("Graph: Shortest Path to Other Vertices: Dijkstra Algorithm >>\n\n");
   printf("Enter Number of Vertices Present in the Graph: ");
   scanf("%d", &n);

   printf("\nEnter the Adjacency Matrix:\n");
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         scanf("%d", &G[i][j]);

   printf("\nEnter The Starting Node: ");
   scanf("%d", &u);


   dijkstra(G, n, u);


   return 0;
}

void dijkstra(int G[MAX][MAX], int n, int startnode) {
   int cost[MAX][MAX], distance[MAX], pred[MAX];
   int visited[MAX], count, mindistance, nextnode, i, j;

   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];

   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;
      for (i = 0; i < n; i++)
         if (distance[i] < mindistance && !visited[i]) {
```

```
            mindistance = distance[i];
            nextnode = i;
         }

      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++;
   }

   for (i = 0; i < n; i++) {
      if (i != startnode) {
         if (distance[i] == 9999) {
            printf("\nThere is no Possible Path Between %d and %d.", i, startnode);
         } else {
            printf("\nDistance of Node %d to %d is: %d", i, startnode, distance[i]);
            printf("\nAnd the Path is: %d ", i);
            j = i;
            do {
               j = pred[j];
               printf(" -> %d", j);
            } while (j != startnode);
         }
      }
      printf("\n");
   }
}
```

OUTPUT:

Graph: Shortest Path to Other Vertices: Dijkstra Algorithm >>

Enter Number of Vertices Present in the Graph: 5

Enter the Adjacency Matrix:
0 3 5 0 0
3 0 2 1 7
5 2 0 4 0
0 1 4 0 6
0 7 0 6 0

Enter The Starting Node: 4

Distance of Node 0 to 4 is: 10
And the Path is: 0 -> 1 -> 4

Distance of Node 1 to 4 is: 7
And the Path is: 1 -> 4

Distance of Node 2 to 4 is: 9
And the Path is: 2 -> 1 -> 4

Distance of Node 3 to 4 is: 6
And the Path is: 3 -> 4

VIVA QUESTIONS:

What is Dijkstra Algorithm?
Can you give me some examples where you would use Dijkstra's algorithm?
What are some alternative ways to find shortest paths?
Is it possible to use Dijkstra's algorithm for directed graphs?
What are some applications of Dijkstra's algorithm?

EXPERIMENT NO: 5

NAME OF THE EXPERIMENT: Broadcast Tree.
AIM: Implement broadcast tree for a given subnet of hosts
HARDWARE REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers the algorithm just finds the broadcast between them on thegraph.

ALGORITHM/FLOWCHART:

Step 1: Declare `w`, `i`, `f`, `frames[50]` as integers
Step 2: Prompt and read `w` → window size
Step 3: Prompt and read `f` → number of frames
Step 4: If `w > f` then
Step 5: Print `"Window size should be less than frame count"`
Step 6: Prompt and re-read `f`
Step 7: Prompt user to input `f` frame values
Step 8: Initialize `i = 1`
Step 9: Repeat Steps 10–11 until `i <= f`
Step 10: Read `frames[i]`
Step 11: Increment `i`
Step 12: Print transmission description:
"Frames will be sent assuming no corruption... Sender waits for ACK after sending w frames."
Step 13: Initialize `i = 1`
Step 14: Repeat Steps 15–20 until `i <= f`
Step 15: If `i % w == 0` then
Step 16: Print `frames[i]`
Step 17: Print `"Acknowledgement of above frames received"`
Step 18: Else
Step 19: Print `frames[i]` (in same line)
Step 20: Increment `i`

Step 21: If `f % w != 0` then
Step 22: Print `"Acknowledgement of above frames sent is received"`

End

SOURCE CODE:
// Write a 'c' program for Broadcast tree from subnet of host #include

```
#include <stdio.h>

#define MAX 50
#define INF 99

int edge[MAX][MAX], parent[MAX];
int n;

// Find function with path compression
int find(int x) {
   while (parent[x] >= 0)
      x = parent[x];
   return x;
}

// Union function
void sunion(int x, int y) {
   parent[x] = y;
}

int main() {
   int i, j, u, v, min, p, q, edges = 0, mincost = 0;
   int t[MAX][3];  // Stores u, v, weight of each MST edge

   printf("Enter the number of nodes: ");
   scanf("%d", &n);

   printf("Enter the adjacency matrix (use 99 for no edge):\n");
   for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++) {
         scanf("%d", &edge[i][j]);
      }
   }

   // Initialize parent array for union-find
   for (i = 0; i < n; i++) {
      parent[i] = -1;
   }

   // Main loop to find n-1 edges
   while (edges < n - 1) {
      min = INF;
      u = v = -1;

      // Find the minimum edge
      for (i = 0; i < n; i++) {
         for (j = 0; j < n; j++) {
            if (edge[i][j] < min) {
               min = edge[i][j];
```

```
                u = i;
                v = j;
            }
        }
    }

    // If no edge is found, break
    if (u == -1 || v == -1) break;

    // Find roots of the nodes
    p = find(u);
    q = find(v);

    // If adding this edge doesn't form a cycle
    if (p != q) {
        t[edges][0] = u;
        t[edges][1] = v;
        t[edges][2] = edge[u][v];  // Store the weight before modifying the matrix
        mincost += edge[u][v];
        sunion(p, q);
        edges++;
    }

    // Mark this edge as visited
    edge[u][v] = edge[v][u] = INF;
}

// Print the result
printf("\nMinimum cost is %d\n", mincost);
printf("Minimum spanning tree is:\n");
for (i = 0; i < edges; i++) {
    printf("%c - %c : %d\n", t[i][0] + 65, t[i][1] + 65, t[i][2]);
}

return 0;
}
```

OUTPUT:

```
Enter the number of nodes: 3
Enter the adjacency matrix (use 99 for no edge):
0 1 2
1 0 2
2 2 0

Minimum cost is 3
Minimum spanning tree is:
A - B : 1
A - C : 2
```

VIVA QUESTIONS:

What is spanning tree
What is broad casttree?
What are the advantages of broad casttree?
Where we should use the broad casttree
What is flooding?
What is the subnet?

EXPERIMENT NO: 6

NAME OF THE EXPERIMENT: Distance Vector routing.
AIM: Obtain Routing table at each node using distance vector routing algorithm for a given subnet.
HARDWARE REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
Distance Vector Routing Algorithms calculate a best route to reach a destination based solely on distance. E.g. RIP. RIP calculates the reach ability based on hop count. It's different from link state algorithms which consider some other factors like bandwidth and other metrics to reach a destination.
Distance vector routing algorithms are not preferable for complex networks and take longer to converge.
ALGORITHM:
Begin
Step1: Create struct node unsigned dist[20],unsigned from[20],rt[10]
Step2: initialize int dmat[20][20], n,i,j,k,count=0,
Step3: write "the number of nodes "Step4: read the number of nodes "n"Step5: write" the cost matrix :"Step6: intialize i=0 Step7: repeat until i<n Step8: increment i Step9: initialize j=0
Step10: repeat Step(10-16)until j<n
Step11: increment j Step12:read dmat[i][j] Step13:intialize dmat[i][j]=0
Step14:intialize rt[i].dist[j]=dmat[i][j] Step15:intialize rt[i].from[j]=j Step16:end
Step17:start do loop Step (17-33)until Step18:intilialize count =0 Step19:initialize i=0
Step20:repeat until i<n Step21:increment i Step22:initialize j=0 Step23:repeat until j<n Step24:increment j
Step25:initialize k=0 Step26:repeat until k<n Step27:increment k
Step28:if repeat Step(28-32) until rt[i].dist[j]>dmat[i][k]+rt[k].dist[j]
Step29:intialize rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]
Step30:intialize rt[i].from[j]=k; Step31:increment count Step32:end if
Step33:end do stmt Step34:while (count!=0) Step35:initialize i=0
Step36:repeat Steps(36-44)until i<n
Step37:increment i
Step38:write ' state values for router',i+1
Step39:initialize j=0
Step40:repeat Steps (40-43)until j<n
Step41:increment j
Step42:write 'node %d via %d distance % ',j+1,rt[i].from[j]+1,rt[i].dist[j]
Step43:end Step44:end end

SOURCE CODE:

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

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

OUTPUT:

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

VIVA QUESTIONS:

What isrouting
What is best algorithm among all routingalgorithms?
What is static routing?
Difference between static and dynamic
How distance vector routingworks
What is optimality principle?

EXPERIMENT NO: 7

NAME OF THE EXPERIMENT: data encryption and data decryption.
AIM: Take a 64 bit playing text and encrypt the same using DES algorithm. HARDWARE
REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB SOFTWARE REQUIREMENTS: Turbo C / Borland C.
THEORY:
Data encryption standard was widely adopted by the industry in security products. Plain text is encrypted in blocks of 64 bits yielding 64 bits of cipher text. The algorithm which is parameterized by a 56 bit key has 19 distinct stages. The first stage is a key independent transposition and the last stage is exactly inverse of the transposition. The remaining stages are functionally identical but are parameterized by different functions of the key. The algorithm has been designed to allow decryption to be done with the same key as encryption

Begin
Step 1: Declare
`pwd[20]` → character array for password
`alpha[26] = "abcdefghijklmnopqrstuvwxyz"`
`num[20]` → integer array to store numeric values of characters
Integers: `i`, `n`, `key`
Step 2: Prompt user to enter a password
Step 3: Read `pwd`
Step 4: Compute password length → `n = strlen(pwd)`
Step 5: For `i = 0` to `n - 1`, repeat:
Convert `pwd[i]` to lowercase
Convert to ASCII → subtract `'a'`
Store result in `num[i]`
Step 6: Prompt user to enter encryption key
Step 7: Read `key`
Step 8: For `i = 0` to `n - 1`, repeat:
Compute `num[i] = (num[i] + key) % 26`
Step 9: For `i = 0` to `n - 1`, repeat:
Map `num[i]` back to character using `alpha[]`
Store result in `pwd[i]`
Step 10: Print the key and encrypted text
Step 11: For `i = 0` to `n - 1`, repeat:
Compute `num[i] = (num[i] - key) % 26`
If `num[i] < 0`, then `num[i] = num[i] + 26`
Step 12: For `i = 0` to `n - 1`, repeat:
Map `num[i]` back to character using `alpha[]`
Store result in `pwd[i]`
Step 13: Print decrypted text
End

SOURCE CODE

```
#include<stdio.h>
 #include<conio.h>
#include<string.h>
#include<ctype.h>
 void main()
```

```
{
char pwd[20];
char alpha[26]="abcdefghijklmnopqrstuvwxyz";  int num[20],i,n,key;
//clrscr();
printf("\nEnter the password:"); scanf("%s",&pwd); n=strlen(pwd);   for(i=0;i<n;i++)
num[i]=toascii(tolower(pwd[i]))-'a';  printf("\nEnter the key:"); scanf("%d",&key);
for(i=0;i<n;i++)
num[i]=(num[i]+key)%26; for(i=0;i<n;i++)
pwd[i]=alpha[num[i]]; printf("\nThe key is:%d",key); printf("\nEncrypted text is:%s",pwd); for(i=0;i<n;i++)
{
num[i]=(num[i]-key)%26; if(num[i]<0)
num[i]=26+num[i]; pwd[i]=alpha[num[i]];
}
printf("\nDecrypted text is:%s",pwd); getch();
}
```

OUTPUT:

```
Enter the password:puji



Enter the key:6



The key is:6

Encrypted text is:vapo

Decrypted text is:puji
```

VIVA QUESTIONS:

ExpandDES__
What is cipher text?
What is plaintext?
Define publickey?
Define encryption?
Substitutions are performed by_____boxes

EXPERIMENT NO: 8

NAME OF THE EXPERIMENT:congestion control using Leaky bucket algorithm
AIM:Implement a program for congestion control using Leaky bucket algorithm. HARDWARE **REQUIREMENTS:** Intel based Desktop PC:- RAM of 512 MB SOFTWARE REQUIREMENTS: Turbo C / Borland C.

Program

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

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

OUTPUT:

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

Enter the incoming packet size : 40

Incoming packet size 40

Bucket buffer size 40 out of 50

After outgoing 0 packets left out of 50 in buffer

Enter the incoming packet size : 20

Incoming packet size 20

Bucket buffer size 20 out of 50

After outgoing -
20 packets left out of 50 in buffer

Enter the incoming packet size : 60

Incoming packet size 60

Bucket buffer size 40 out of 50

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

VIVA QUESTIONS:

What is Congestion?
What is Congestion Control?
What are Congestion control algorithms?
Differentiate between Leaky bucket and Token bucket?
What is the purpose of leaky bucket algorithm?

EXPERIMENT NO: 9

NAME OF THE EXPERIMENT:frame sorting technique used in buffers.
AIM:Implement frame sorting technique used in buffers.
HARDWARE REQUIREMENTS: Intel based Desktop PC:- RAM of 512 MB
SOFTWARE REQUIREMENTS: Turbo C / Borland C.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define FRAME_SIZE 3
#define MAX_FRAMES 127

struct frame {
   char text[FRAME_SIZE + 1]; // +1 for '\0'
   int seq_no;
};

int main() {
   char str[FRAME_SIZE * MAX_FRAMES];
   struct frame fr[MAX_FRAMES], temp;
   int i, j, k = 0, len;

   printf("Enter the message: ");
   fgets(str, sizeof(str), stdin);
   len = strlen(str);

   if (str[len - 1] == '\n') str[len - 1] = '\0';

   // Split into frames and assign sequence numbers
   i = 0;
   while (i < strlen(str)) {
      fr[k].seq_no = k;
      for (j = 0; j < FRAME_SIZE && str[i] != '\0'; j++)
         fr[k].text[j] = str[i++];
      fr[k].text[j] = '\0'; // null terminate
      k++;
   }

   int no_frames = k;

   // Display frames
   printf("\nFrames with sequence numbers:\n");
   for (i = 0; i < no_frames; i++)
      printf("%d:%s ", fr[i].seq_no, fr[i].text);
```

```
    // Shuffle using Fisher-Yates shuffle
    srand(time(NULL));
    for (i = no_frames - 1; i > 0; i--) {
        int r = rand() % (i + 1);
        temp = fr[i];
        fr[i] = fr[r];
        fr[r] = temp;
    }

    printf("\n\nAfter shuffling:\n");
    for (i = 0; i < no_frames; i++)
        printf("%d:%s ", fr[i].seq_no, fr[i].text);

    // Sort frames by sequence number (Bubble Sort)
    for (i = 0; i < no_frames - 1; i++) {
        for (j = 0; j < no_frames - 1 - i; j++) {
            if (fr[j].seq_no > fr[j + 1].seq_no) {
                temp = fr[j];
                fr[j] = fr[j + 1];
                fr[j + 1] = temp;
            }
        }
    }

    printf("\n\nAfter sorting (Reconstructed Message):\n");
    for (i = 0; i < no_frames; i++)
        printf("%s", fr[i].text);
    printf("\n");

    return 0;
}
```

OUTPUT:

Enter the message: welcometocnlab

Frames with sequence numbers:
0:wel 1:com 2:eto 3:cnl 4:ab

After shuffling:
0:wel 2:eto 3:cnl 1:com 4:ab

After sorting (Reconstructed Message):
welcometocnlab

VIVA QUESTIONS:

What is frame sorting?
What is purpose of buffering?
What is buffer technique?
What is buffer size in computer?
What is the purpose of sorting?

EXPERIMENT NO: 10

NAME OF THE EXPERIMENT: Wireshark
Packet Capture Using Wire shark
Starting Wire shark
Viewing Captured Traffic
Analysis and Statistics & Filters.

How does Wireshark work?

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

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

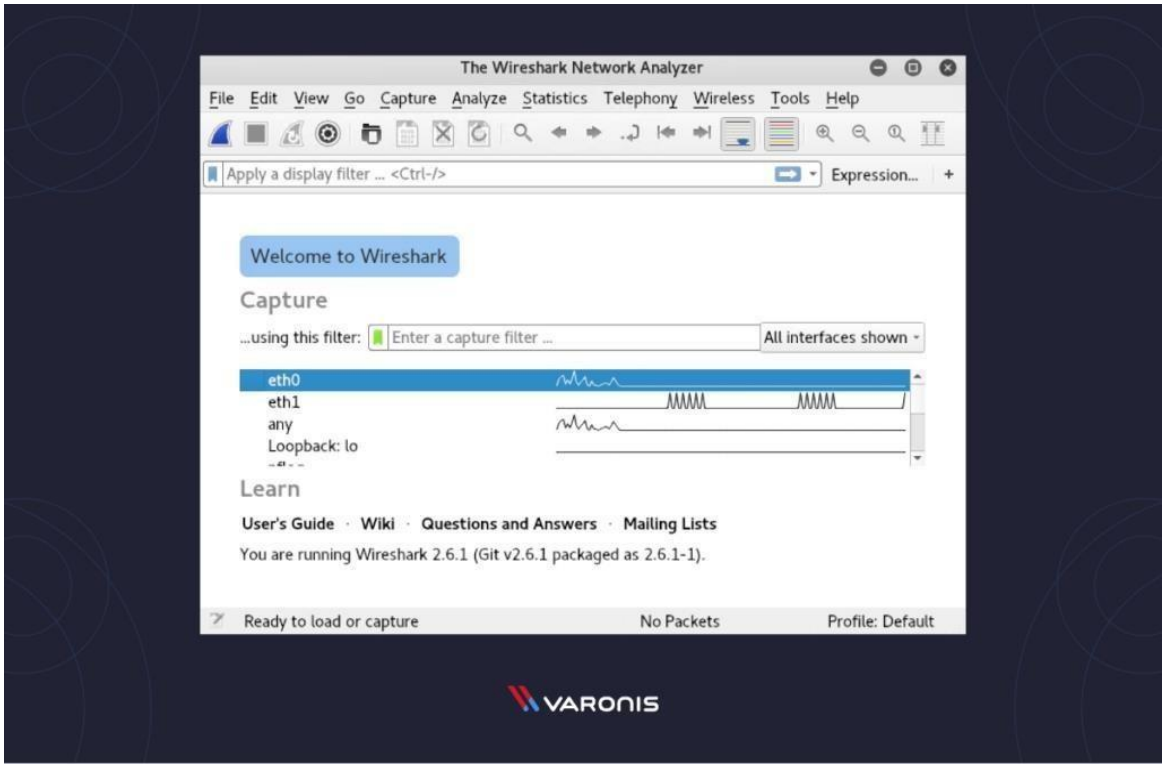
Wireshark allows you to filter the log either before the capture starts or during analysis, so you can narrow down and zero into what you are looking for in the network trace. For example, you can set a filter to see TCP traffic between two IP addresses. You can set it only to show you the packets sent from one computer. The filters in Wireshark are one of the primary reasons it became the standard tool for packet analysis.

Wireshark for Windows

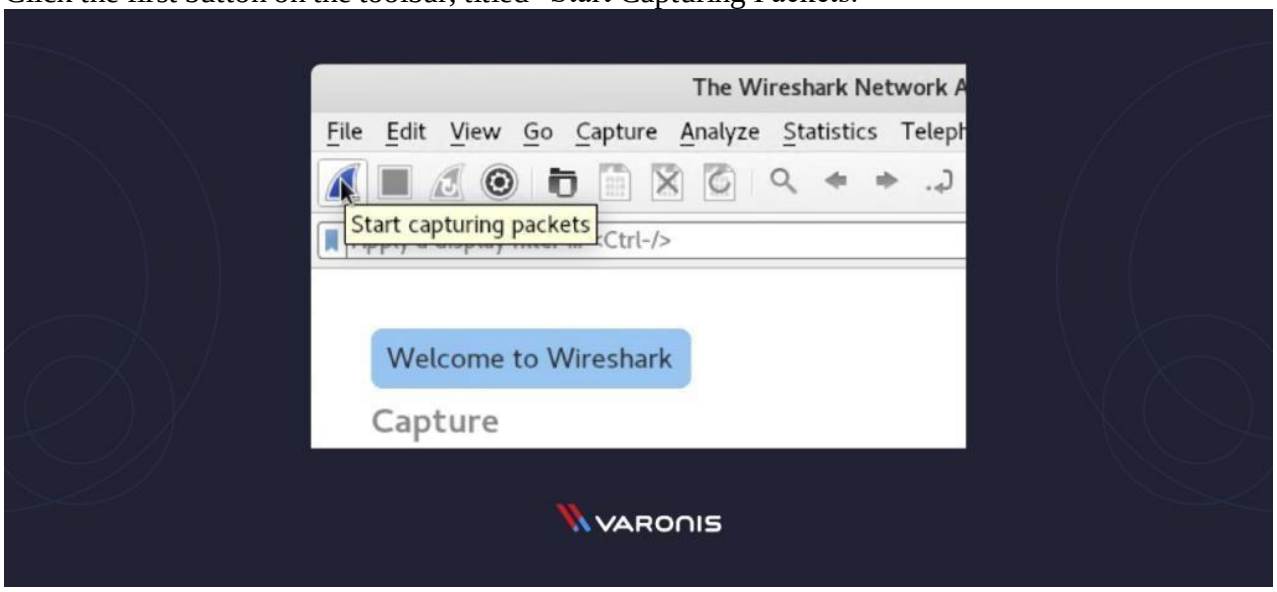
Wireshark comes in two flavors for Windows, 32 bit and 64 bit. Pick the correct version for your OS. The current release is 3.0.3 as of this writing. The installation is simple and shouldn't cause any issues.

Capturing Data Packets on Wireshark

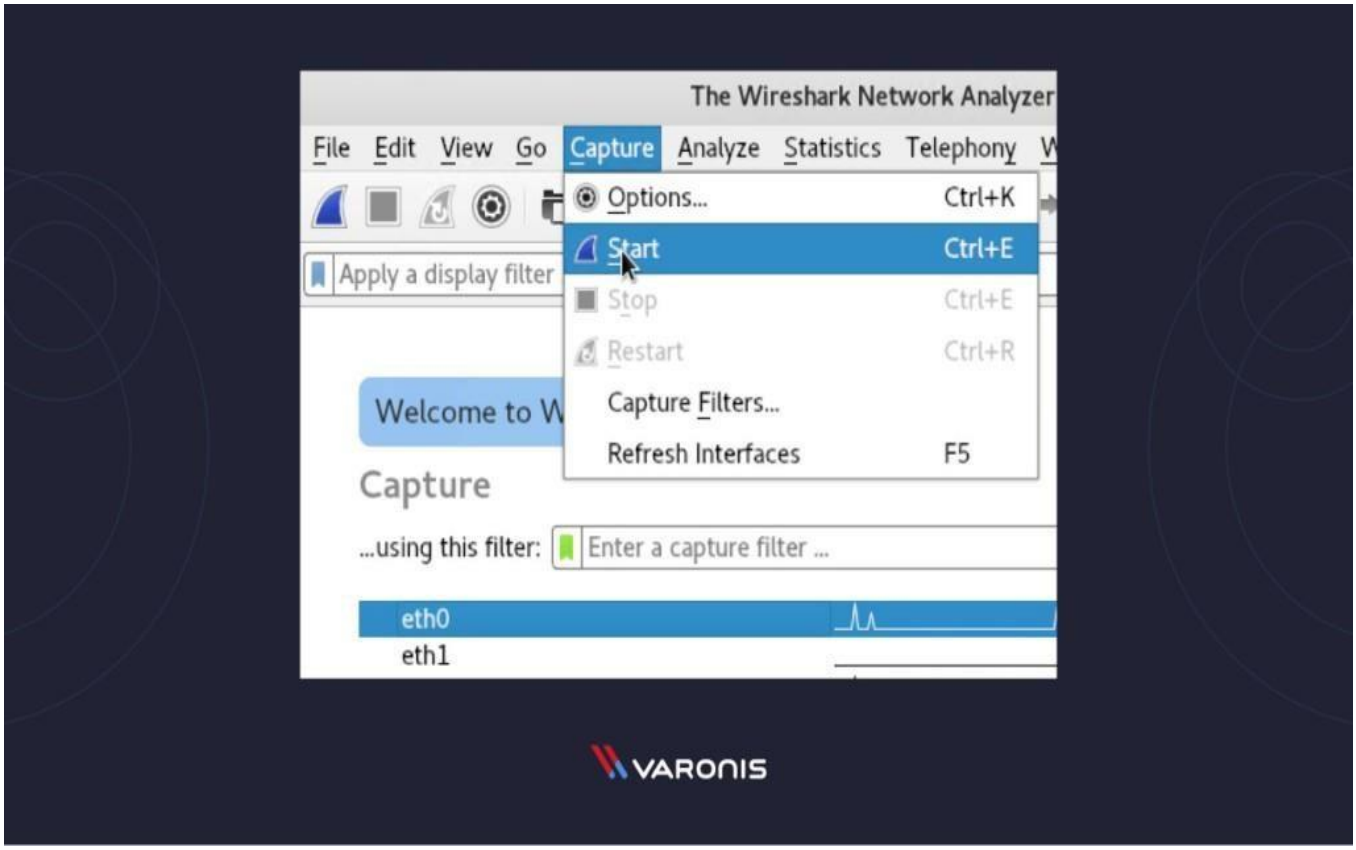
When you open Wireshark, you see a screen that shows you a list of all of the network connections you can monitor. You also have a capture filter field, so you only capture the network traffic you want to see.



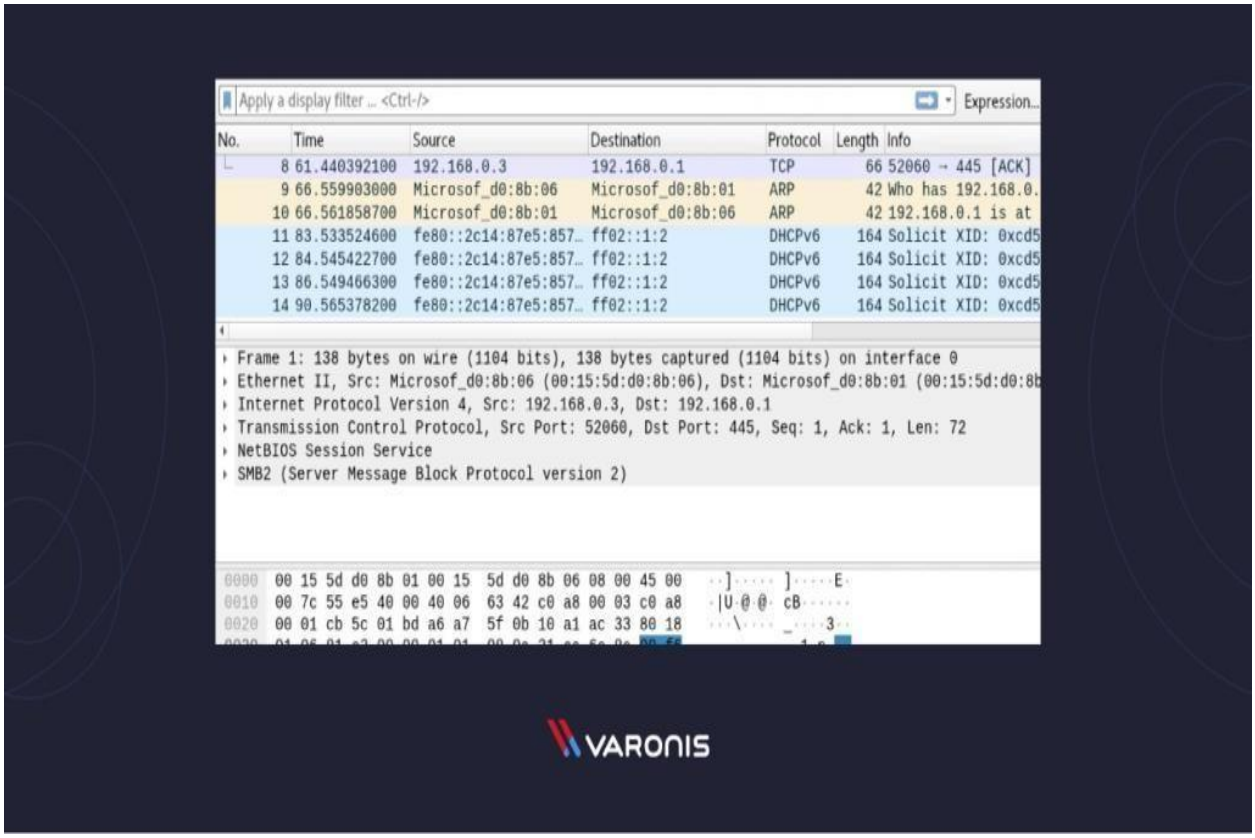
You can select one or more of the network interfaces using “shift left- click.” Once you have the network interface selected, you can start the capture, and there are several ways to do that.
Click the first button on the toolbar, titled “Start Capturing Packets.”



You can select the menu item Capture -> Start.



Or you could use the keystroke Control – E.
During the capture, Wireshark will show you the packets that it captures in real-time.



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

EXPERIMENT NO: 11,12

How to run Nmap scan

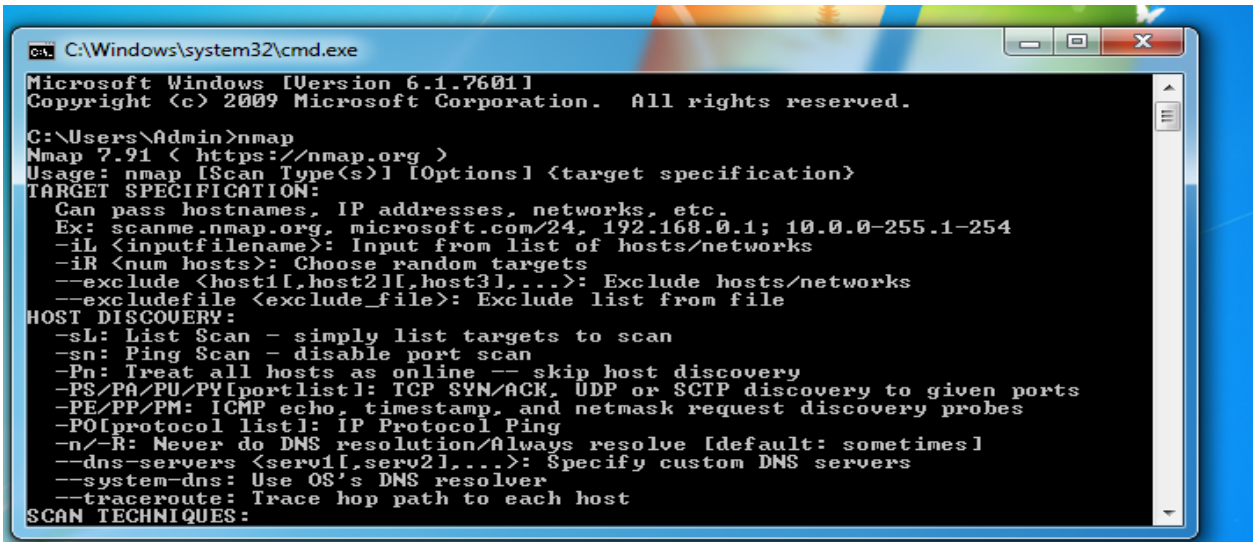
Operating System Detection using Nmap

Nmap builds on previous network auditing tools to provide quick, detailed scans of network traffic. It works by using IP packets to identify the hosts and IPs active on a network and then analyze these packets to provide information on each host and IP, as well as the operating systems they are running.
network administrators to scan for:

Open ports and services
Discover services along with their versions
Guess the operating system running on a target machine
Get accurate packet routes till the target machine
Monitoring hosts

Latest stable command-line zipfile: nmap-7.91-win32.zip Install above nmap in c:/programfiles
Go for computer properties->advanced system settings-> environment variables-> In user variables for admin click on new button
New window will be open In that variable name=path
Value =C:\Program Files\nmap-7.91

Take command prompt:
Nmap



OS Scanning

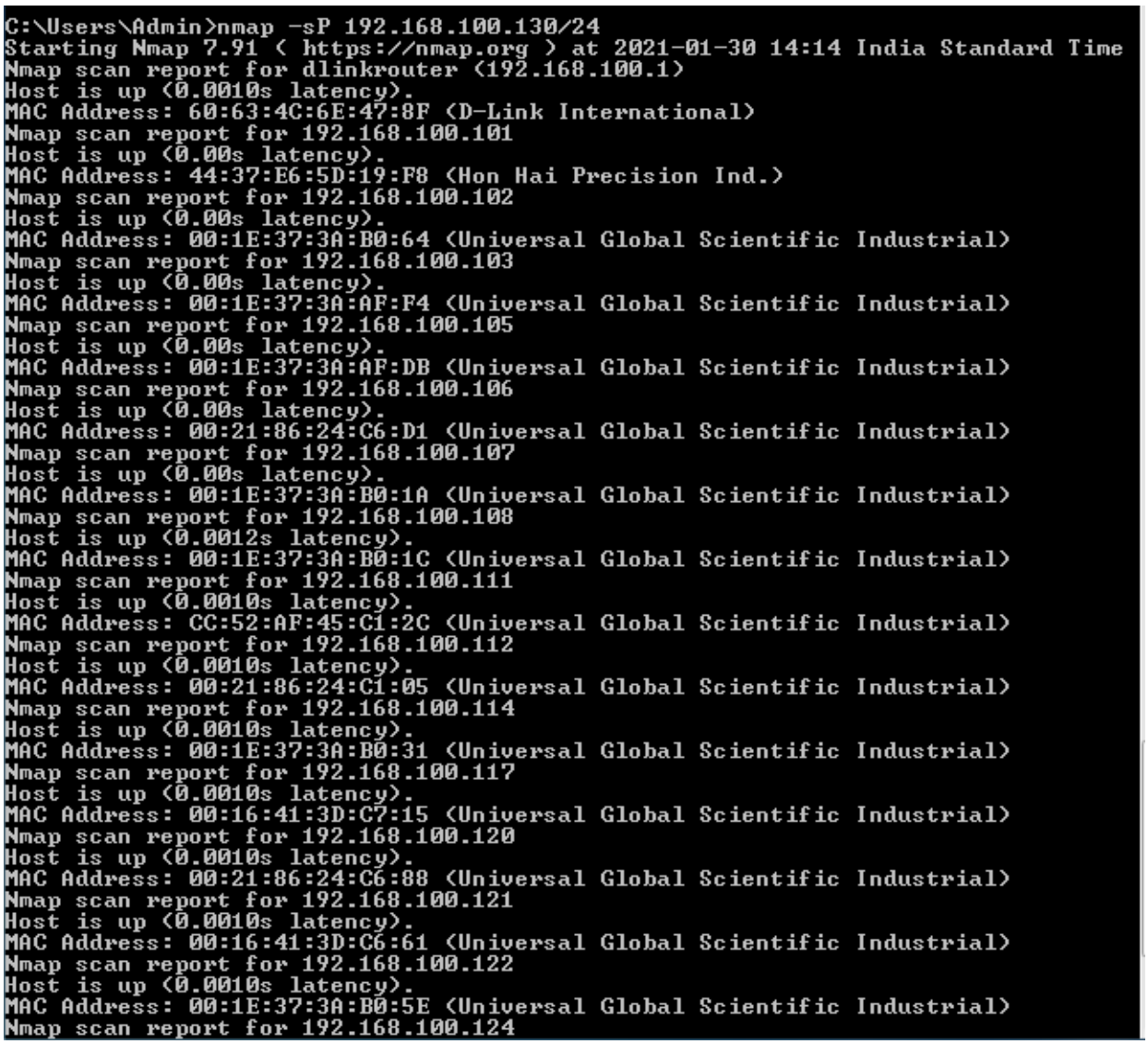
OS scanning is one of the most powerful features of Nmap. When using this type of scan, Nmap sends TCP and UDP packets to a particular port, and then analyze its response. It compares this response to a database of 2600 operating systems, and return information on the OS (and version) of a host.

To run an OS scan, use the following command:

```
C:\Users\Admin>nmap -O 192.168.100.130
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-30 14:02 India Standard Time
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn

Nmap done: 1 IP address (0 hosts up) scanned in 1.88 seconds

C:\Users\Admin>nmap -O 192.168.100.130/24
Starting Nmap 7.91 ( https://nmap.org ) at 2021-01-30 14:04 India Standard Time
Nmap scan report for dlinkrouter (192.168.100.1)
Host is up (0.00087s latency).
Not shown: 996 closed ports
PORT    STATE    SERVICE
21/tcp  open     ftp
23/tcp  filtered telnet
80/tcp  open     http
443/tcp filtered https
MAC Address: 60:63:4C:6E:47:8F (D-Link International)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.23 - 2.6.38
Network Distance: 1 hop

Nmap scan report for 192.168.100.101
Host is up (0.0012s latency).
All 1000 scanned ports on 192.168.100.101 are closed
MAC Address: 44:37:E6:5D:19:F8 (Hon Hai Precision Ind.)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.100.102
Host is up (0.0011s latency).
All 1000 scanned ports on 192.168.100.102 are closed
MAC Address: 00:1E:37:3A:B0:64 (Universal Global Scientific Industrial)
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

Nmap scan report for 192.168.100.103
Host is up (0.0011s latency).
```

nmap -O <target IP>

Running nmap:
To find all active hosts on network is Nmap –sP ipaddress


C:\Users\Admin>nmap -sP 192.168.100.130/24
Starting Nmap 7.91 (https://nmap.org) at 2021-01-30 14:14 India Standard Time
Nmap scan report for dlinkrouter (192.168.100.1)
Host is up (0.0010s latency).
MAC Address: 60:63:4C:6E:47:8F (D-Link International)
Nmap scan report for 192.168.100.101
Host is up (0.00s latency).
MAC Address: 44:37:E6:5D:19:F8 (Hon Hai Precision Ind.)
Nmap scan report for 192.168.100.102
Host is up (0.00s latency).
MAC Address: 00:1E:37:3A:B0:64 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.103
Host is up (0.00s latency).
MAC Address: 00:1E:37:3A:AF:F4 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.105
Host is up (0.00s latency).
MAC Address: 00:1E:37:3A:AF:DB (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.106
Host is up (0.00s latency).
MAC Address: 00:21:86:24:C6:D1 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.107
Host is up (0.00s latency).
MAC Address: 00:1E:37:3A:B0:1A (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.108
Host is up (0.0012s latency).
MAC Address: 00:1E:37:3A:B0:1C (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.111
Host is up (0.0010s latency).
MAC Address: CC:52:AF:45:C1:2C (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.112
Host is up (0.0010s latency).
MAC Address: 00:21:86:24:C1:05 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.114
Host is up (0.0010s latency).
MAC Address: 00:1E:37:3A:B0:31 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.117
Host is up (0.0010s latency).
MAC Address: 00:16:41:3D:C7:15 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.120
Host is up (0.0010s latency).
MAC Address: 00:21:86:24:C6:88 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.121
Host is up (0.0010s latency).
MAC Address: 00:16:41:3D:C6:61 (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.122
Host is up (0.0010s latency).
MAC Address: 00:1E:37:3A:B0:5E (Universal Global Scientific Industrial)
Nmap scan report for 192.168.100.124

EXPERIMENT 13 : Do the following using NS2 Simulator

i. NS2 Simulator-Introduction

Network simulator 2:

Network Simulator (NS) is simply a discrete event-driven network simulation tool for studying the dynamic nature of communication networks. Network Simulator 2 (NS2) provides substantial support for simulation of different protocols over wired and wirelessnetworks. It provides a highly modular platform for wired and wireless simulations supporting different network elements, protocols, traffic, and routing types [20].
NS2 is a simulation package that supports several network protocols including TCP, UDP, HTTP, and DHCP and these can be modeled using this package [21]. In addition,several kinds of network traffic types such as constant bit rate (CBR), available bit rate (ABR), and variable bit rate (VBR) can be generated easily using this package. It isa very popular simulation package in academic environments.
NS2 has been developed using the C++ programming language and OTcl. OTcl is a relatively new language that uses object-oriented aspects. It was developed at MIT asan object-oriented extension of the Tool command language (Tcl).

Simulate to Find the Number of Packets Dropped
As the first step, we create the simulator object.

```
# create a simulator object set ns [new Simulator]
```

Set up the network topology by creating node objects, and connecting the nodeswith link objects. If the output queue of a router is implemented as a part of a link, we need to specify the queue type. In this project, DropTail queue is used.In some cases, we may define the layout of the network topology for better NAM display.

```
# 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
```

Define traffic patterns by creating agents, applications and flows. In NS2, packets are always sent from one agent to another agent or a group of agents. Inaddition, we need to associate these agents with nodes.

```
# TCP traffic source
# create a TCP agent and attach it to node node1set tcp [new Agent/TCP]
$ns attach-agent $node1 $tcp
$tcp set fid_ 1
```

```
$tcp set class_ 1
# window_ * (packetsize_ + 40) / RTT
$tcp set window_ 30
$tcp set packetSize_ $packetSize

# create a TCP sink agent and attach it to node node4set sink [new Agent/TCPSink]
$ns attach-agent $node4 $sink

# connect both agents
$ns connect $tcp $sink

# create an FTP source "application";set ftp [new Application/FTP]
$ftp attach-agent $tcp
# UDP traffic source
# create a UDP agent and attach it to node 2set udp [new Agent/UDP]
$udp set fid_ 2 # red color
$ns attach-agent $node2 $udp
# create a CBR traffic source and attach it to udp set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ $packetSize
$cbr set rate_ 0.25Mb
$cbr set random_ false
$cbr attach-agent $udp
# creat a Null agent (a traffic sink) and attach it to node 4set null [new Agent/Null]
$ns attach-agent $node4 $null
$ns connect $udp $null
```

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

```
# open the nam trace file
set nam_trace_fd [open tcp_tahoe.nam w]
$ns  namtrace-all
$nam_trace_fd set trace_fd [open tcp_tahoe.tr w]

#Define a 'finish' procedureproc finish {}
{
global ns nam_trace_fd trace_fd

# close the nam trace file
$ns flush-trace close
$nam_trace_fd

# execute nam on the trace fileexit 0
}
```

Schedule the simulation by defining the start and stop of the simulation, trafficflows, tracing, and other events.

```
# schedule events for all the flows
$ns at 0.25 "$ftp start"
$ns at 0.25 "$cbr start"
$ns at 5.0 "$cbr stop"
$ns at 5.0 "$ftp stop"
# call the finish procedure after 6 seconds of simulation time
$ns at 6 "finish"
# run the simulation
$ns run
```

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

```
#Create a simulator object set ns [new Simulator]

#Open a nam trace file

set nf [open PING.nam w]

$ns namtrace-all $nf

#Open a trace file

set nt [open PING.tr w]

$ns trace-all $nt

#Define a finish procedure proc finish {} {
global ns nf nt

$ns flush-trace close $nf
close $nt

exec nam PING.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 instproc recv {from rtt} {
$self instvar node_

puts "node [$node_ id] received ping answer from \

$from with round-trip-time $rtt ms."

}

#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

proc SendPingPacket { } { global ns p2 p3
set intervalTime 0.00

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
```

AWK FILE:(Open a new editor using “gedit command” and write awk file and save with “.awk” extension)

```
BEGIN{
        count=0;

}
{
if($1==”d”)

count++;

} END{
```

printf (“No. of packets dropped is= %d\n”, count);

}

Steps for execution
 Open gedit editor and type program. Program name should have the extension “ .tcl ” [root@localhost ~]# gedit lab2.tcl

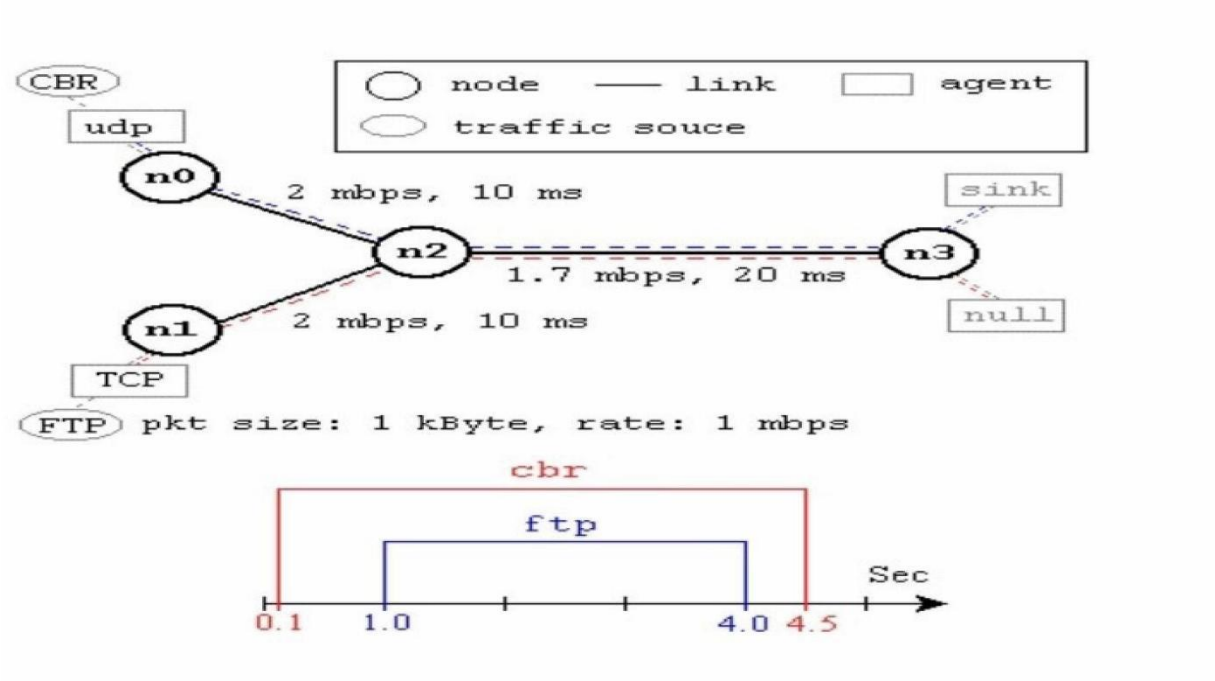
 Save the program and close the file.
 Open gedit editor and type awk program. Program name should have the extension “.awk ” [root@localhost ~]# gedit lab2.awk
 Save the program and close the file.
 Run the simulation program [root@localhost~]# ns lab2.tcl
 Here “ns” indicates network simulator. We get the topology shown in the snapshot.
 Now press the play button in the simulation window and the simulation will begins.
 After simulation is completed run awk file to see the output , [root@localhost~]# awk –f lab2.awk lab2.tr
 To see the trace file contents open the file as , [root@localhost~]# gedit lab2.t

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

```
#Creating a TCP agent and connecting it to n1set tcp1 [new Agent/TCP]
#Specifying tcp traffic to have blue color as defined in the second line of the program
$tcp1 set fid_ 0
$ns attach-agent $n1 $tcp1
#Creating a Sink Agent and attaching it to n3set sinkTCP3 [new Agent/TCPSink]
$ns attach-agent $n3 $sinkTCP3
#Connecting TCP agent with Sink agent
$ns connect $tcp1 $sinkTCP3
#Specifying the UDP agent set udp0 [new Agent/UDP]
#Specifying udp traffic to have red color as defined in the second line of program
$udp0 set fid_ 1
#Attaching the UDP agent with n0
$ns attach-agent $n0 $udp0
#Specifying the Null agentset null0 [new Agent/Null]
#Attaching the NULL agent with n3
$ns attach-agent $n3 $null0
#Connecting both udp0 and null0 agents for transferring data between n0 and n1
$ns connect $udp0 $null0
#Creating FTP agent for traffic and attaching it to tcp1set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp1
#Specifying the CBR agent to generate the traffic over udp0 agentset cbr0 [new Application/Traffic/CBR]
#Each packet having 1K bytes
$cbr0 set packetSize_ 1000
#Each packet will be generated after 10ms i.e., 100 packets per second
$cbr0 set interval 0.010
#Attaching cbr0 with udp0
$cbr0 attach-agent $udp0
#Starting the FTP Traffic
$ns at 0.5 "$ftp0 start"
$ns at 4.0 "$ftp0 stop"
```

```
#Starting the cbr0 at 0.5 simulation time
$ns at 0.1 "$cbr0 start"

#Stoping the cbr0 at 4.5 simulation time
$ns at 4.5 "$cbr0 stop"

#Calling the finish procedure
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Solution:



V) Simulate to Compare Data Rate& Throughput.

```
#Make a NS simulator set ns
   [new Simulator]

   # Define a 'finish' procedure
```

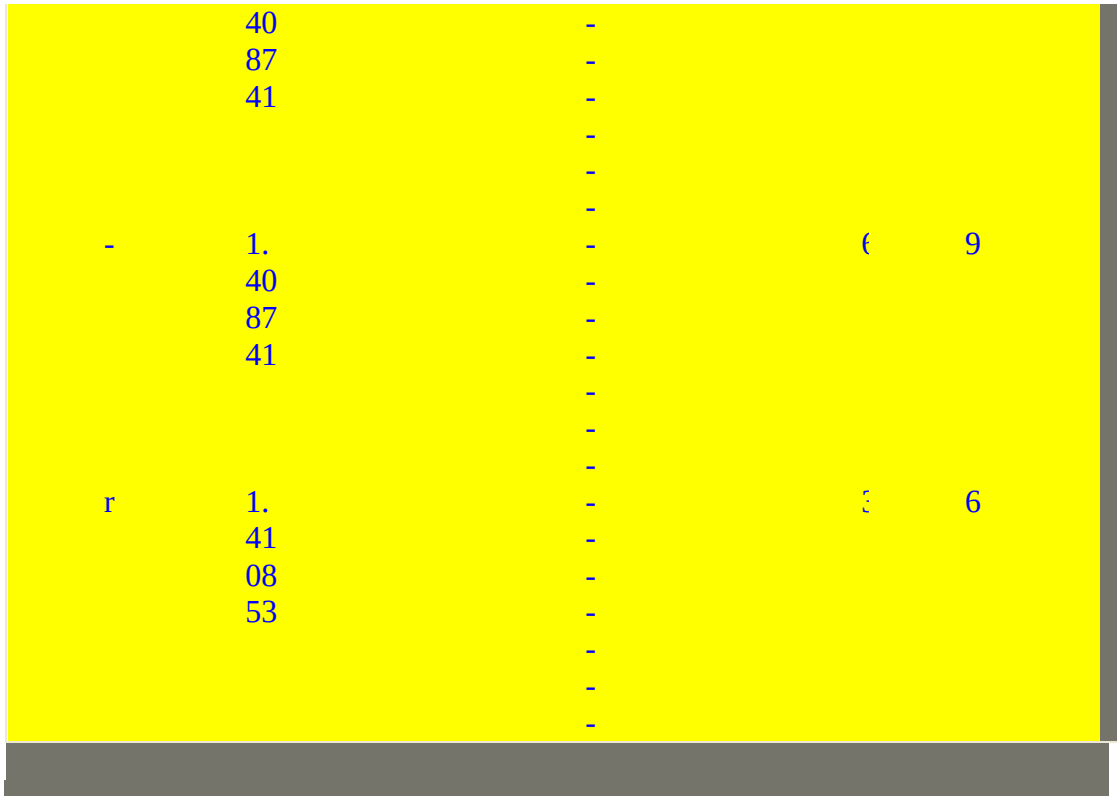
```
proc finish {} {exit 0
}

# Create the 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]
# Create the links:
```

```
$ns duplex-link $n0 $n2          2Mb   10ms DropTail
$ns duplex-link $n1 $n2          2Mb   10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail
$ns duplex-link $n3 $n4 0.5Mb   40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb   30ms DropTail

# Add a TCP sending module to node n0set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n0 $tcp1

# Add a TCP receiving module to node n4set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
# Direct traffic from "tcp1" to "sink1"
$ns connect $tcp1 $sink1
# Setup a FTP traffic generator on "tcp1"set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP     (no necessary)
# Schedule start/stop times
$ns at 0.1        "$ftp1 start"
$ns at 100.0 "$ftp1 stop"
# Set simulation end time
$ns at 125.0 "finish"    (Will invoke "exit 0")

#####################################################
## Obtain Trace date at destination (n4) ################################################## set
trace_file [open "out.tr" w]
$ns  trace-queue   $n3   $n4   $trace_file

# Run simulation !!!!
$ns run
```

Output:

```
+ 0.311227 3 4 tcp 40 0 0.0 4.0 0 0
(@ 0.311227 sec: 40 bytes of TCP data arrives at node 3)
- 0.311227 3 4 tcp 40  0 0.0 4.0 0 0
(@ 0.311227 sec: 40 bytes of TCP data departed from node 3)r 0.351867 3 4
tcp 40 -
-          0 0.0 4.0 0 0
(@ 0.351867 sec: 40 bytes of TCP data is received by node 4)
+ 0.831888 3 4 tcp 592          0 0.0 4.0 1 2
(@ 0.831888 sec: 592 bytes of TCP data arrives at node 3)
- 0.831888 3 4 tcp 5920 0.0 4.0 1 2
(@ 0.831888 sec: 592 bytes of TCP data departed from node 3)
+ 0.847675 3 4 tcp 592          0 0.0 4.0 2 3
(@ 0.847675 sec: 592 bytes of TCP data arrives at node 3)
- 0.847675 3 4 tcp 5920 0.0 4.0 2 3
(@ 0.847675 sec: 592 bytes of TCP data departed from node 3)r 0.88136 3 4
tcp 592
-          0 0.0 4.0 1 2
(@ 0.88136 sec: 592 bytes of TCP data is received by node 4)
```

r	0. 89 71 47	- - - - - - -	2	3
+	1. 36 13 81	- - - - - - -	3	6
-	1. 36 13 81	- - - - - - -	3	6
+	1. 37 71 68	- - - - - - -	4	7
-	1. 37 71 68	- - - - - - -	4	7
+	1. 39 29 55	- - - - - - -	5	8
-	1. 39 29 55	- - - - - - -	5	8
+	1.	-	6	9



Vi) Simulate to Plot Congestion for Different Source/Destination

```
set cwnd [$tcpSource set cwnd_]puts $file "$now $cwnd"
$ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}
#end #=================================
#         Nodes Definition #=================================
#Create 6 nodes set n0 [$ns node]set n1 [$ns node]set n2 [$ns node]set n3 [$ns node]set n4 [$ns node]set n5
[$ns node]
$n0 label "Source0"
$n1 label "Source1"
$n2 label "R1"
$n3 label "R2"
$n4 label "Dest0"
$n5 label "Dest1"
$ns color 1 "red"
$ns color 2 "green"
$ns color 3 "blue"
$ns color 4  "orange"

#=================================
#         Links Definition #=================================

#add manually
set lan [$ns newLan "$n0 $n1 $n2" 0.5Mb 40ms LL Queue/DropTailMAC/802_3 Channel]
$ns duplex-link $n2 $n3 10Mb 100ms DropTail
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

```
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTailMAC/802_3 Channel] set loss_module
[new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n2 $n3

#end #=================================
#        Agents Definition #=================================

#Setup a TCP connection set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0 set sink2 [new Agent/TCPSink]
$ns attach-agent $n4 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500#Setup a TCP connection set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1 set sink3 [new Agent/TCPSink]
$ns attach-agent $n5 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500 #=================================
#        Applications Definition #=================================
#Setup a FTP Application over TCP connectionset ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 0.1 "$ftp0 start"
$ns at 9.8 "$ftp0 stop"
#Setup a FTP Application over TCP connectionset ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns at 1 "$ftp1 start"
$ns at 9.9 "$ftp1 stop"#add manually
$ns at 0.1 "PlotWindow $tcp0 $wf0"
$ns at 0.5 "PlotWindow $tcp1 $wf1"
$tcp0 set class_ 1
$tcp1 set class_ 2#end #=================================
#        Termination #=================================
#Define a 'finish' procedureproc finish {}
{
global ns tracefile namfile
$ns flush-trace close
$tracefileclose $namfile exec nam 5.nam &
exec xgraph WinFile0 WinFile1 &exit 0
}

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Process to Execute the Program

We need to have ns2 pre installed

Head to Cmd

Navigate to ns 3.tcl file saved using command

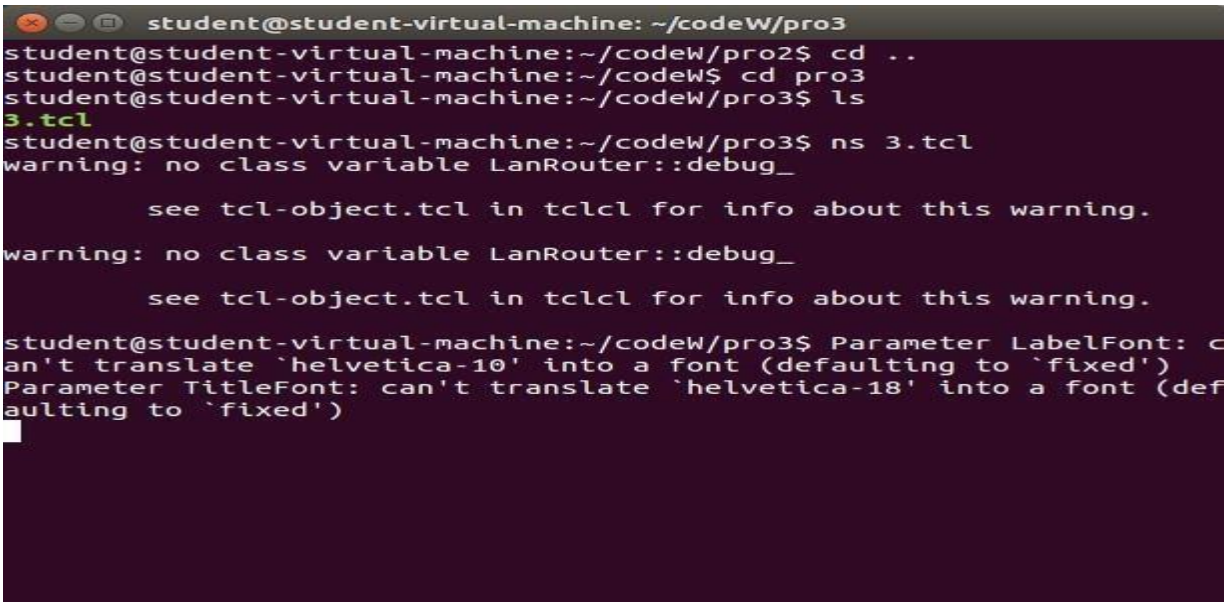


Fig 3.1: Output .

Fig 3.2: Output .

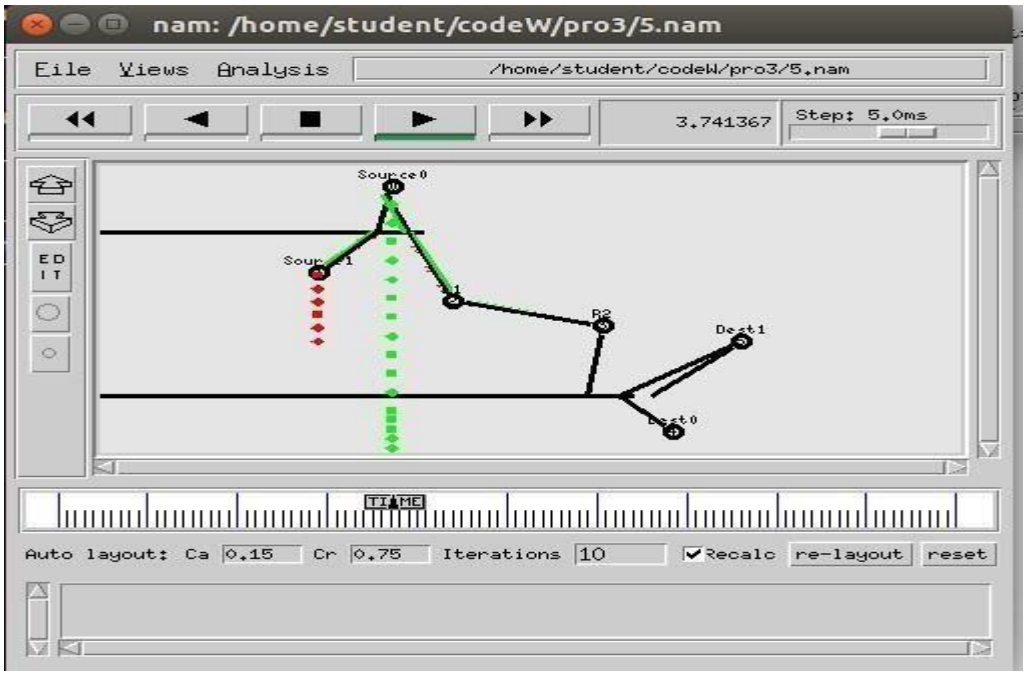


Fig 3.3: Output .

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

```
# basic1.tcl simulation: A---R---B
#Create a simulator objectset ns
[new Simulator]

#Open the nam file basic1.nam and the variable-trace file basic1.trset namfile
[open basic1.nam w]
$ns namtrace-all $namfile
set tracefile [open basic1.tr w]
$ns trace-all $tracefile

#Define a 'finish' procedureproc
finish {} {
        global ns namfile tracefile
        $ns flush-trace close
```

```
exit 0
}
#Create the network nodesset A [$ns node]
set R [$ns node]set B [$ns node]
#Create a duplex link between the nodes
$ns duplex-link $A $R 10Mb 10ms DropTail
$ns duplex-link $R $B 800Kb 50ms DropTail
# The queue size at $R is to be 7, including the packet being sent
$ns queue-limit $R $B 7
# some hints for nam
```

```
# color packets of flow 0 red
$ns color 0 Red
$ns duplex-link-op $A $R orient right
$ns duplex-link-op $R $B orient right
$ns duplex-link-op $R $B queuePos 0.5
# Create a TCP sending agent and attach it to Aset tcp0 [new Agent/TCP/Reno]
# We make our one-and-only flow be flow 0
$tcp0 set class_ 0
$tcp0 set window_ 100
$tcp0 set packetSize_ 960
$ns attach-agent $A $tcp0 # Let's trace some variables
$tcp0 attach $tracefile
$tcp0 tracevar cwnd_
$tcp0 tracevar ssthresh_
$tcp0 tracevar ack_
$tcp0 tracevar maxseq_
#Create a TCP receive agent (a traffic sink) and attach it to Bset end0 [new Agent/TCPSink]
$ns attach-agent $B $end0
#Connect the traffic source with the traffic sink
$ns connect $tcp0 $end0
#Schedule the connection data flow; start sending data at T=0, stop at T=10.0set myftp [new
Application/FTP]
$myftp attach-agent $tcp0
$ns at 0.0 "$myftp start"
$ns at 10.0 "finish" #Run the simulation
$ns run
```

After running this script, there is no command-line output (because we did not ask for any); however, the files basic1.tr and basic1.nam are created. Perhaps the simplest thing to do at this point is to view the animation with nam, using the command nam basic1.nam.

In the animation we can see slow start at the beginning, as first one, then two, then four andthen eight packets are sent. A little past T=0.7, we can see a string of packet losses. This is visible in the animation as a tumbling series of red squares from the top of R's queue. After that, the TCP sawtooth takes over; we alternate between the cwnd linear-increase phase (congestion avoidance), packet loss, and threshold slow start. During the linear-increase phase the bottleneck link is at first incompletely utilized; once the bottleneck link is saturated the router queue begins to build.