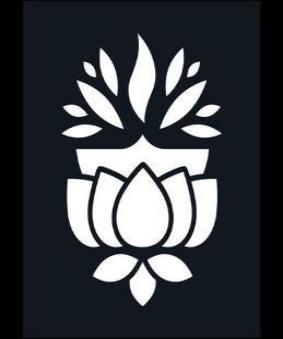
# COMPUTER NETWORKS LAB

LAB MANUAL



**Department of Computer Science and Engineering**

**(AIML)**

**Department of Computer Science and Engineering**

# COMPUTER NETWORKS LAB ACADEMIC YEAR 2024-2025

|  |  |
| --- | --- |
| **Lab Name /Course Name** | **COMPUTER NETWORKS LAB** |
| **Subject Code / Course Code** |  |
| **Year &Semester** | **III YEAR/I SEM** |
| **Branch** | **CSE – AIML** |
| **Name of the Faculty** |  |
| **Designation** |  |

**VISION AND MISSION OF THE DEPARTMENT**

**VISION:**

To be frontier of Computer Science & Engineering and to produce globally competent engineers committed to serve the society.

**MISSION:**

**M1:** To strengthen core competence in Computer Science & Engineering through Outcome Based Education.

**M2:** To produce successful graduates by providing state of art infrastructure and skill development activities.

**M3:** To produce innovative research in Computer Science & Engineering and encourage community development programs.

# LAB DESCRIPTION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Course Title** | **COMPUTER NETWORKS LAB** | | | |
| **Course Code** |  | | | |
| **Regulation** | **R18** | | | |
| **Course Structure** | **Lectures** | **Tutorials** | **Practical** | **Credits** |
| **0** | **0** | **3** | **1.5** |
| **Course Faculty** |  | | | |

### OVERVIEW:

To understand the functionalities of various layers of OSI model, to understand the operating System functionalities

### PREREQUISITES:

1. Prior knowledge of C and C++ Programming.
2. Prior knowledge about NS2 Simulator.

### COURSE OBJECTIVES:

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

### COURSE OUTCOME (CO):

|  |  |  |
| --- | --- | --- |
| **S.NO** | **Course Outcomes (CO)** | **Blooms Taxonomy Level** |
| After completing this course the student must demonstrate the knowledge and ability to: | | |
| **CO1** | Implement data link layer farming methods. | L2: Understand |
| **CO2** | Analyze error detection and error correction codes. | L3: Apply |
| **CO3** | Implement and analyze routing and congestion issues in  network design. | L2: Understand |
| **CO4** | Implement Encoding and Decoding techniques used in  presentation layer. | L2: Understand |
| **CO5** | To be able to work with different network tools. | L3: Apply |

**PROGRAM OUTCOME (PO):**

|  |  |
| --- | --- |
| **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. |

### PROGRAM SPECIFIC OUTCOMES (PSO):

|  |  |
| --- | --- |
| **PSO1** | **Software Development and Research Ability:** Ability to understand the structure and  development methodologies of software systems. Possess professional skills and |

|  |  |
| --- | --- |
|  | knowledge of software design process. Familiarity and practical competence with a broad range of programming language and open source platforms. Use knowledge in various domains to identify research gaps and hence to provide solution to new ideas and  innovations. |
| **PSO2** | **Foundation of mathematical concepts:** Ability to apply the acquired knowledge of basic skills, principles of computing, mathematical foundations, algorithmic principles, modeling and design of computer- based systems in solving real world engineering  Problems. |
| **PSO3** | **Successful Career:** Ability to update knowledge continuously in the tools like Rational Rose, MATLAB, Argo UML, R Language and technologies like Storage, Computing, Communication to meet the industry requirements in creating innovative career paths for  immediate employment and for higher studies. |

### LAB CODE

* 1. Students should report to the concerned lab as per the time table.
  2. Students who turn up late to the labs will in no case be permitted to do the program schedule for the day.
  3. After completion of the program, certification of the concerned staff in-charge in the observation book is necessary.
  4. Student should bring a notebook of 100 pages and should enter the readings observations into the notebook while performing the experiment.
  5. The record of observations along with the detailed experimental procedure of the experiment in the immediate last session should be submitted and certified staff member in-charge.
  6. Not more than 3-students in a group are permitted to perform the experiment on the set.
  7. The group-wise division made in the beginning should be adhered to and no mix up of students among different groups will be permitted.
  8. The components required pertaining to the experiment should be collected from stores in-charge after duly filling in the requisition form.
  9. When the experiment is completed, should disconnect the setup made by them, and should return all the components/instruments taken for the purpose.
  10. Any damage of the equipment or burn-out components will be viewed seriously either by putting penalty or by dismissing the total group of students from the lab for the semester/year.
  11. Students should be present in the labs for total scheduled duration.
  12. Students are required to prepare thoroughly to perform the experiment before coming to laboratory.

### LIST OF EXPERIMENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **EXPERIMENT NO.** | **NAME OF THE EXPERIMENT** | **PAGE NO.** |
| 1 | 1 | Implement the data link layer framing methods such as character, character-stuffing and bit stuffing. | 10 |
| 2 | 2 | Write a program to compute CRC code for the polynomials CRC-12, CRC-16 and CRC CCIP | 18 |
| 3 | 3 | Develop a simple data link layer that performs the flow control using the sliding window protocol, and loss recovery using the Go-Back-N mechanism. | 24 |
| 4 | 4 | Implement Dijsktra’s algorithm to compute the shortest path through a network | 27 |
| 5 | 5 | Take an example subnet of hosts and obtain a broadcast tree for the subnet. | 32 |
| 6 | 6 | Implement distance vector routing algorithm for obtaining routing tables at each node. | 39 |
| 7 | 7 | Implement data encryption and data decryption | 44 |
| 8 | 8 | Write a program for congestion control using Leaky bucket algorithm. | 46 |
| 9 | 9 | Write a program for frame sorting technique used in buffers. | 49 |

**EXPERIMENT NO: 1. (a)**

### Implement the data link layer framing methods such as character, character-stuffing and 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 ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to character stuffing, in which a DLE is stuffed into the outgoing character stream before DLE in the data.

### PROGRAM ALGORITHM:

Begin

Step 1: Read frame length n

Step 2: Repeat step (3 to 4) until i<n(: Read values into the input frame (0’s and 1’s) i.e.

Step 3: initialize I 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

### PROGRAM CODE: // BIT STUFFING PROGRAM

### #include <stdio.h>

### #include <string.h>

### void main()

### {

### int a[20], b[30], i, j, k, count, n;

### // Take the length of the input frame

### printf("Enter frame length:");

### scanf("%d", &n);

### // Input the frame (0's and 1's)

### printf("Enter input frame (0's & 1's only):");

### for (i = 0; i < n; i++)

### scanf("%d", &a[i]);

### i = 0;

### count = 1; // To count consecutive 1's

### j = 0; // Index for output array

### // Loop through the input frame

### while (i < n)

### {

### if (a[i] == 1)

### {

### // Copy 1 to output

### b[j] = a[i];

### // Start checking the next bits for consecutive 1's

### for (k = i + 1; a[k] == 1 && k < n && count < 5; k++)

### {

### j++;

### b[j] = a[k];

### count++;

### // If 5 consecutive 1's are found, insert a 0

### if (count == 5)

### {

### j++;

### b[j] = 0; // Stuffing 0 after 5 consecutive 1's

### }

### i = k;

### }

### }

### else

### {

### // If current bit is 0, just copy it to the output

### b[j] = a[i];

### }

### i++;

### j++;

### count = 1; // Reset count for the next group of consecutive 1's

### }

### // Output the stuffed frame

### printf("After stuffing the frame is:");

### for (i = 0; i < j; i++)

### printf("%d", b[i]);

### printf("\n");

### }

### PROGRAM OUTPUT:

Enter frame length:5

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

1

1

1

1

After stuffing the frame is:111110

### EXPERIMENT NO: 1. (b)

**NAME OF THE EXPERIMENT:** Character Stuffing.

**AIM:** Implement the data link layer framing methods such as character, character stuffing. **HARDWARE REQUIREMENTS:** Intel based Desktop PC**:-**RAM of 512 MB **SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

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

### PROGRAM ALGORITHM:

Begin

Step 1: Initialize I and j as 0

Step 2: Declare n and pos as integer and a[20],b[50],ch as character Step 3: read the string a

Step 4: find the length of the string n, i.e n-strlen(a) Step 5: read the position, pos

Step 6: if pos > n then

Step 7: print invalid position and read again the position, pos Step 8: endif

Step 9: read the character, ch

Step 10: Initialize the array b, b[0…5] as ’d’, ’l’, ’e’, ’s’, ’t’,’x’ respectively Step 11: j=6;

Step 12: Repeat step[(13to22) until i<n Step 13: if i==pos-1 then

Step 14: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’, ‘e’ ,’ch, ’d’, ‘l’,‘e’ respectively

Step 15: increment j by 7, i.e j=j+7 Step 16: endif

Step 17: if a[i]==’d’ and a[i+1]==’l’ and a[i+2]==’e’ then Step 18: initialize array b, b[13…15]=’d’, ‘l’, ‘e’ respectively Step 19: increment j by 3, i.e j=j+3

Step 20: endif

Step 21: b[j]=a[i]

Step 22: increment I and j;

Step 23: initialize b array,b[j],b[j+1]…b[j+6] as‘d’, ‘l’,‘e’,’e’,‘t’, ‘x’,‘\0’ respectively Step 24: print frame after stuffing

Step 25: print b End

### PROGRAM CODE: //PROGRAM FOR CHARACTER STUFFING

### #include<stdio.h>

### #include<string.h>

### #include<process.h>

### void main() {

### int i = 0, j = 0, n, pos;

### char a[20], b[50], ch;

### printf("Enter string\n");

### scanf("%s", &a);

### 

### n = strlen(a);

### 

### printf("Enter position\n");

### scanf("%d", &pos);

### 

### if (pos > n) {

### printf("invalid position, Enter again :");

### scanf("%d", &pos);

### }

### printf("Enter the character\n");

### ch = getche();

### b[0] = 'd'; b[1] = 'l'; b[2] = 'e'; b[3] = 's'; b[4] = 't'; b[5] = 'x';

### j = 6;

### while (i < n) {

### if (i == pos - 1) {

### b[j] = 'd'; b[j + 1] = 'l'; b[j + 2] = 'e'; b[j + 3] = ch;

### b[j + 4] = 'd'; b[j + 5] = 'l'; b[j + 6] = 'e';

### j = j + 7;

### }

### if (a[i] == 'd' && a[i + 1] == 'l' && a[i + 2] == 'e') {

### b[j] = 'd'; b[j + 1] = 'l'; b[j + 2] = 'e';

### j = j + 3;

### }

### b[j] = a[i];

### i++;

### j++;

### }

### b[j] = 'd'; b[j + 1] = 'l'; b[j + 2] = 'e'; b[j + 3] = 'e';

### b[j + 4] = 't'; b[j + 5] = 'x';

### b[j + 6] = '\0';

### printf("\nframe after stuffing:\n");

### printf("%s", b);

### }

### PROGRAM OUTPUT:

Enter string

hello

Enter position

2

Enter the character

y

frame after stuffing:

dlexdlexstxhdyledledleoletx

### EXPERIMENT NO: 2.

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

**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 sometimes known as Cyclic Redundancy Code.

### PROGRAM ALGORITHM/FLOWCHART:

Begin

Step 1:Declare I, j , fr[8], dupfr[11], recfr[11], tlen, flag, gen[4], genl, frl, rem[4] as integer Step 2: initialize frl=8 and genl=4

Step 3: initialize i=0

Step 4: Repeat step(5to7) until i<frl Step 5: read fr[i]

Step 6: dupfr[i]=fr[i] Step 7: increment i Step 8: initialize i=0

Step 9: repeat step(10to11) until i<genl Step 10: read gen[i]

Step 11: increment i Step 12: tlen=frl+genl-1

Step 13: initialize i=frl

Step 14: Repeat step(15to16) until i<tlen Step 15: dupfr[i]=0

Step 16: increment i

Step 17: call the function remainder(dupfr) Step 18: initialize i=0

Step 19: repeat step(20 to 21) until j<genl Step 20: recfr[i]=rem[j]

Step 21: increment I and j

Step 22: call the function remainder(dupfr) Step 23: initialize flag=0 and i=0

Step 24: Repeat step(25to28) until i<4 Step 25: if rem[i]!=0 then

Step 26: increment flag Step 27: end if

Step 28: increment i Step 29: if flag=0 then

Step 25: print frame received correctly Step 25: else

Step 25: print the received frame is wrong End

Function: Remainder(int fr[]) Begin

Step 1: Declare k,k1,I,j as integer Step 2: initialize k=0;

Step 3: repeat step(4 to 14) until k< frl Step 4: if ((fr[k] == 1) then

Step 5: k1=k

Step 6: initialize i=0, j=k

Step 7: repeat step(8 to 9) until i< genl Step 8: rem[i] =fr[j] exponential gen[i] Step 9: increment I and j

Step 10: initialize I = 0

Step 11: repeat step(12to13) until I <genl Step 12: fr[k1] = rem[i]

Step 13: increment k1 and i Step 14: end if

End

### PROGRAM CODE: // PROGRAM FOR CYCLIC REDUNDENCY CHECK

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <unistd.h> // For sleep()

#include <time.h>

int copy();

int check();

int i, j, k, t, count = 0, num = 0;

int gen[10], frame[30], rem[30], temp[30];

int main() {

char c, plym[50];

char ch[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9'};

// Initialize arrays to zero

memset(plym, '\0', sizeof(plym));

memset(temp, 0, sizeof(temp));

memset(rem, 0, sizeof(rem));

memset(frame, 0, sizeof(frame));

memset(gen, 0, sizeof(gen));

printf("\t\t\t\*\*CYCLIC REDUNDANCY CHECK-12\*\*\n\n\n");

printf("Enter the polynomial: ");

fgets(plym, sizeof(plym), stdin); // Safe string input

// Remove the newline character from fgets

size\_t len = strlen(plym);

if (len > 0 && plym[len - 1] == '\n') {

plym[len - 1] = '\0';

}

for (i = 0; i < strlen(plym); i++) {

if (plym[i] == 'x') {

i++;

for (; plym[i] != '+' && plym[i] != '\0'; count++, i++) { }

if (count == 3) {

for (i = i - 1, j = 3; j <= 9; j++) {

if (plym[i] == ch[j]) {

printf("\nError: Polynomial's degree is too high\n");

return 1; // Exit due to error

}

}

for (j = 0, num = 10; j <= 2; j++) {

if (plym[i] == ch[j]) {

num = num + j;

frame[num] = 1;

}

}

}

if (count == 2) {

for (i = i - 1, j = 1, num = 0; j <= 9; j++) {

if (plym[i] == ch[j]) num = j;

frame[num] = 1;

}

}

if (count == 0) frame[1] = 1;

count = 0;

} else if (plym[i] == '1') {

frame[0] = 1;

}

}

printf("FRAME is: ");

for (i = 12, j = 0; i >= 0; i--, j++) {

temp[j] = frame[i];

printf("%d", frame[i]);

}

printf("\n\n\n>>>> Both high & low order bits of the GENERATOR must be 1 <<<<");

printf("\nEnter the generator: ");

for (num = i = 0; (c = getchar()) != '\n'; i++, num++) {

if (c == '1') gen[i] = 1;

else if (c == '0') gen[i] = 0;

else {

printf("\nError: Generator should only contain 0 or 1\n");

return 1; // Exit due to error

}

}

for (j = 13, i = i - 1; i > 0; i--, j++) temp[j] = 0;

printf("\n\nFRAME after appending 0's: ");

copy(); check();

printf("\nThe REMAINDER is: ");

for (i = 13; i < j; i++) {

temp[i] = rem[i];

printf("%d", rem[i]);

}

printf("\n\n\nTransmitting FRAME...");

sleep(1); // Delay in seconds

printf("\n\n\nTransmitted FRAME is: ");

copy(); check();

printf("\nFrame received");

printf("\n\n\nChecking for errors...");

sleep(1);

printf("\n\n\nReceived frame is: ");

copy(); check();

printf("\nThe remainder is: ");

for (i = 13; i < j; i++) printf("%d", rem[i]);

printf("\nDATA SENT SUCCESSFULLY\n");

return 0;

}

int check() {

for (i = 0; i <= 12; i++) {

if (rem[i] == 0) continue;

else {

for (k = 0, t = i; k < num; k++, t++) {

rem[t] = rem[t] ^ gen[k]; // XOR operation for CRC

}

}

}

return 0;

}

int copy() {

for (i = 0; i < j; i++) {

printf("%d", temp[i]);

rem[i] = temp[i];

}

return 0;

}

### PROGRAM OUTPUT:

### Enter the polynomial: x^3+x^2

### FRAME is: 0000000001101

### >> Both high & low order bits of the GENERATOR must be 1 <<<<

### Enter the generator: 111

### FRAME after appending 0's: 000000000110100

### The REMAINDER is: 10

### Transmitting FRAME...

### Transmitted FRAME is: 000000000110110

### Frame received

### Checking for errors...

### Received frame is: 000000000110110

### The remainder is: 00

### DATA SENT SUCCESSFULLY

### === Code Execution Successful ===

### EXPERIMENT NO: 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:** 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.

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

Sliding window is a technique for controlling transmitted data frames between two network computers where reliable and sequential delivery of data frames is required, such as when using the Data Link Layer (OSI model) or Transmission Control Protocol (TCP).In the sliding window technique, each data frame (for most data link layers) and byte (in TCP) include a unique consecutive sequence number, which is used by the receiving computer to place data in the correct order. Data corruption occurred in transit. The objective of the sliding window technique is to use the sequence numbers to avoid duplicated at a and to request missing data. Sliding window is also known as windowing.

# Go-Back-N ARQ:

Stop and wait ARQ mechanism does not utilize the resources at their best. When the acknowledgement is received, the sender sits idle and does nothing. In Go-Back-N ARQ method, both sender and receiver maintain a window. The sending-window size enables the sender to send multiple frames without receiving the acknowledgement of the previous ones. The receiving-window enables the receiver to receive multiple frames and acknowledge them. The receiver keeps track of incoming frame’s sequence number. When the sender sends all the frames in window, it checks up to what sequence number it has received positive acknowledgement. If all frames are positively acknowledged, the sender sends next set of

frames. If sender finds that it has received NACK or has not receive any ACK for a particular frame, it retransmits all the frames after which it does not receive any positive ACK.

# SOURCE CODE:

# #include<stdio.h>

# int main() {

# int w, i, f, frames[50];

# 

# // Input window size

# printf("Enter window size: ");

# scanf("%d", &w);

# 

# // Input number of frames to transmit

# printf("\nEnter number of frames to transmit: ");

# scanf("%d", &f);

# 

# // Input the frames

# 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, the sender waits for an acknowledgment sent by the receiver\n\n", w);

# 

# // Loop to simulate sending frames in windows and receiving acknowledgments

# for(i = 1; i <= f; i++) {

# printf("%d ", frames[i]);

# 

# // Once the window size is reached, send acknowledgment

# if(i % w == 0) {

# printf("\nAcknowledgment of above frames is received by sender\n\n");

# }

# }

# 

# // For the remaining frames if the number of frames isn't a perfect multiple of the window size

# if(f % w != 0) {

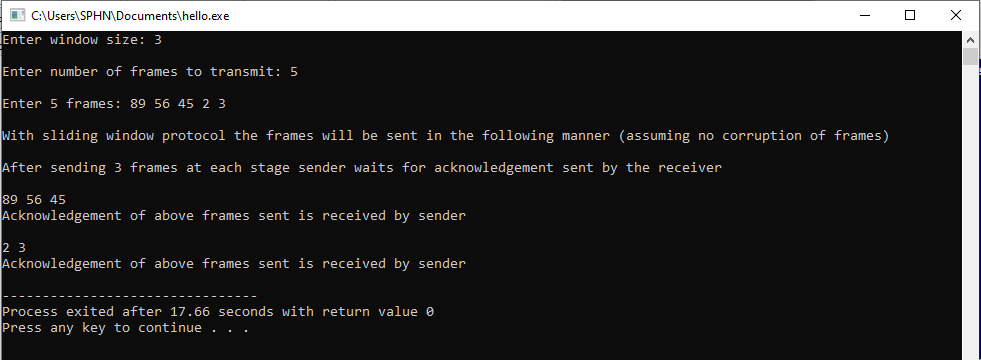
# printf("\nAcknowledgment of above frames is received by sender\n");

# }

# return 0;

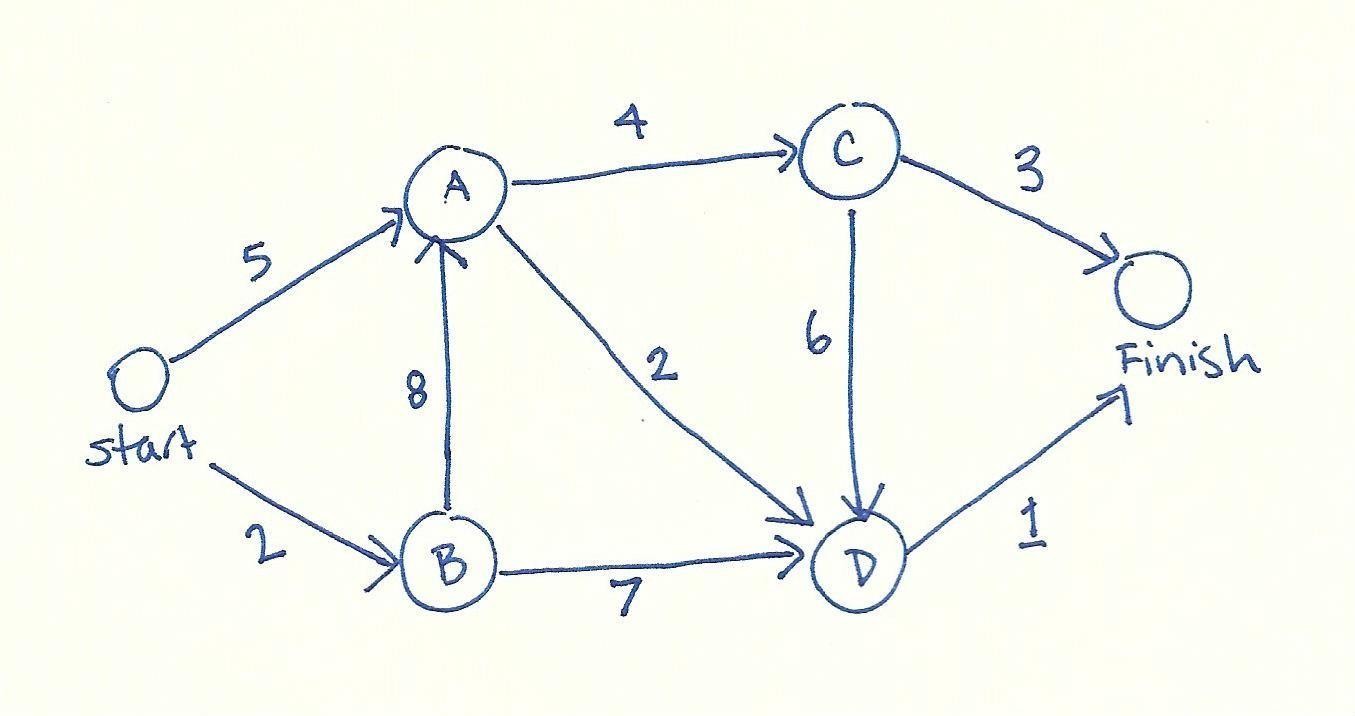
# }

# OUTPUT CONSOLE:



### EXPERIMENT NO: 4

**Implement Dijsktra’s algorithm to compute the shortest path through a network 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.



### PROGRAM 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

### PROGRAM CODE: SHORTEST PATH FOR A GIVEN GRAPH

#include <stdio.h>

#define N 5 // Assuming matrix size 5x5

int main() {

int path[N][N], i, j, min, a[N][N], p, st = 1, ed = 5, stp, edp, t[N], index;

printf("Enter the cost matrix\n");

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++) {

scanf("%d", &a[i][j]);

}

}

printf("Enter the number of paths: ");

scanf("%d", &p);

printf("Enter possible paths (each path with 5 nodes):\n");

for (i = 0; i < p; i++) {

for (j = 0; j < N; j++) {

scanf("%d", &path[i][j]);

}

}

// Calculate the cost of each path

for (i = 0; i < p; i++) {

t[i] = 0; // Initialize cost of path i to 0

stp = st - 1; // Adjusting for 0-based indexing in C arrays

for (j = 0; j < N; j++) {

edp = path[i][j] - 1; // Adjusting for 0-based indexing

t[i] += a[stp][edp]; // Add cost of moving from stp to edp

if (edp == ed - 1) {

break; // If end node is reached, stop counting for this path

} else {

stp = edp; // Move to next starting point in the path

}

}

}

// Find the minimum cost and its corresponding path

min = t[0];

index = 0;

for (i = 1; i < p; i++) {

if (min > t[i]) {

min = t[i];

index = i;

}

}

// Output the results

printf("Minimum cost: %d\n", min);

printf("Minimum cost path: ");

for (i = 0; i < N; i++) {

printf("-> %d ", path[index][i]);

if (path[index][i] == ed) {

break; // Stop if end node is reached

}

}

return 0;

}

### PROGRAM OUTPUT:

Enter the cost matrix 1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

Enter the paths 2

Enter possible paths 1 2 3 4 5

1 2 3 4 5

Minimum cost 14

### EXPERIMENT NO: 5

**Take an example subnet of hosts and obtain a broadcast tree for the subnet. NAME OF THE EXPERIMENT:** Broadcast Tree.

**AIM:** Implement broadcast tree for a given subnet of hosts

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

This technique is widely used because it is simple and easy to understand. The idea of this algorithm is to build a graph of the subnet with each node of the graph representing a router and each arc of the graph representing a communication line. To choose a route between a given pair of routers the algorithm just finds the broadcast between them on the graph.

### ALGORITHM/FLOWCHART:

step 1:declare variable as int p,q,u,v,n; step 2:Initialize min=99,mincost=0;

step 3: declare variable as int t[50][2],i,j;

step 4: declare variable as int parent[50],edge[50][50]; step 5: Begin

step 6: write "Enter the number of nodes" step 7: read "n"

step 8: Initialize i=0

step 9: repeat step(10-12) until i<n step10: increment i

step11: write"65+i"

step12: Initialize parent[i]=-1

step13:wite "\n" step14: Initialize i=0 step15: repeat step(15-21) until i<n step16: increment i step17: write"65+i" step18: Initialize j=0

step19: repeat until j<n step20: increment j step21: read edge[i][j] step22: Initialize i=0

step23: repeat step(23-43) until i<n step24: increment i

step25: Initialize j=0 step26: repeat until j<n step27: increment j step28: if'edge[i]j]!=99

step29: if'min>edge[i][j] repeat step (29-32) step30: intialize min=edge[i][j]

step31: intialize u=i step32: intialize v=j

step33: calling function p=find(u); step34: calling function q=find(v); step35: if'P!=q repeat steps(35-39) step36: intialize t[i][0]=u

step37: intialize t[i][1]=v

step38: initialize mincost=mincost+edge[u][v] step39: call function sunion(p,q)

step40: else repeat steps(40-42) step41: Intialize t[i][0]=-1; step42: Intialize t[i][1]=-1; step43: intialize min=99;

step44; write"Minimum cost is %d\n Minimum spanning tree is",mincost step45: Initialize i=0

step46: repeat until i<nstep47: increment istep48: if't[i][0]!=-1 && t[i][1]!=-1'repeat step(48- 50)

step49: write "%c %c %d", 65+t[i][0], 65+t[i][1], edge[t[i][0]][t[i][1]]

step50: write"\n"step51: endstep52: called function sunion(int l,int m) repeat step(51-52) step53: intialize parent[l]=m

step54: called function find(int l) repeat step(53-56) step55: if parent([l]>0)

step56: initialize l=parent

step57: return l

### SOURCE CODE:

**#include <stdio.h>**

**#define MAX 50**

**#define INF 99 // Used to represent infinity (for non-connected edges)**

**int p, q, u, v, n;**

**int min = INF, mincost = 0;**

**int t[MAX][2], i, j;**

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

**// Function to find the root of a node**

**int find(int i) {**

**while (parent[i] >= 0) {**

**i = parent[i];**

**}**

**return i;**

**}**

**// Function to perform union of two subsets**

**void sunion(int i, int j) {**

**parent[j] = i;**

**}**

**int main() {**

**printf("\nEnter the number of nodes: ");**

**scanf("%d", &n);**

**for (i = 0; i < n; i++) {**

**parent[i] = -1; // Initializing parent array to -1 (no parent)**

**}**

**printf("\nEnter the adjacency matrix (use 99 for infinity):\n");**

**for (i = 0; i < n; i++) {**

**for (j = 0; j < n; j++) {**

**scanf("%d", &edge[i][j]);**

**}**

**}**

**int edge\_count = 0;**

**while (edge\_count < n - 1) {**

**min = INF;**

**// Finding the minimum cost 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;**

**}**

**}**

**}**

**p = find(u);**

**q = find(v);**

**if (p != q) { // Check if it forms a cycle**

**t[edge\_count][0] = u;**

**t[edge\_count][1] = v;**

**mincost += min;**

**sunion(p, q);**

**edge\_count++;**

**}**

**edge[u][v] = edge[v][u] = INF; // Remove this edge from consideration**

**}**

**printf("Minimum cost is %d\n", mincost);**

**printf("Minimum spanning tree is:\n");**

**for (i = 0; i < edge\_count; i++) {**

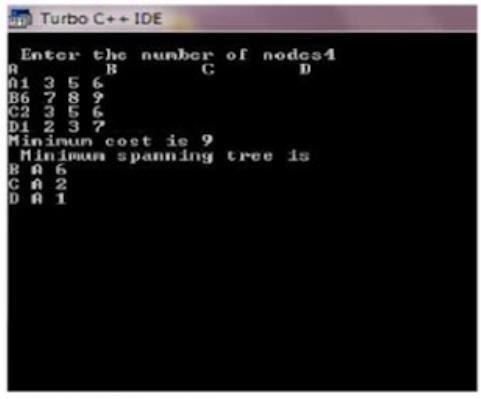
**printf("%c - %c: %d\n", 65 + t[i][0], 65 + t[i][1], edge[t[i][0]][t[i][1]]);**

**}**

**return 0;**

**}**

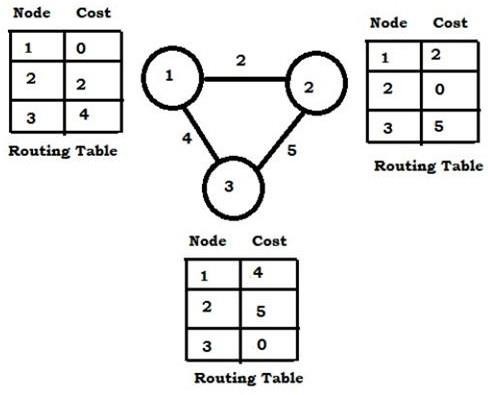
### OUTPUT:



**EXPERIMENT NO: 6**

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

**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

### SOURCE CODE:

#include <stdio.h>

struct node {

unsigned dist[20];

unsigned from[20];

} rt[10];

int main() {

int dmat[20][20]; // Distance matrix

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; // Distance to itself is zero

rt[i].dist[j] = dmat[i][j]; // Initialize distances

rt[i].from[j] = j; // Initialize "from" nodes

}

}

// Distance Vector Routing Algorithm

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] = dmat[i][k] + rt[k].dist[j];

rt[i].from[j] = k;

count++;

}

}

}

}

} while (count != 0);

// Display the results

for (i = 0; i < n; i++) {

printf("\nState value for router %d is:\n", i + 1);

for (j = 0; j < n; j++) {

printf("Node %d via %d Distance %d\n", j + 1, rt[i].from[j] + 1, rt[i].dist[j]);

}

}

printf("\n");

return 0;

}

### 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

### EXPERIMENT NO: 7

**Implement data encryption and data decryption AIM:** Implement data encryption and data decryption

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

**SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### SOURCE CODE:

#include <stdio.h>

int main() {

int i, x;

char str[100];

printf("Please enter a string: ");

fgets(str, sizeof(str), stdin); // Using fgets to prevent buffer overflow

printf("\nPlease choose one of the following options:\n");

printf("1 = Encrypt the string.\n");

printf("2 = Decrypt the string.\n");

scanf("%d", &x);

// Using switch case for options

switch (x) {

case 1: // Encrypt the string

for (i = 0; str[i] != '\0' && i < 100; i++) {

str[i] = str[i] + 3; // Key for encryption is 3

}

printf("\nEncrypted string: %s\n", str);

break;

case 2: // Decrypt the string

for (i = 0; str[i] != '\0' && i < 100; i++) {

str[i] = str[i] - 3; // Key for decryption is 3

}

printf("\nDecrypted string: %s\n", str);

break;

default:

printf("\nError: Invalid option\n");

}

return 0;

}

### OUTPUT:

**#Encryption**

### #Decryption



**EXPERIMENT NO: 8**

**Write a program for 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.

### SOURCE CODE:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#define NOF\_PACKETS 10

int rand(int a) {

int rn = (random() % 10) % a;

return rn == 0 ? 1 : rn;

}

int main() {

int packet\_sz[NOF\_PACKETS], i, clk, b\_size, o\_rate, p\_sz\_rm = 0, p\_sz, p\_time, op;

// Generating random packet sizes

for (i = 0; i < NOF\_PACKETS; ++i)

packet\_sz[i] = rand(6) \* 10;

// Displaying the packet sizes

for (i = 0; i < NOF\_PACKETS; ++i)

printf("\npacket[%d]: %d bytes\t", i, packet\_sz[i]);

// Input output rate and bucket size

printf("\nEnter the Output rate: ");

scanf("%d", &o\_rate);

printf("Enter the Bucket Size: ");

scanf("%d", &b\_size);

// Processing each packet

for (i = 0; i < NOF\_PACKETS; ++i) {

// Check if packet size + remaining size exceeds bucket capacity

if ((packet\_sz[i] + p\_sz\_rm) > b\_size) {

if (packet\_sz[i] > b\_size) // Packet size exceeds bucket size

printf("\n\nIncoming packet size (%d bytes) is greater than bucket capacity (%d bytes) - PACKET REJECTED", packet\_sz[i], b\_size);

else

printf("\n\nBucket capacity exceeded - PACKETS REJECTED!!");

} else {

// Add packet size to remaining size

p\_sz\_rm += packet\_sz[i];

printf("\n\nIncoming Packet size: %d", packet\_sz[i]);

printf("\nBytes remaining to Transmit: %d", p\_sz\_rm);

// Generate random time left for transmission

p\_time = rand(4) \* 10;

printf("\nTime left for transmission: %d units", p\_time);

// Transmitting packet in intervals

for (clk = 10; clk <= p\_time; clk += 10) {

sleep(1);

if (p\_sz\_rm) {

// Transmit based on output rate

if (p\_sz\_rm <= o\_rate) {

op = p\_sz\_rm;

p\_sz\_rm = 0;

} else {

op = o\_rate;

p\_sz\_rm -= o\_rate;

}

printf("\nPacket of size %d Transmitted", op);

printf(" --- Bytes Remaining to Transmit: %d", p\_sz\_rm);

} else {

printf("\nTime left for transmission: %d units", p\_time - clk);

printf("\nNo packets to transmit!!");

}

}

}

}

return 0;

}

### OUTPUT:

**EXPERIMENT NO: 9**

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

**AIM:** Implement a program for frame sorting technique used in buffers. **HARDWARE REQUIREMENTS:** Intel based Desktop PC**:-** RAM of 512 MB **SOFTWARE REQUIREMENTS:** Turbo C / Borland C.

### THEORY:

A frame is a digital data transmission unit in computer networking and telecommunication. A frame typically includes frame synchronization features consisting of a sequence of bits or symbols that indicate to the receiver the beginning and end of the payload data within the stream of symbols or bits it receives. If a receiver is connected to the system in the middle of a frame transmission, it ignores the data until it detects a new frame synchronization sequence*.*

### SOURCE CODE:

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

// Structure for storing frame information

struct frame {

int fslno;

char finfo[20];

};

struct frame arr[10];

int n;

// Function to sort frames based on sequence number

void sort() {

int i, j, ex;

struct frame temp;

for (i = 0; i < n; i++) {

ex = 0;

for (j = 0; j < n - i - 1; j++) {

if (arr[j].fslno > arr[j + 1].fslno) {

temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

ex++;

}

}

if (ex == 0) break; // Exit if no exchange was made, array is sorted

}

}

int main() {

int i;

// Seed for random number generation

srand(time(0));

printf("\n Enter the number of frames: ");

scanf("%d", &n);

// Input frame sequence numbers and contents

for (i = 0; i < n; i++) {

arr[i].fslno = rand() % 50; // Random sequence number between 0 and 49

printf("\n Enter the frame contents for sequence number %d: ", arr[i].fslno);

scanf("%s", arr[i].finfo);

}

// Sort frames by sequence number

sort();

// Display frames in sorted sequence

printf("\n The frames in sequence are:\n");

for (i = 0; i < n; i++) {

printf("%d\t%s\n", arr[i].fslno, arr[i].finfo);

}

return 0;

}

### OUTPUT:

Enter the number of frames:3

Enter the frame contents for sequence number 23 Wrote

Enter the frame contents for sequence number 45 Program

Enter the frame contents for sequence number 9 Vikas

The frames in sequence 9 Vikas

23 Wrote

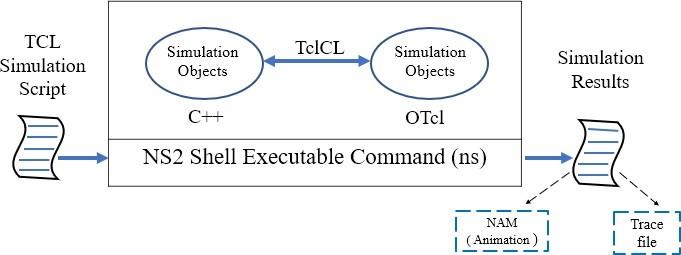
45 Program.

**Introduction to NS2 Simulator**

Network Simulator (Version 2), widely known as NS2, is simply an event- driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours.

Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation on which NS is invented. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported the development of NS through the Virtual Internetwork Test bed (VINT) project [10].2 Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile.

# NS2- Architecture:



NS2 provides users with an executable command “ns” which takes one input argument, the name of a Tcl simulation scripting file. In most cases, a simulation trace file is created and is used to plot graph and/or to create animation.

**NS2 consists of two key languages:** CCC and Object-oriented Tool Command Language (OTcl). While the CCC defines the internal mechanism (i.e., a backend) of the simulation, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The CCC and the OTcl are linked together using TclCL. Mapped to a CCC object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle is just a string (e.g., “\_o10”) in the OTcl domain and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped CCC object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may define its own procedures and variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively.

Operarational Procedure:

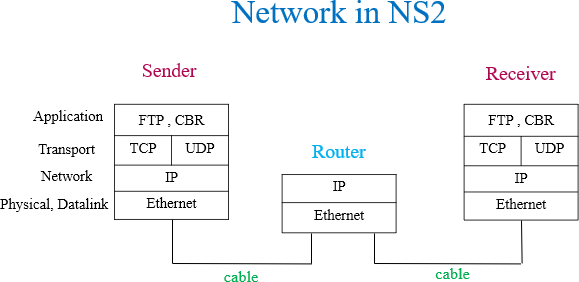
* NS2 use Tcl language for creating simulation scenario ﬁle (for example, ns1.tcl).
* **ns1.tcl :**
  1. It is the name of a tcl script file which defines the simulation scenario.
  2. Network topology, transmission time, using protocol etc... are deﬁned in scenario ﬁle.
* If we execute this scenario ﬁle, the simulation results will be ns1.tr and ns1.nam ﬁle.

## ns1.tr :

1. all the information about communication is written in this ﬁle.
2. This ﬁle is called as **Trace ﬁle**.

## ns1.nam :

1. Used to visualize the simulation.
2. This ﬁle can be execute by Nam, an animation software.



Network in NS2 Visualization:

* The node (terminal, router) in real network have 4 layer using TCP/IP model.
* Actually, forwarding node (router) have only 2 bottom layer.
* Two bottom layers are ready automatically when a node setting up.
* TCP or UDP in transport layer are shown as Agent. FTP, CBR are in application layers.
* Realistic networks use cable for link between 2 nodes. In NS2, a link is used for connection. Each link has a queue which is similar to buﬀer in realistic network. Packets sent from node should be queuing in queue.

## Basic LINUX Commands:

* **cd :** to change the directory.
* **cd .. :** previous directory
* **ls :** to list the contents of current directory
* **tar :** tape archive (to compress or decompress)
* **gedit :** editors (similar to notepad)
* **pwd :** Present Working Directory

## Installation Procedure for NS2:

1. Now install nsallinone-2.35 software, download the software from given below link

<https://sourceforge.net/projects/nsnam/files/latest/download>

1. Now we have to update the ubuntu with its latest components. Open terminal and run this command

sudo apt update

1. Before installing the NS2, we have to install the basic required packages from internet. In terminal type the following and click enter

sudo apt-get install build-essential autoconf automake libxmu-dev sudo apt install gcc-4.8 g++-4.8

1. To Extract nsallinone-2.35 to desktop, you can do it in two ways. Right click over the above file and extract it to the Desktop.

(or)

Go to the location of the download folder and copy the file to the Desktop folder and open the terminal and give the command

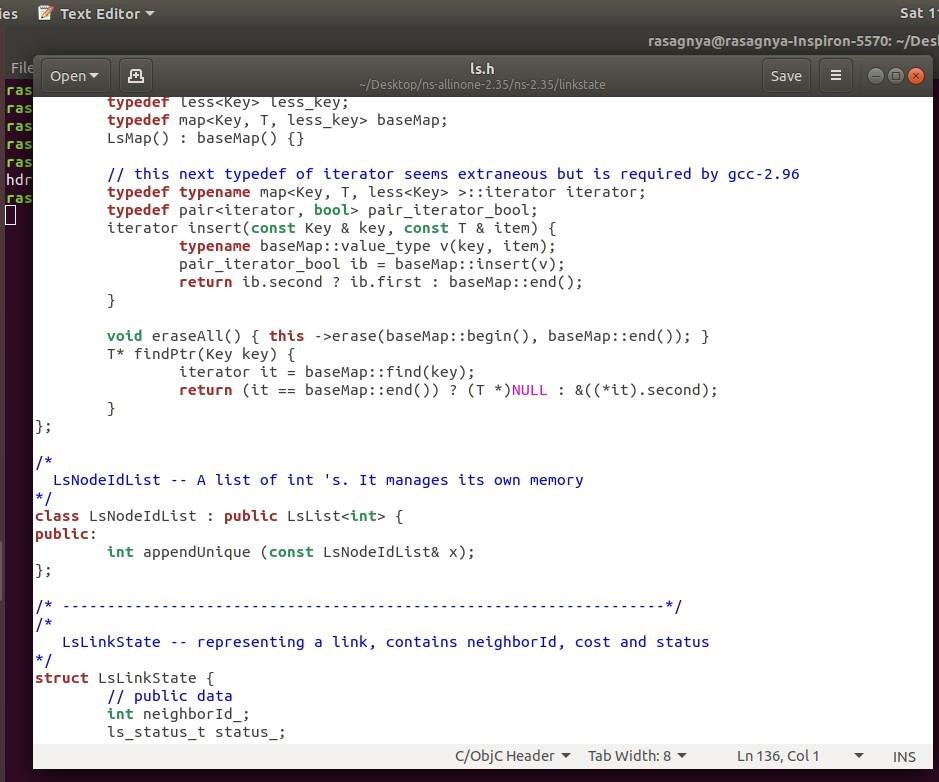
cd Desktop

tar zxvf nsallinone-2.35.tar.gz

1. Go to Desktop/ns-allinone-2.35/ns-2.35/linkstate select and open ls.h file Edit the 137 line as

void eraseAll() { this->erase(baseMap::begin(),baseMap::end());} save

the file



1. Go to Desktop/ns-allinone-2.35/otcl-1.14/Makefile.in Edit line 7: CC = gcc-4.8

Save the file

1. Go to Desktop/ns-allinone-2.35/xgraph-12.2/Makefile.in Edit line 120: CC = gcc-4.8

line 123: CPP = g++-4.8

Save the file

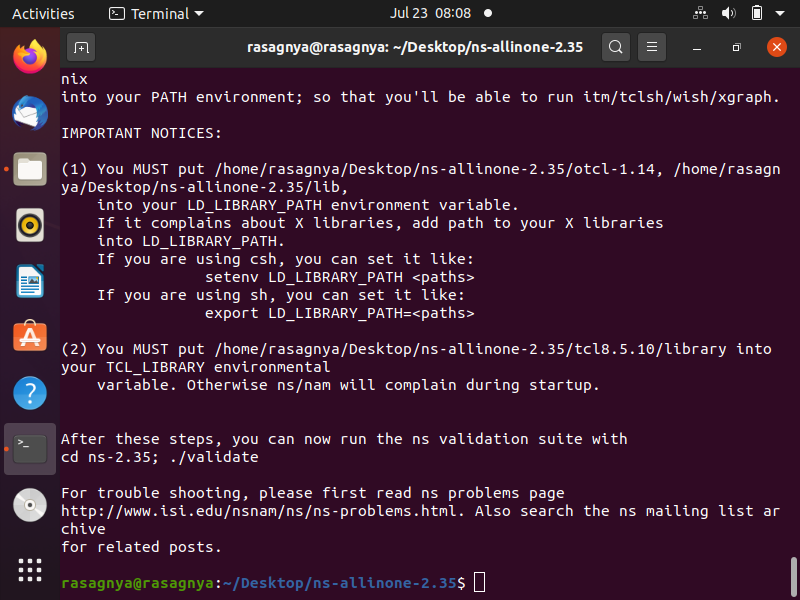
1. Now install ns2,now go to terminal and type

cd

cd Desktop

cd ns-allinone-2.35 type ./install

* + ns installation takes place for 10-20 minutes



1. Now open terminal, type pwd and copy that what it comes in a separate place. It

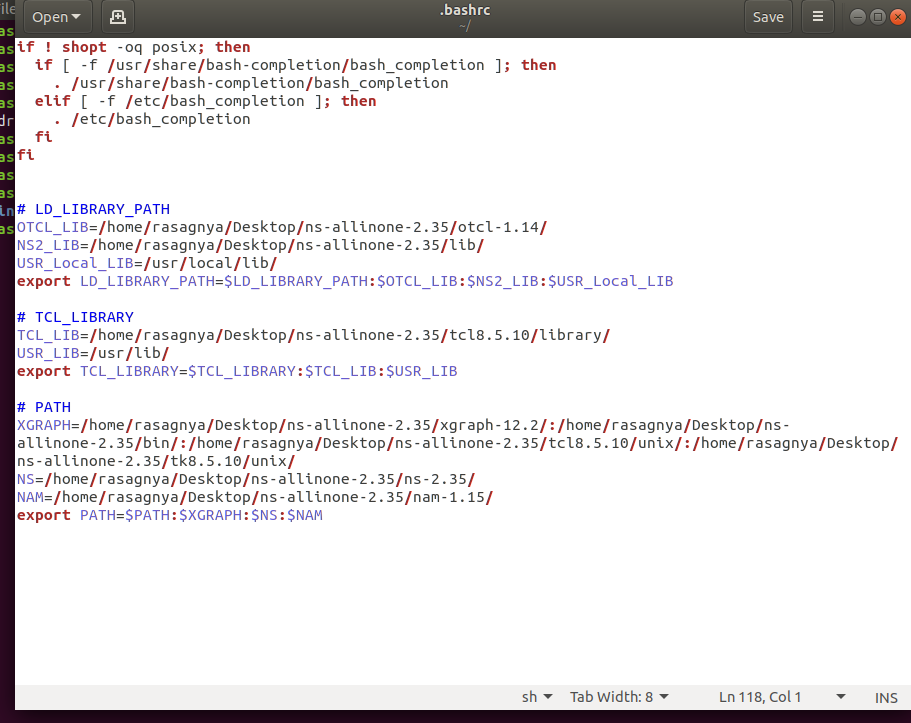
comes as /home/narasimha/Desktop/ns-allinone-2.35/

1. Now type in the terminal, type cd to go to the home folder and then type

gedit ~/.bashrc

a file opens

* now make some changes in it like, path changes
* copy the path came when u type pwd in step11 in following places, that is we are replacing the path
* save the file

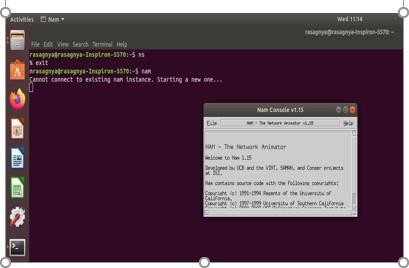


1. Now type in the terminal

source ~/.bashrc cd press enter ns press enter

%exit

nam press enter



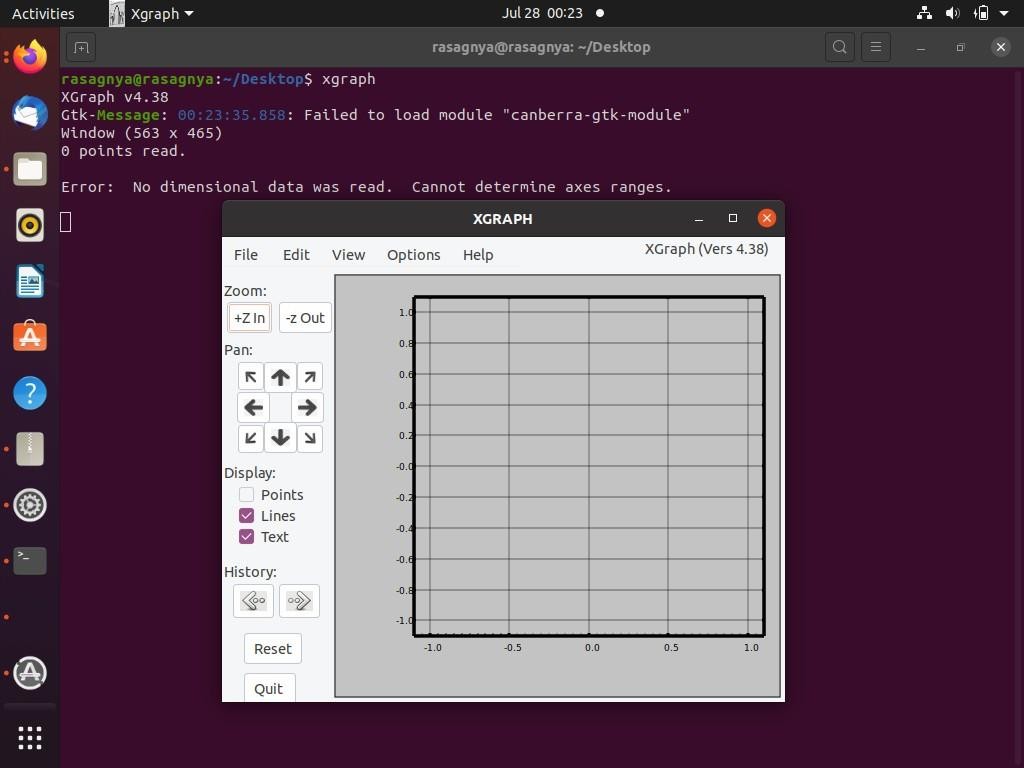
1. If you get following error(segmentation fault error) for nam, open terminal and do following
2. cd Desktop/ns-allinone-2.35/nam-1.15/
3. sudo cp nam /usr/local/bin/
4. After doing the above procedure if you get once again segmentation fault, download the nam file from the given below link

<https://drive.google.com/file/d/0B4nUSbTYSK4TclRYODFvbWgxeWM/view>

1. If xgraph is not working download the other xgraph from the given link

<http://www.xgraph.org/linux/index.html>

1. Now extract the downloaded file on to Desktop and open terminal and type xgraph it will display the xgraph.
2. If xgraph does not open, then open terminal and type the following commands
3. cd Desktop/
4. sudo cp xgraph /usr/local/bin/



1. Download NSG2.1 software from following site <https://sites.google.com/site/pengjungwu/nsg/NSG2.jar?attredirects=0> put it on the desktop, no need of extracting it
2. To run nsg we need java to be installed in our system, to install it run the following

commands,

* + open terminal and type cd press enter
  + sudo apt install default-jre

1. Gnuplot is used to plot the results, to install gnuplot open terminal and type cd press enter

sudo apt install gnuplot

Command to plot the results using gnuplot, open terminal and type

gnuplot filename gnuplot

plot “filename”

## Procedure for simulation and analysis:

1. Open terminal and go to the directory where the jar file is present.
2. After going to the directory to open the jar file. Type the command:

java –jar NSG2.jar

Where, NSG2.jar is the jar file name.

1. In scenario select new wired or wireless scenario. Place nodes wherever necessary.
2. Attach tcp or udp to source node and tcp sink or null to destination node in the Agent tab.
3. Connect to source and destination i.e., tcp and tcpsink or udp and null.
4. Attach ftp or cbr to source in the Application tab.

6.In parameters give nam and trace files name in simulation tab. Select routing protocol in wireless tab.

1. Click on tcl to get tcl code. Save the file as <filename>.tcl
2. Go to the tcl file directory in terminal and execute file by command

ns <filename>.tcl

1. After executing tcl file, NAM file and trace file are generated.
2. The simulation can be seen in the network animator.
3. Analyze the files using awk files, to calculate the parameter values type the following command

awk –f <awk filename >.awk <trace filename >.tr

# Steps to create a wired network scenario ﬁle:

* 1. Declare Simulator
  2. Create output ﬁles (tracing and animation file )
  3. Setting Node
  4. Setting Link between the nodes
  5. Setting Agent
  6. Setting Application
  7. Setting time schedules for Simulation
  8. Declare ﬁnish
  9. Declare Simulator :
     1. S should be capital letter in Simulator.
     2. Simulator is the name of the c++ class. We have to call this c++ class from the precompiled library.
     3. new is to create the instants of this simulator.
* Declare Simulator :

1. S should be capital letter in Simulator.

* set ns [new Simulator]

1. Simulator is the name of the c++ class. We have to call this c++ class from the precompiled library.
2. new is to create the instants of this simulator.

**Create output ﬁles :**

* set tracefile [open ns1.tr w]

$ns trace-all $tracefile

* set namfile [open ns1.nam w]

$ns namtrace-all $namfile

**Setting Node :**

set n0 [$ns node] set n1 [$ns node]

**Setting Link :**

$ns duplex-link $n0 $n1 3Mb 5ms DropTail

$ns duplex-link $n0 $n1 orient right-down

1. In line 1, a duplex link between n0 and n1 is declared.
2. The link between two nodes n0 and n1 has bandwidth 3Mbps and delay 5ms.
3. DropTail is a waiting queue type.
4. Line 2 sets positions of node and link for Nam.

**Setting Agent :**

**UDP Agent :**

1. To use UDP in simulation, the sender sets the Agent as UDP Agent while the receiver sets to Null Agent.
2. Null Agents do nothing except receiving the packets.
3. In ﬁrst 4 lines, udp and null Agent are set for n0 and n1, respectively.
4. Line 5 declares the transmission between udp and null. set udp [new Agent/UDP]

$ns attach-agent $n0 $udp set null [new Agent/Null]

$ns attach-agent $n1 $null

$ns connect $udp $null

**TCP Agent:**

1. To use TCP in simulation, the sender sets the Agent as TCP Agent while the receiver sets to TCPSink Agent.
2. When receiving a packet, TCPSink Agent will reply an acknowledgment packet (ACK).
3. In ﬁrst 4 lines, tcp and sink Agent are set for n0 and n1, respectively.
4. Line 5 declares the transmission between tcp and sink.

set tcp [new Agent/TCP]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $sink

$ns connect $tcp $sink

**Setting Application :**

* UDP Agent uses CBR Application
* TCP Agent uses FTP Application.

set cbr [new Application/Traffic/CBR] $cbr attach-agent $udp set ftp [new Application/FTP]

$ftp attach-agent $tcp

**Setting time schedule for simulation :**

cbr transmits data from 1.0[sec] to 8.5[sec] ftp transmits data from 2.5[sec] to 8.0[sec]

$ns at 1.0 "$cbr start"

$ns at 2.5 "$ftp start“

$ns at 8.0 "$ftp stop"

$ns at 8.5 "$cbr stop"

**Declare ﬁnish :** Simulation will stop at 10 sec

$ns at 10.0 "finish" proc finish {} {

global ns tracefile namfile

$ns flush-trace close $file close $namfile exit 0

}

$ns run

## Execution of Simulation:

* By executing below command line, simulation will be started.
  + ns ns1.tcl
* After the simulation, Nam will be started and shows the animation of simulation.
* Command for opening nam file directly
  + nam ns1.nam

## Analysis of Simulation:

AWK is a high level programming language which is used to process the text files.

Its name is derived from the family names of its authors A : Alfred Aho

W : Peter Weinberger K : Brian Kernighan

AWK scripts are very good in processing the data column wise from long trace files which we get from NS2.

* It contains mainly three parts
  1. Begin
  2. Content
  3. End
* We have to run the AWK scripts using the following command awk –f wired.awk ns1.tr

## QOS Parameters Formula:

**Throughput :** Received Bytes \*8 / Simulation Time **Total Delay :** Time delay of all received packets **Average Delay :** Total Delay / Received Packets

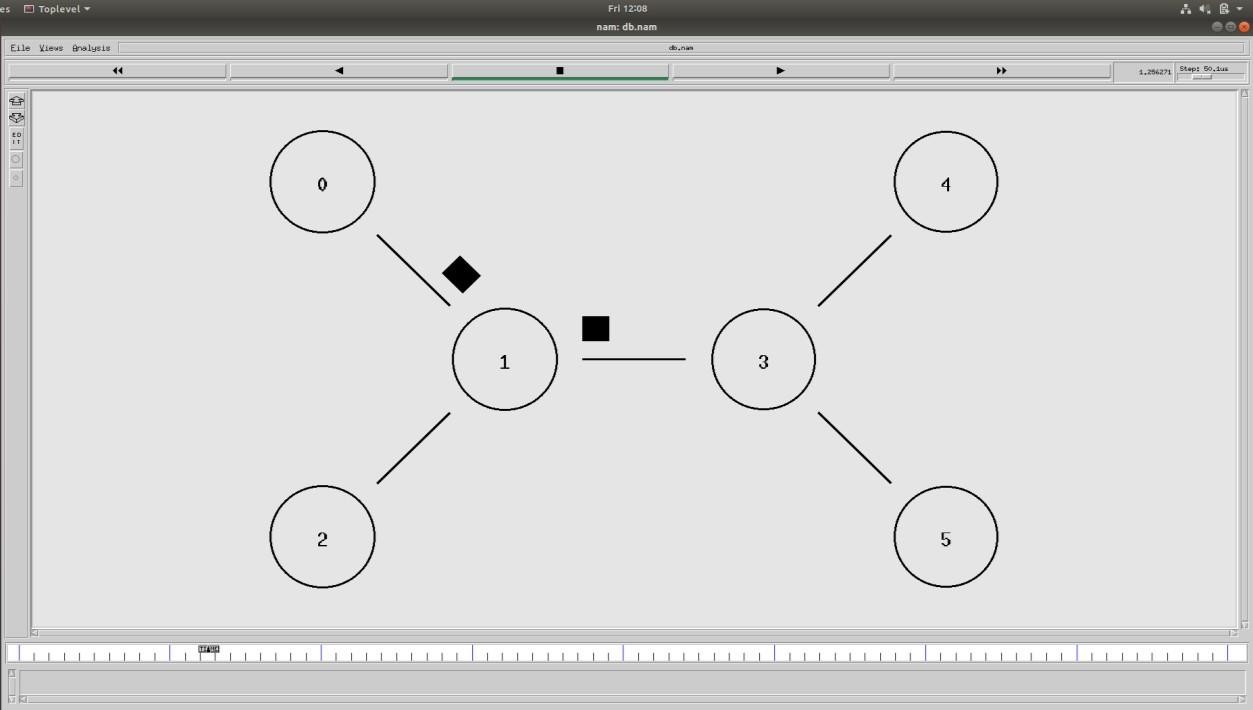
**Average Energy Consumed** : Total Energy Consumed

Number of Nodes

**Average Remaining Energy :** Initial Energy – Average Energy Consumed

**Control Overhead :** Total Number of Routing Packets

Total Number of Data Packets

SAMPLE NETWORK

**EXPERIMENT 01:**

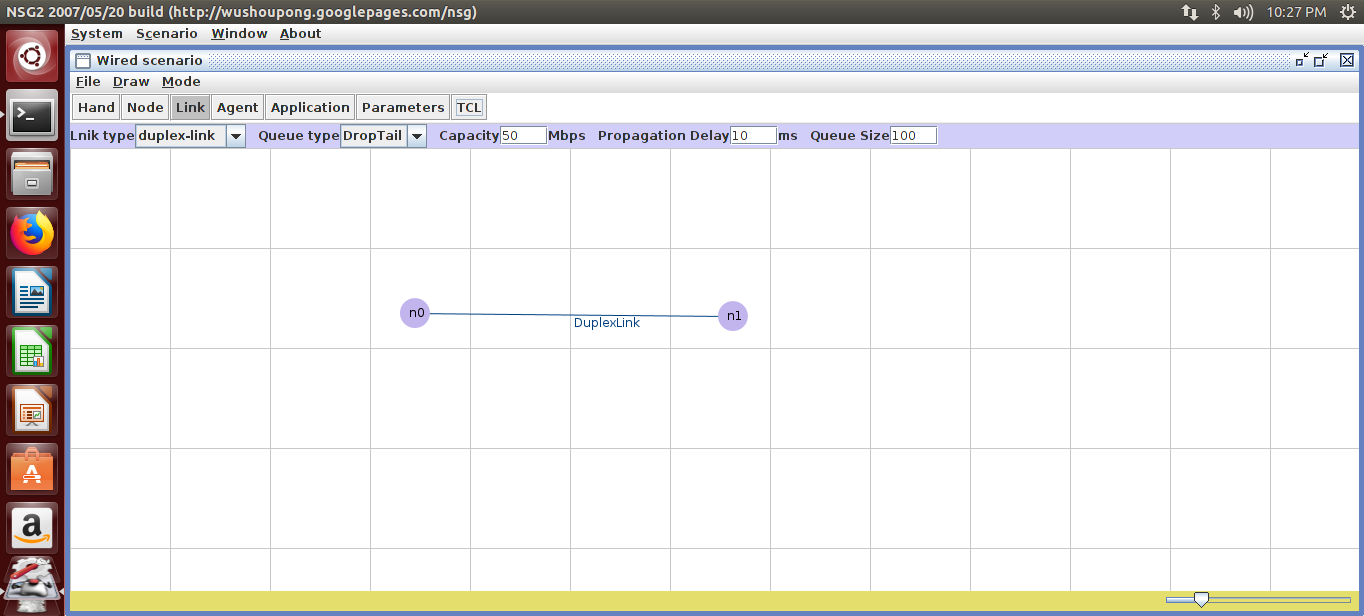
## 1.Writing a TCL Script to create two nodes and links between nodes. Aim:

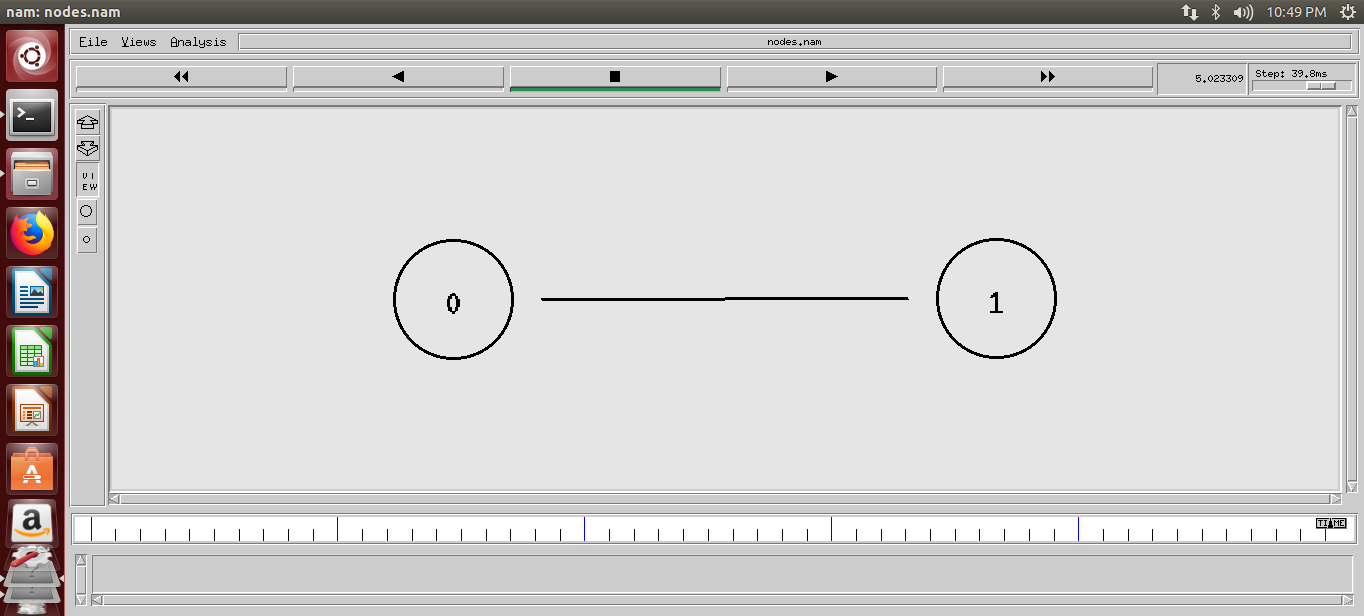
To write a TCL Script to create two nodes and links between nodes and perform data transfer between them.

## Apparatus:

PC, Ubuntu Linux, NS-2 Simulator, NSG2.1 Script Generator.

## Nodal Diagram:



**Flow Diagram:**

**TCL Script:**

# This script is created by NSG2 beta1

# <<http://wushoupong.googlepages.com/nsg>>

#===================================

# Simulation parameters setup #===================================

set val(stop) 5.0 ;# time of simulation end #===================================

# Initialization #===================================

#Create a ns simulator set ns [new Simulator]

#Open the NS trace file

set tracefile [open nodes.tr w]

$ns trace-all $tracefile

#Open the NAM trace file

set namfile [open nodes.nam w]

$ns namtrace-all $namfile #===================================

Nodes Definition #===================================

#Create 2nodes

setn0 [$ns node]

set n1 [$ns node]

#===================================

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

#Createlinks between nodes

$ns duplex-link $n0 $n1 50.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 100

#Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right

#===================================

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

#===================================

# Applications Definition #===================================

#===================================

# Termination #============================

=======

#Define a 'finish' procedure proc finish {} {

global ns tracefile namfile

$ns flush-trace close $tracefile close $namfile

exec nam nodes.nam & 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

**Procedure:**

* 1. Open a Network simulator by using NSG2.jar and create a New wired scenario.
  2. Place a required node on the worksheet.
  3. Provide a link between (Simplex/Duplex) them.
  4. Assign Agents for the Node to transmit the data from source node to the Destination node.
  5. Provide Parameters as trace, nam file to verify the analysis on packet delivery.
  6. Save the TCL Script in NS2.35 location to execute.
  7. Run the network animator with specified time interval and List the parameters like Throughput, Packet Delivery Ratio, Delay etc.

**Analysis:**

1. Throughput : NA
2. Packet Delivery ratio : NA
3. Delay : NA
4. Sent Packets : NA
5. Received Packets : NA
6. Dropped Packets : NA
7. Dropped packet ratio : NA

**Result:** Hence, TCL Script to create two nodes and links between nodes are designed, generated and executed successfully.

## EXPERIMENT 02:

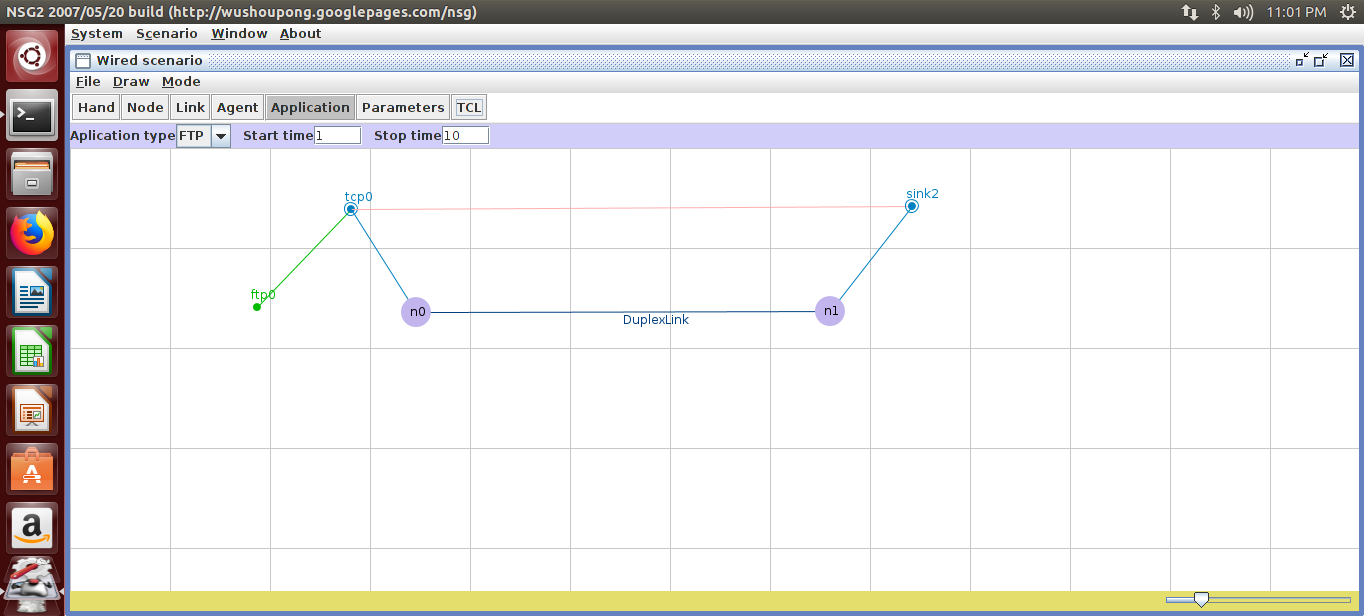
1. **Writing a TCL Script to transmit data between nodes. Aim:**

To write a TCL Script to transmit data between nodes and measure the parameter like throughput, Packet Drop, Average Delay, Sent Packets, Received Packets, Time Delay.

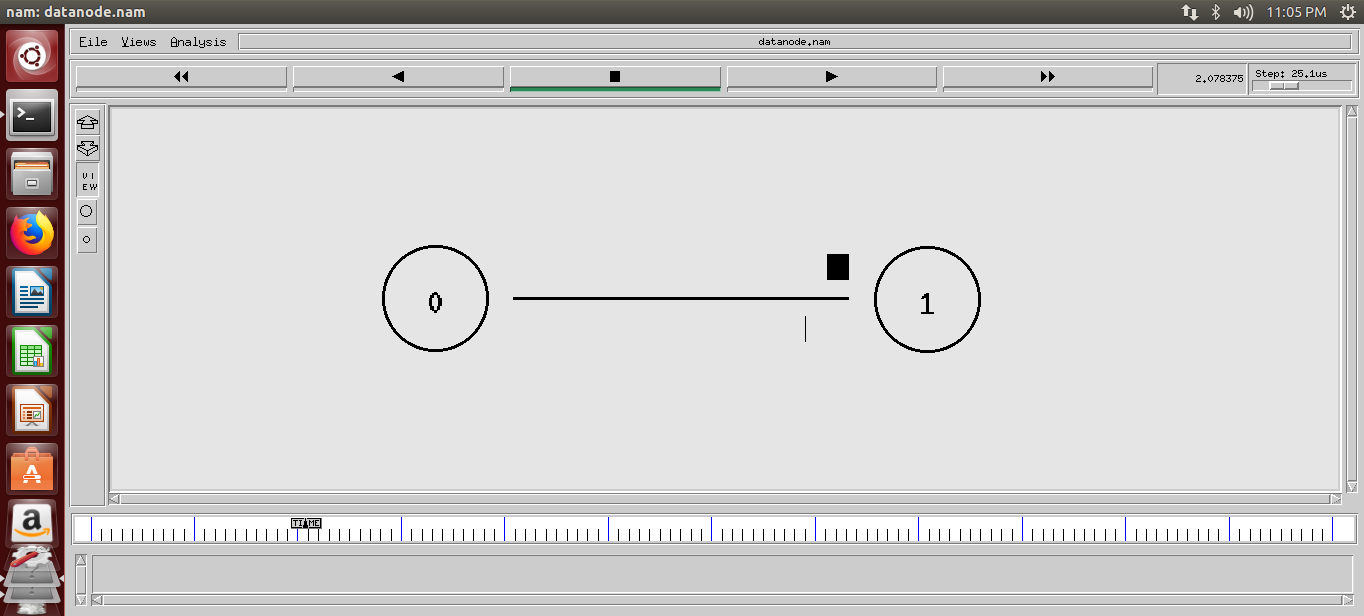
## Apparatus:

PC, Ubuntu Linux, NS-2 Simulator, NSG2.1 Script Generator.

## Nodal Diagram:



**Flow Diagram:**



**TCL Script:**

# This script is created by NSG2 beta1

# <<http://wushoupong.googlepages.com/nsg>> #===================================

# Simulation parameters setup #===================================

set val(stop) 12.0 ;# time of simulation end #===================================

# Initialization #===================================

#Create a ns simulator set ns [new Simulator]

#Open the NS trace file

set tracefile [open datanode.tr w]

$ns trace-all $tracefile

#Createlinks between nodes

$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail

$ns queue-limit $n0 $n1 50

#Give node position (for NAM)

$ns duplex-link-op $n0 $n1 orient right #===================================

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

=======

#Setup a TCP connection set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink2 [new Agent/TCPSink]

$ns attach-agent $n1 $sink2

$ns connect $tcp0 $sink2

$tcp0 set packetSize\_ 1500 #===================================

# Applications Definition #===================================

#Setup a FTP Application over TCP connection set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 10.0 "$ftp0 stop"

#===================================

# Termination #===================================

#Define a 'finish' procedure proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile close $namfile

exec nam datanode.nam & 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

**Procedure:**

* 1. Open a Network simulator by using NSG2.jar and create a New wired scenario.
  2. Place a required node on the worksheet.
  3. Provide a link between (Simplex/Duplex) them.
  4. Assign Agents for the Node to transmit the data from source node to the Destination node.
  5. Provide Parameters as trace, nam file to verify the analysis on packet delivery.
  6. Save the TCL Script in NS2.35 location to execute.
  7. Run the network animator with specified time interval and List the parameters like Throughput, Packet Delivery Ratio, Delay etc.

**Analysis:**

|  |  |  |
| --- | --- | --- |
| 1. **Throughput** | = | 10942833.06415 |
| 2. **Average delay** | = | 0.01013 |
| 3. **Sent packets** | = | 8891.00000 |
| 4. **Received Packets** | = | 8891.00000 |
| 5. **Packet drop** | = | 0.00000 |
| 6. **Packet Delivery Ratio** | = | 100.00000 |
| 7. **Packet Drop Ratio** | = | 0.00000 |
| 8. **T** | = | 10.00885 |
| 9. **Total Delay** | = | 90.02361 |

**Result:** Hence, TCL Script for the two nodes which transfers the data between was executed successfully.