Laboratory Manual

COMPUTER NETWORK
(CS-602)

For

Third Year Students CSE
Department: Computer Science & Engineering

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Department of Computer Science and Engineering

Vision of CSE Department:

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the CSE Department:

I. To provide high quality education through effective teaching-learning process emphasizing active participation of students.

II. To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.

III. To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Program Educational Objectives:

Graduates will be able to

I. Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.

II. Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.

III. Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.

IV. Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Program Outcomes (POs):

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering Fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **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.
3. **Design/development of solutions**: Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, andenvironmental considerations.
4. **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.
5. **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.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge toassess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **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.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilitiesand norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member orleader in diverse teams, and in multidisciplinary settings.
10. **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, andgive and receive clear instructions.
11. **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.
12. **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.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Course Outcomes

Computer Network(CS 602)

CO1:	Understand the concept of various networking models and Able to Apply knowledge of the TCP/IP and OSI layering model to intelligently debug the networking problems.
CO2:	Describe and Analyze the methods to examine various data link layer design issues and data link protocols.
CO3:	Understand Medium Access Sub layer and different protocols working and Evaluate contention scheme for data services(ALOHA) and Local Area Networks(CSMA, CSMA/CD, CSMA/CA).
CO4:	Learn and Define network routing through algorithm and use IP addressing to create subnets for any specific requirements.
CO5:	Identify Application Layer protocol (such as HTTP, FTP, SMTP, DNS, Bit torrent) as per the requirements of the network application and work with available tools to create the working of these protocols.

Course	Course Outcomes	CO Attainment	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	Understand the concept of various networking models and Able to Apply knowledge of the TCP/IP and OSI layering model to intelligently debug the networking problems		3	2											2		
CO2	Describe and Analyze the methods to examine various data link layer design issues and data link protocols.		2	3			2									2	
CO3	Understand Medium Access Sub layer and different protocols working and Evaluate contention scheme for data services(ALOHA) and Local Area Networks(CSMA, CSMA/CD, CSMA/CA). tuples, dictionary and set build in container data types.		3		2										2	3	
CO4	Learn and Define network routing through algorithm and use IP addressing to create subnets for any specific requirements.		2		3		1				1		2			2	3

CO5	Identify Application Layer protocol (such as HTTP, FTP, SMTP, DNS, Bit torrent) as per the requirements of the network application and work with available tools to create the working of these protocols.			2	3		1				2				3	2	1

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

List Of Programs

INTRODUCTION TO COMPUTER NETWORK

A **computer network** is a system in which multiple computers are connected to each other to share information and resources.

Characteristics of a Computer Network

- Share resources from one computer to another.

  • Create files and store them in one computer, access those files from the other computer(s) connected over the network.

- Connect a printer, scanner, or a fax machine to one computer within the network and let other computers of the network use the machines available over the network.

  Following is the list of hardware's required to set up a computer network.

- Network Cables
- Distributors
- Routers
- Internal Network Cards
- External Network Cards

The features of a computer network are −

- **Sharing** − Computer networks enable sharing of files, software, hardware resources and computing capabilities.
- **Speed:** The communication speed among the components is fast enough to be comparable with a centralized system.
- **Scalability** − Sizes of computer networks dynamically increase with time. The networks have to be scalable so that they can evolve adequately for future deployments.

- **Integration** − All the components of the network work in a coordinated manner for a seamless user experience.
- **Security** − Networks allow security and access rights to the users for restricted sharing of resources and information.
- **Cost Effectiveness** − Networking reduces the deployment cost of hardware and software of a centralized system.

Elements of a Computer Network

The computer network involves the following networking elements −

Sender

The computer wants to give a message to some mainframe or computers.

Receiver

The computer takes the data from the sender who wishes to communicate.

Communication Medium

It is the medium over which the sender connects to the receiver. It can be used if the distance between sender and receiver is less, or it can be wireless if the distance between sender and receiver is much more.

Protocols

It is a set of rules which the sender and receiver will follow.

Hardware and Software Requirement

Hardware Requirement
RJ-45 connector, Climping Tool, Twisted pair Cable

Software Requirement
Command Prompt And Packet Tracer.

PROGRAM 1

Study of Different Type of LAN& Network Equipments.

LANs are classified as below according to the methods used for sharing data:

1. Ethernet:

- It is a network protocol that controls how data is transferred over a local area network.
- In this type of LAN, the user is able to transfer data at a rate of more than 10 megabits per second.
- Firstly, the system checks the medium used for the transfer of data; if the medium is available, then only the data transmission is done.
- It is used in wired local area networks.

2. **Token Ring:**

- It is a type of local area network in which all devices are connected in a ring arrangement.
- All the devices are connected in a circle, and they receive a token as per their requirements. A token keeps on rotating in the circular ring.
- A token is used to avoid collisions of data. It is of 3 bytes and keeps on traveling in the ring of servers or workstations.
- The details of three 1-byte fields of a free token frame are:
  - Starting Delimiter (SD): It signals the beginning of the token frame.
  - Access Control (AC): Contains the priority field, reservation field, a token bit, and a monitor bit.
  - Ending Delimiter (ED): It refers to the end of the token frame.

3. Token Bus:

- This is also a type of Local Area Network developed by IBM.
- Token Bus standard uses copper cables which are coaxial for connecting multiple devices to the main large computers or workstations. The coaxial cable acts as the common communication bus.

- In this protocol, also a token is created by this protocol to manage access for communication.
- Any computer that holds the token can transfer the data. The token is released when the station completes its data transmission or when a higher priority device needs to transmit (such as the mainframe).

Network equipments can be classified in following ways:

- **Repeaters:** repeater operates at the physical layer. Its job is to regenerate the signal over the same network before the signal becomes too weak or corrupted so as to extend the length to which the signal can be transmitted over the same network. An important point to be noted about repeaters is that they do not amplify the signal. When the signal becomes weak, they copy the signal bit by bit and regenerate it at the original strength. It is a 2 port device.

- **Hubs:** hub is basically a multiport repeater. A hub connects multiple wires coming from different branches, for example, the connector in star topology which connects different stations. Hubs cannot filter data, so data packets are sent to all connected devices. In other words, collision domain of all hosts connected through Hub remains one.

- **Bridge**: bridge operates at data link layer. A bridge is a repeater, with add on the functionality of filtering content by reading the MAC addresses of source and destination. It is also used for interconnecting two LANs working on the same protocol. It has a single input and single output port, thus making it a 2 port device.

- **Switch:** switch is a multiport bridge with a buffer and a design that can boost its efficiency (a large number of ports imply less traffic) and performance. A switch is a data link layer device. The switch can perform error checking before forwarding data, that makes it very efficient as it does not forward packets that have errors and forward good packets selectively to correct port only. In other words, switch divides collision domain of hosts, but broadcast domain remains same.

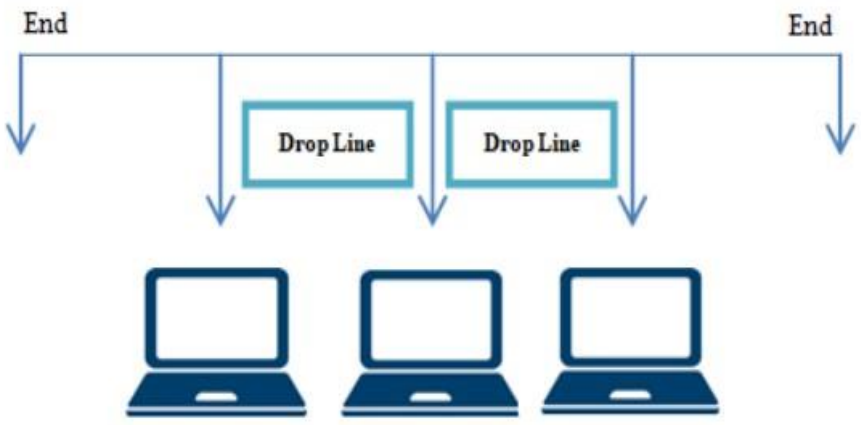
- **Routers:** router is a device like a switch that routes data packets based on their IP addresses. Router is mainly a Network Layer device. Routers normally connect LANs and WANs together and have a dynamically updating routing table based on which they make decisions on routing the data packets. Router divide broadcast domains of hosts connected through it.

Program-2

Study of standard Network topologies i.e. Star, Bus, Ring etc.

Bus Topology

In this type of topology, a single network cable runs in the building or campus called the bus or backbone.
Each device can be simply connected to the backbone.



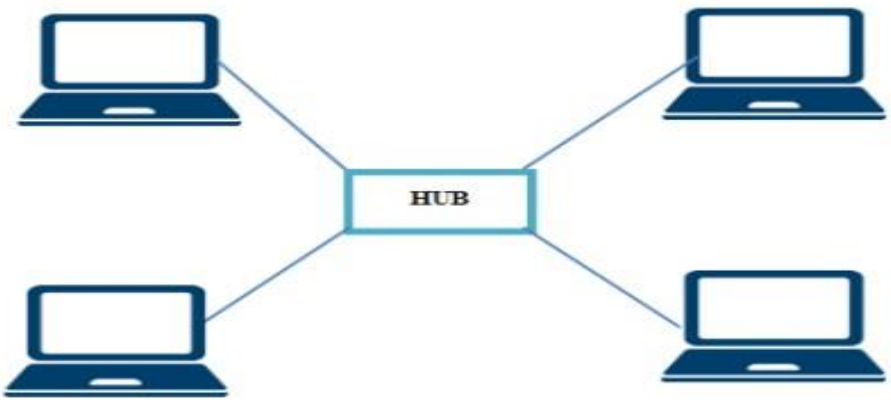
Advantages
This type of topology is reliable in very small networks and it is easy to use and understand. This also requires the least amount of cables to connect the computers together and it's easy to extend.

Disadvantages
Because of the fact that any computer in the network can transmit data at any time, network traffic can slow down a bus. Also, each connection between two cables can deteriorate the electrical signal. In this type of topology, the bus configuration can be difficult to find and can cause the whole network to stop working.

Star Topology

In this type of topology, each of the devices on a network connects to a central hub.

This hub acts as a conduit to transmit messages.



Advantages

In a star topology, if one of the computers fails, it does not affect the others and this also contributes to its good performance. The center of a star network is a good place to diagnose network faults and if one computer fails, the whole network is not disturbed. Replacement or removal of devices can be easily done.

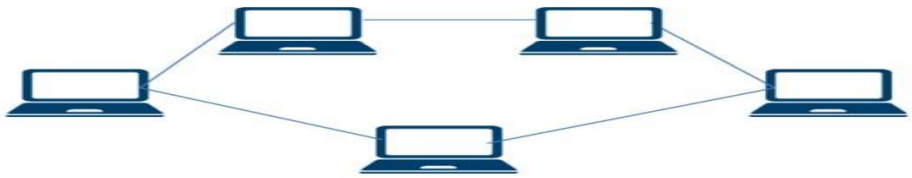
Disadvantages

This type of topology is expensive to install as it requires more cable, it costs more for all network cables must be pulled to one central point. This requires more cable length than other types of networking topologies.

Since the network depends upon the hub, or the central node, if it fails to operate, then the whole network also fails to operate.

Ring Topology

In a ring topology all of the nodes or devices are connected to the devices on each of their side.



Advantages

In this type of topology, the ring network offers high performance for a small number of workstations or for larger networks where each station has a similar workload. The ring networks can also span longer distances compared to other types of networks and these are also easily extendable. Unlike a bus topology, there is no signal loss in Ring topology because the tokens are data packets that are re-generated at each node.

Disadvantages

This type of topology costs much money, time and effort for its installation. If one computer fails, it can affect the whole network and it is difficult to find fault in the network. The whole system is disrupted when computers are being added or replaced in the network. This is much slower than an Ethernet network under normal load.

Program-3

LAN installation and configuration.

1. Identify the local services that you want available on the network. Identify network-attached printers, network disk drives, any server that will share printers or disks.

2. Identify how many devices will have to connect to the network. Each device, server or workstation will require a unique address.

3. Run cables to workstations where possible. A wired LAN will always get better performance and be more secure than a wireless LAN. Wherever possible, run a cable to servers, printers, IP phones or work locations. Run a cable to any area where you are likely to work. Use standard Ethernet cables or building wiring as installed according to the TIA-568 standard.

4. Select and purchase a switch or cable router. The simple secure way to connect to the Internet is to use a cable router. Many makes and models are available. If the model you choose does not have enough ports to connect all of your computers, then you will need to purchase a switch as well.

5. Configure the WAN port of the cable router. Configuration details will vary from vendor to vendor. Key information you will need to configure the WAN port will be supplied by your internet service provider.

6. Configure the LAN ports of your cable router. Most cable routers will act as a Dynamic Host Configuration Server, or DHCP server. This means that the router will give addresses to workstations automatically. Be certain that the address pool has enough addresses for all of the workstations. Make certain that there are enough addresses outside of the range for any hosts that need static addresses. For example, a network address with a mask of 255.255.255.0 has a total of 254 hosts. If the dynamic pool has 200 addresses available, that means the remaining 54 addresses are available to give printers or servers static addresses.

7. Connect the wires for the network. Workstations and servers can be connected with standard Ethernet cables. Connect the switch to the cable router LAN ports by using the up-link or straight port on the switch. If the switch does not have an up-link port, connect any standard port of the switch to a LAN port on the cable router with an Ethernet crossover cable. Ethernet rossover cables can be purchased at any electronics store.

8. Test the services and Internet connectivity. Test each of the workstations to ensure they can connect to the Internet and test any local servers and printers. Print test pages on the shared printers. Tests read and write permissions on shared file servers by copying files to the servers and copying files from the server to a workstation.

Program-4

Write a program to implement various types of error-correcting techniques.

Hamming Codes

```
#include <math.h>
#include <stdio.h>
 // Store input bits
int input[32];
 // Store hamming code
int code[32];
 int ham_calc(int, int);
void solve(int input[], int);

// Function to calculate bit for
// ith position
int ham_calc(int position, int c_l)
{
    int count = 0, i, j;
    i = position - 1;

    // Traverse to store Hamming Code
    while (i < c_l) {

        for (j = i; j < i + position; j++) {
            // If current boit is 1
            if (code[j] == 1)
                count++;
        }
         // Update i
        i = i + 2 * position;
    }
     if (count % 2 == 0)
        return 0;
    else
 return 1;
}

// Function to calculate hamming code
void solve(int input[], int n)
{
    int i, p_n = 0, c_l, j, k;
    i = 0;
```

```
// Find msg bits having set bit
   // at x'th position of number
   while (n > (int)pow(2, i) - (i + 1)) {
      p_n++;
      i++;
   }
    c_l = p_n + n;
    j = k = 0;

   // Traverse the msgBits
   for (i = 0; i < c_l; i++) {

      // Update the code
      if (i == ((int)pow(2, k) - 1)) {
         code[i] = 0;
         k++;
      }

      // Update the code[i] to the
      // input character at index j
      else {
         code[i] = input[j];
         j++;
      }
   }

   // Traverse and update the
   // hamming code
   for (i = 0; i < p_n; i++) {
       // Find current position
      int position = (int)pow(2, i);
      // Find value at current position
      int value = ham_calc(position, c_l);
       // Update the code
      code[position - 1] = value;
   }
    // Print the Hamming Code
   printf("\nThe generated Code Word is: ");
   for (i = 0; i < c_l; i++) {
      printf("%d", code[i]);
   }
}
```

```
// Driver Code
void main()
{
    // Given input message Bit
    input[0] = 0;
    input[1] = 1;
    input[2] = 1;
    input[3] = 1;

    int N = 4;

    // Function Call
    solve(input, N);
}
```

Program-5

Write a program to implement various types of framing methods.
Bit Stuffing

```
#include <bits/stdc++.h>
#include <iostream>
using namespace std;

// Function for bit stuffing
void bitStuffing(int N, int arr[])
{
        // Stores the stuffed array
    int brr[30];
     // Variables to traverse arrays
    int i, j, k;
    i = 0;
    j = 0;
     // Stores the count of consecutive ones
    int count = 1;
     // Loop to traverse in the range [0, N)
    while (i < N)
    {
              // If the current bit is a set bit
        if (arr[i] == 1)
        {
            // Insert into array brr[]
            brr[j] = arr[i];

            // Loop to check for
            // next 5 bits
            for(k = i + 1; arr[k] == 1 && k < N && count < 5;
                k++)
            {
                j++;
                brr[j] = arr[k];
                count++;
                // If 5 consecutive set bits
                // are found insert a 0 bit
                if (count == 5)
                {
                    j++;
                    brr[j] = 0;
```

```
                }
                i = k;
            }
        }

        // Otherwise insert arr[i] into
        // the array brr[]
        else
        {
            brr[j] = arr[i];
        }
        i++;
        j++;
    }
     // Print Answer
    for(i = 0; i < j; i++)
        cout << brr[i];
}

// Driver code
int main()
{
    int N = 6;
    int arr[] = { 1, 1, 1, 1, 1, 1 };

    bitStuffing(N, arr);;
    return 0;
}
```

Byte Stuffing

```
#include<stdio.h>
#include<string.h>
void main(){
char frame[50][50],str[50][50];
 char flag[10];
strcpy(flag,"flag");
 char esc[10];
 strcpy(esc,"esc");
 int i,j,k=0,n;
  strcpy(frame[k++],"flag");
 printf("Enter no.of String :\t");
 scanf("%d",&n);
```

```
printf("Enter String \n");
for(i=0;i<=n;i++)
  {
 gets(str[i]);
  }
printf("You entered :\n");
for(i=0;i<=n;i++)
   {
 puts(str[i]);
 }
printf("\n");
for(i=1;i<=n;i++)
  {
     if(strcmp(str[i],flag)!=0 && strcmp(str[i],esc)!=0)
          {
               strcpy(frame[k++],str[i]);
     }
     else
 {
          strcpy(frame[k++],"esc");
          strcpy(frame[k++],str[i]);
        }
  }
 strcpy(frame[k++],"flag");
//frame[k++]='\0';
printf("------------------------------\n");
printf("Byte stuffing at sender side:\n\n");
printf("------------------------------\n");
for(i=0;i<k;i++)
  {
 printf("%s\t",frame[i]);

}
}
```

Program-6

Study of Tool Command Language (TCL).

Ans. TCL is a tool command language, commands are the most vital part of the language. TCL commands are built in-to with each having predefined function.

1.Command Substitution: In command substitution, square brackets are used to evaluate the scripts inside the square brackets.
Example- puts [expr 1+6+9]

2.Variable Substitution: In variable substitution, $ is used before the variable name and this return the content of the variable. Example- set a 3
puts $a

3.Backslash Substitution: These are commonly called escape sequence; with each backslash followed by a letter.
Example- puts "hello\world"

4.Set Command: we can directly assign value to variable there is no step of declaration in it there can be internal representation for these kinds of objects. Example- set my1 18
puts $my1

5.Info Command: the info command returns information. Example- puts [info host]
puts [info var]

6.Unset Command: The unset command is used to destroy variable. Example- set x 23
puts $x puts [set x] Unset x
puts [info exists x]

7. Command line arguments: TCL scripts like any other script can take command line arguments.

$argc- the number of arguments passed to the script.

$argv- the list of arguments

Example- puts "the script has $argc arguments" puts "the list of arguments $argv"

Program-7

Study and installation of Standard Network Simulator : NS-2, NS-3, OpNet, QualNet etc.

Steps is to install NS2

1. Download NS2 from following link: https://www.isi.edu/nsnam/dist/ns-allinone-2.28.tar.gz

2. Decompress the file use winrar. Copy the decompressed folder the Cygwin installation directory under the subdirectory home. It will be C:\cygwin\home\system_name : where system_name is name of your system in above Cygwin installation this path will be C:\Cygwin\home\sys27

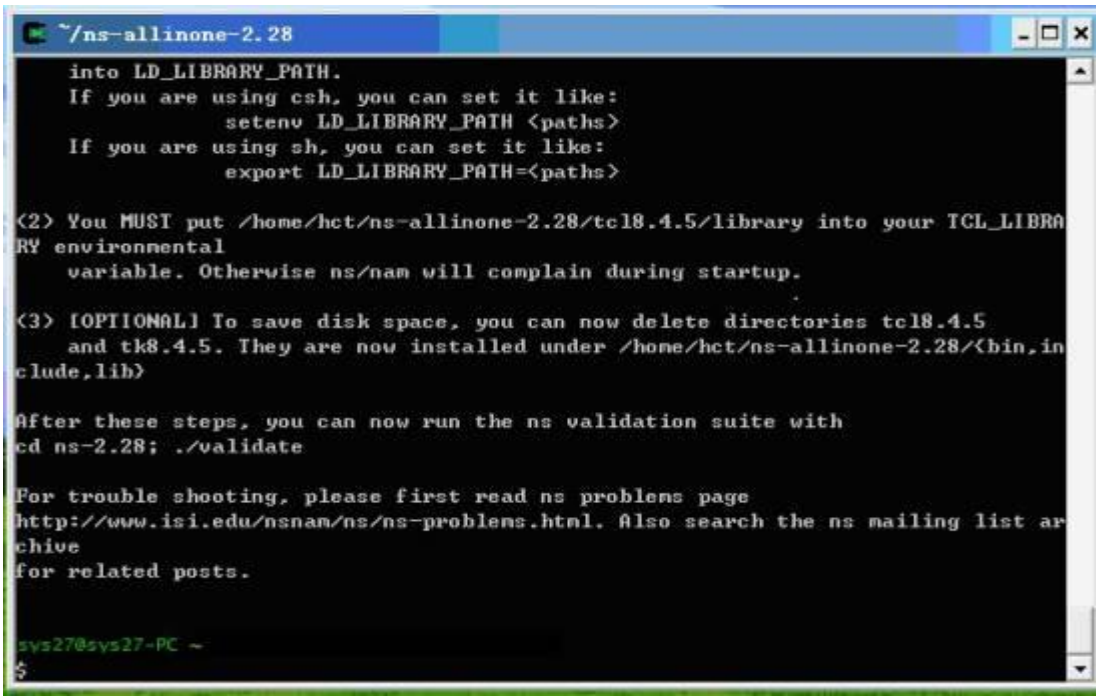
3. Run Cygwin from desktop and change the directory to folder you copied just now in step 2 command to change directory:cd /home/sys27/ns-allinone-2.28

NOTE: please change sys27 to name of your system

4. To start installation type following command:**"./install"**(WITHOUT qoutes)

This will began the installation process if any Cygwin package is missing it will be reported to you if so the run Cygwin setu.exe and install the missing package and start again from step 2.

Installation is a long process and take quite some time once it is finished you will get a screen as shown below:



5. Add following lines to the .bashrc

```
export NS_HOME=/home/sys27/ns-allinone-2.28

export PATH=$NS_HOME/nam-1.11:$NS_HOME/tcl8.4.5/unix:$NS_HOME/tk8.4.5/unix:$NS_HOME/bin:$PATH

export LD_LIBRARY_PATH=$NS_HOME/tcl8.4.5/unix:$NS_HOME/tk8.4.5/unix:$NS_HOME/otcl-1.9:$NS_HOME/lib:$LD_LIBRARY_PATH

export TCL_LIBRARY=$NS_HOME/tcl8.4.5/library
```

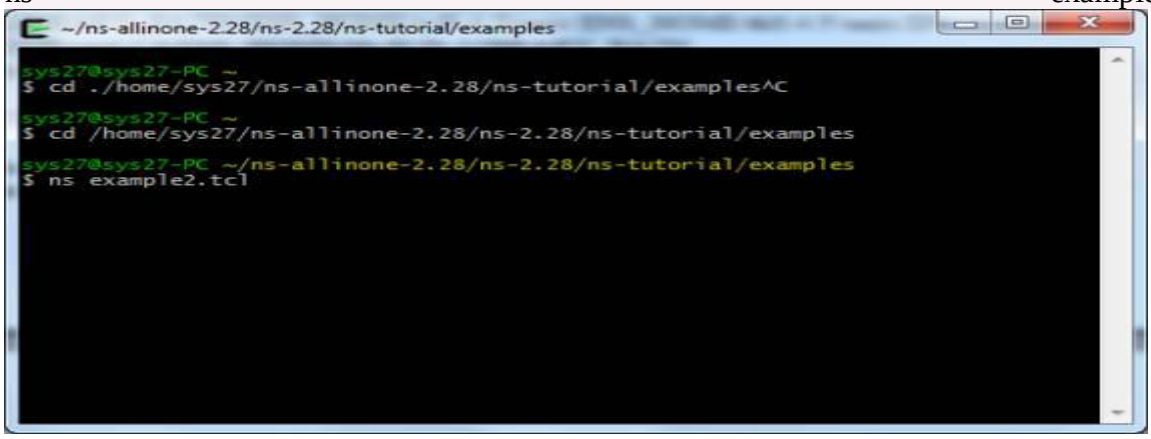
NOTE: replace sys27 with your system name

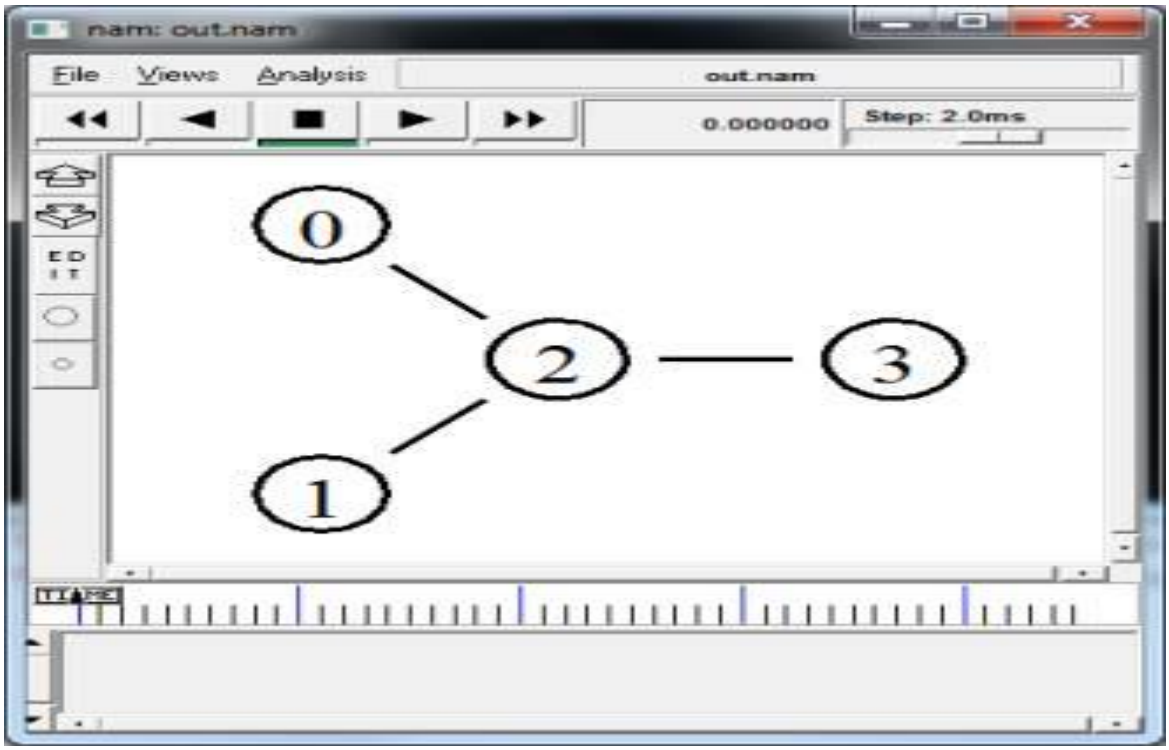
6. To check if NS2 is installed correctly you can run one sample example given in ns-tutorials folder

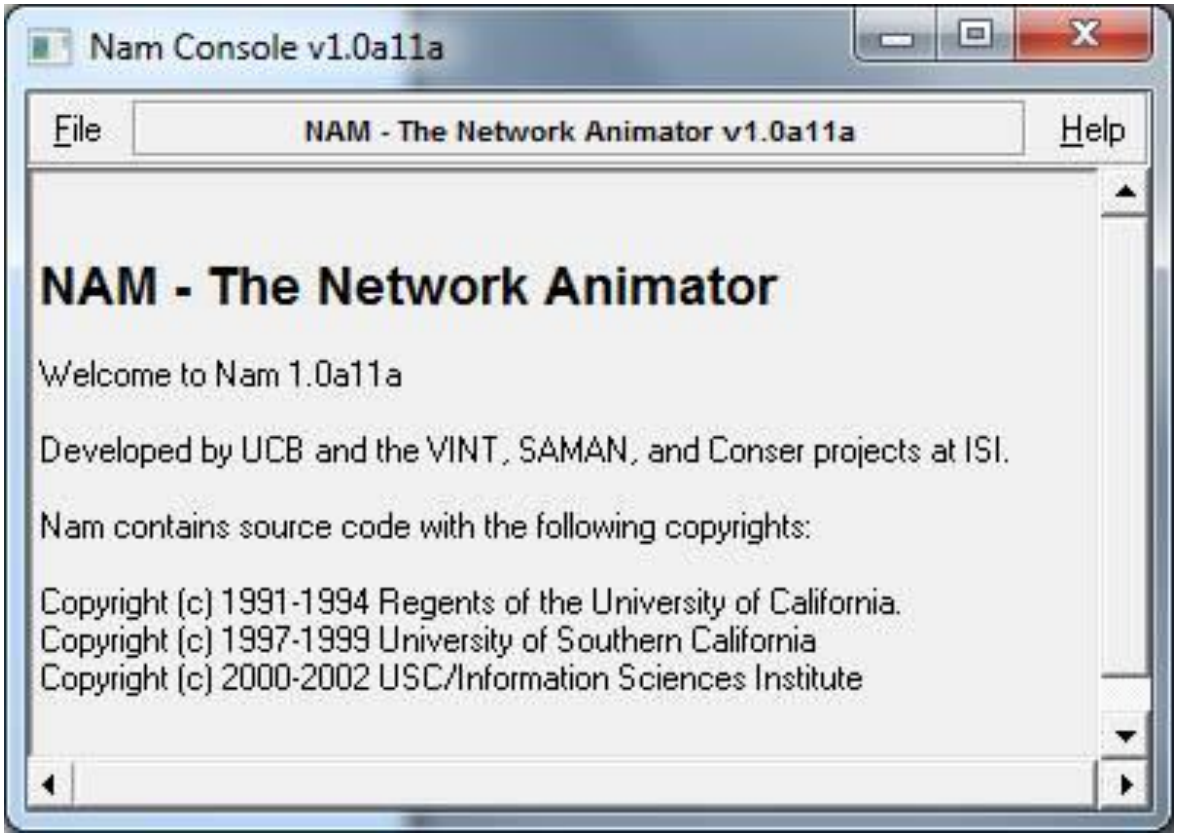
To run the example change the directory to examples folder: cd ./home/sys27/ns-allinone-2.28/ns-tutorial/examples

Then type following command:

```
ns                                                    example2.tcl
```




nam: out.nam
File
Views
Analysis
out.nam
0.000000
Step: 2.0ms
0
1
2
3


Nam Console v1.0a11a
File
NAM - The Network Animator v1.0a11a
Help
NAM - The Network Animator
Welcome to Nam 1.0a11a
Developed by UCB and the VINT, SAMAN, and Conser projects at ISI.
Nam contains source code with the following copyrights:
Copyright (c) 1991-1994 Regents of the University of California.
Copyright (c) 1997-1999 University of Southern California
Copyright (c) 2000-2002 USC/Information Sciences Institute

NS-3 Installation

Follow from Step 1 to Step 7 in order to create NS3 using Simulation projects. Quick guide to create NS3 simulation in Ubuntu 18.04. Reach us , if you want an customize NS3 simulation projects works for scholars.

1. **Update libraries**

   Initially,open the terminal by press ctrl+alt+T buttons or search from the installed software list., update the system related libraries for the further new updated software installations, by using the commands sudo apt update and sudo apt upgrade

2. **Install the supported packages**

   Next install the supported packages for the new NS3 installation. For the supported package installation we use the following command. sudo apt-get install build-essential autoconf automake libxmu-dev python-pygoocanvas python-pygraphviz cvs mercurial bzr git cmake p7zip-full python-matplotlib python-tk python-dev python-kiwi python-gnome2 python-gnome2-desktop-dev python-rsvg qt4-dev-tools qt4-qmake qt4-qmake qt4-default gnuplot-x11 wireshark

3. **Download the ns3 package**

   Download the ns3 package from https://www.nsnam.org, the downloaded file looks like ns-allinone-3.30.tar.bz2.
   To unzip the Right click over the above file and extract it to the folder (/home/user name/). Most preferred place to install is to put it in the home folder.

4. **Build the NS-3 Package**

   For build the Ns3 package , we perform the process of change the location by using the command , cd ns-allinone-3.30/ and also build the package by using the command ./build.py

5. **Build processing**

   We perform the project development of change the location by using the command of step 6.

6. **Command building**

   By using the command , cd ns-allinone-3.30/ and also build the package by using the command ./build.py

7. **Command Building Result**

   Let we look for Command Building Result in NS3 to build the package by using the command ./build.py

Opnet Installation

1) Install visual studio 2012 (just the C++ features are needed)
2) add/edit the following Environment Variables
Name: DevEnvDir
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE\
Name: Framework35Version
Value: v3.5
Name: FrameworkDir
Value: C:\WINDOWS\Microsoft.NET\Framework64\
Name: FrameworkVersion
Value: v2.0.50727
Name: INCLUDE
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\atlmfc\include;C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\include;C:\Program Files (x86)\Windows Kits\8.0\Include\um;C:\Program Files (x86)\Windows Kits\8.0\Include\shared;
Name: LIB
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\atlmfc\lib\amd64;C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\lib\amd64;C:\Program Files (x86)\Windows Kits\8.0\Lib\win8\um\x64;
Name: LIBPATH
Value:
C:\WINDOWS\Microsoft.NET\Framework64\v3.5;C:\WINDOWS\Microsoft.NET\Framework64\v2.0.50727;C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\atlmfc\lib\amd64;C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\lib\amd64;
Edit this
Name: Path
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\bin\amd64\;C:\Windows\Microsoft.NET\Framework64\v3.5\;C:\WINDOWS\Microsoft.NET\Framework64\v2.0.50727;C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\vcpackages;C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\IDE;C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools;C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin\x64;C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bin
Name: VCINSTALLDIR
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\VC\
Name: VSINSTALLDIR
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\
Name: WindowsSdkDir
Value: C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\
Make sure that this exist
Name: VS110COMNTOOLS
Value: C:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools\
3) Install the modeler
4) Install the modeler docs
5) Install the models

Program-8

Study and Installation of One(Opportunistic Network Environment) Simulator for High Mobility Networks

The ONE is a Opportunistic Network Environment simulator which provides a powerful tool for generating mobility traces, running DTN messaging simulations with different routing protocols, and visualizing both simulations interactively in real-time and results after their completion.

Quick start
Running
ONE can be started using the included one.bat (for Windows) or one.sh (for Linux/Unix) script. Following examples assume you're using the Linux/Unix script (just replace .sh by .bat for Windows).

Synopsis:
one.sh [-b] [conf-file] [run-index (count)]

Options:
-b Run simulation in batch mode. Doesn't start GUI but prints information about the progress to terminal.

Parameters: conf-file: The configuration file where simulation parameters are read from.

run-index: If running in GUI mode (without -b option), you can give which
run index to use in the given run. In batch mode, the last parameter is the run index count, i.e., how many runs with different run indeces are done.

Configuring
All simulation parameters are given using configuration files. These files are normal text files that contain key-value pairs. Syntax for most of the variables is:
Namespace.key = value

I.e., the key is (usually) prefixed by a namespace, followed by a dot, and then key name. Key and value are separated by equals-sign. Namespaces start with capital letter and both namespace

and keys are written in CamelCase (and are case sensitive). Namespace defines (loosely) the part of the simulation environment where the setting has effect on. Many, but not all, namespaces are equal to the class name where they are read. Especially movement models, report modules and routing modules follow this convention.

Numeric values use '.' as the decimal separator and can be suffixed with kilo (k) mega (M) or giga (G) suffix. Boolean settings accept "true", "false", "0", and "1" as values.

Many settings define paths to external data files. The paths can be relative or absolute but the directory separator must be '/' in both Unix and Windows environment.

Some variables contain comma-separated values, and for them the syntax is:
Namespace.key = value1, value2, value3, etc.

For run-indexed values the syntax is:
Namespace.key = [run1value; run2value; run3value; etc]
I.e., all values are given in brackets and values for different run are separated by semicolon. Each value can also be a comma-separated value.

For more information about run indexing, go to section "Run indexing".

Setting files can contain comments too. A comment line must start with "#" character. Rest of the line is skipped when the settings are read. This can be also useful for disabling settings easily.

Some values (scenario and report names at the moment) support "value filling". With this feature, you can construct e.g., scenario name dynamically from the setting values. This is

especially useful when using run indexing. Just put setting key names in the value part prefixed and suffixed by two percent (%) signs. These placeholders are replaces by the current setting value from the configuration file. See the included snw_comparison_settings.txt for an example.

File "default_settings.txt" is always read and the (optional) configuration file given as parameter can define more settings or override some (or even all) settings in the default settings file. The idea is that you can define in the default file all the settings that are common for all the simulations and run different, specific, simulations using different configuration files. If your

simulations don't have any common parameters (which is highly unlikely) just provide an empty default settings file. If you want to use more than one default configuration set, just create separate folders for all configuration sets and provide one default settings file for each folder.

Run indexing

Run indexing is a feature that allows you to run large amounts of different configurations using only single configuration file. The idea is that you provide an array of settings (using the syntax described above) for the variables that should be changed between runs. For example, if you want to run the simulation using five different random number generator seeds for movement models, you can define in the settings file the following:

MovementModel.rngSeed = [1; 2; 3; 4; 5]

Now, if you run the simulation using command: one.sh -b my_config.txt 5

you would run first using seed 1 (run index 0), then another run using seed 2 etc. Note that you have to run it using batch mode (-b option) if you want to use different values. Without the batch mode flag the last parameter is the run index to use when running in GUI mode.

Run indexes wrap around: used value is the value at index (runIndex % arrayLength). Because of wrapping, you can easily run large amount of permutations easily. For example, if you define two key-value pairs:

key1 = [1; 2] key2 = [a; b; c]

and run simulation using run-index count 6, you would get all permutations of the two values (1,a; 2,b; 1,c; 2,a; 1,b; 2,c). This naturally works with any amount of arrays. Just make sure that the smallest common nominator of all array sizes is 1 (e.g., use arrays whose sizes are primes) -- unless you don't want all permutations but some values should be paired.

Movement models

Movement models govern the way nodes move in the simulation. They provide coordinates, speeds and pause times for the nodes. The basic installation contains 5 movement models: random waypoint, map based movement, shortest path map based movement, map route movement and external movement. All models, except external movement, have configurable speed and pause time distributions. A minimum and maximum values can be given and the

movement model draws uniformly distributed random values that are within the given range. Same applies for pause times. In external movement model the speeds and pause times are interpreted from the given data.

When a node uses the random waypoint movement model (RandomWaypoint), it is given a random coordinate in the simulation area. Node moves directly to the given destination at constant speed, pauses for a while, and then gets a new destination. This continues throughout the simulations and nodes move along these zig-zag paths.

Map-based movement models constrain the node movement to predefined paths. Different types of paths can be defined and one can define valid paths for all node groups. This way e.g., cars can be prevented from driving indoors or on pedestrian paths.

The basic map-based movement model (MapBasedMovement) initially distributes the nodes between any two adjacent (i.e., connected by a path) map nodes and then nodes start moving from adjacent map node to another. When node reaches the next map node, it randomly selects the next adjacent map node but chooses the map node where it came from only if that is the only option (i.e., avoids going back to where it came from). Once node has moved trough 10-100 map nodes, it pauses for a while and then starts moving again.

The more sophisticated version of the map-based movement model (ShortestPathMapBasedMovement) uses Dijkstra's shortest path algorithm to find its way trough the map area. Once a node reaches its destination, and has waited for the pause time, a new random map node is chosen and node moves there using the shortest path that can be taken using only valid map nodes.

For the shortest path based movement models, map data can also contain Points Of Interest (POIs). Instead of selecting any random map node for the next destination, the movement model can be configured to give a POI belonging to a certain POI group with a configurable probability. There can be unlimited amount of POI groups and all groups can contain any amount of POIs. All node groups can have different probabilities for all POI groups. POIs can be used to model e.g., shops, restaurants and tourist attractions.

Route based movement model (MapRouteMovement) can be used to model nodes that follow certain routes, e.g. bus or tram lines. Only the stops on the route have to be defined and then

the nodes using that route move from stop to stop using shortest paths and stop on the stops for the configured time.

All movement models can also decide when the node is active (moves and can be connected to) and when not. For all models, except for the external movement, multiple simulation time intervals can be given and the nodes in that group will be active only during those times.

All map-based models get their input data using files formatted with a subset of the Well Known Text (WKT) format. LINESTRING and MULTILINESTRING directives of WKT files are supported by the parser for map path data. For point data (e.g. for POIs), also the POINT directive is supported. Adjacent nodes in a (MULTI)LINESTRING are considered to form a path and if some lines contain some vertex(es) with exactly the same coordinates, the paths are joined from those places (this is how you create intersections). WKT files can be edited and generated from real world map data using any suitable Geographic Information System (GIS) program. The map data included in the simulator distribution was converted and edited using the free, Java based OpenJUMP GIS program.

Different map types are defined by storing the paths belonging to different types to different files. Points Of Interest are simply defined with WKT POINT directive and POI groups are defined by storing all POIs belonging to a certain group in the same file. All POIs must also be part of the map data so they are accessible using the paths. Stops for the routes are defined with

LINESTRING and the stops are traversed in the same order they appear in the LINESTRING. One WKT file can contain multiple routes and they are given to nodes in the same order as they appear in the file.

The experimental movement model that uses external movement data (ExternalMovement) reads timestamped node locations from a file and moves the nodes in the simulation accordingly. See javadocs of ExternalMovementReader class from input package for details of the format. A suitable, experimental converter script (transimsParser.pl) for TRANSIMS data is included in the toolkit folder.

The movement model to use is defined per node group with the "movementModel" setting. Value of the setting must be a valid movement model class name from the movement package. Settings that are common for all movement models are read in the MovementModel class and

movement model specific settings are read in the respective classes. See the javadoc documentation and example configuration files for details.

Routing modules and message creation

Routing modules define how the messages are handled in the simulation. Six different active routing modules (First Contact, Epidemic, Spray and Wait, Direct delivery, PRoPHET and MaxProp) and also a passive router for external routing simulation are included in the package. The active routing modules are implementations of the well known routing algorithms for DTN routing. See the classes in routing package for details.

Passive router is made especially for interacting with other (DTN) routing simulators or running simulations that don't need any routing functionality. The router doesn't do anything unless commanded by external events. These external events are provided to the simulator by a class that implements the EventQueue interface.

The current release includes two classes that can be used as a source of message events: ExternalEventsQueue and MessageEventGenerator. The former can read events from a file that can be created by hand, with a suitable script (e.g., createCreates.pl script in the toolkit folder), or by converting e.g., dtnsim2's output to suitable form. See StandardEventsReader class from input package for details of the format. MessageEventGenerator is a simple

message generator class that creates uniformly distributed message creation patterns with configurable message creation interval, message size and source/destination host ranges.

The toolkit folder contains an experimental parser script (dtnsim2parser.pl) for dtnsim2's output (there used to be a more capable Java-based parser but it was discarded in favor of this more easily extendable script). The script requires a few patches to dtnsim2's code and those can be found from the toolkit/dtnsim2patches folder.

The routing module to use is defined per node group with the setting "router". All routers can't interact properly (e.g., PRoPHET router can only work with other PRoPHET routers) so usually it makes sense to use the same (or compatible) router for all groups.

Reports

Reports can be used to create summary data of simulation runs, detailed data of connections and messages, files suitable for post-processing using e.g., Graphviz (to create graphs) and also to interface with other programs. See javadocs of report-package classes for details.

There can be any number of reports for any simulation run and the number of reports to load is defined with "Report.nrofReports" setting. Report class names are defined with "Report.reportN" setting, where N is an integer value starting from 1. The values of the settings must be valid report class names from the report package. The output directory of all reports (which can be overridden per report class with the "output" setting) must be defined with Report.reportDir -setting. If no "output" setting is given for a report class, the resulting report file name is "ReportClassName_ScenarioName.txt".

All reports have many configurable settings which can be defined using ReportClassName.settingKey -syntax. See javadocs of Report class and specific report classes for details (look for "setting id" definitions).

Host groups

A host group is group of hosts (nodes) that shares movement and routing module settings.

Different groups can have different values for the settings and this way they can represent different types of nodes. Base settings can be defined in the "Group" namespace and different node groups can override these settings or define new settings in their specific namespaces (Group1, Group2, etc.).

The settings

There are plenty of settings to configure; more than is meaningful to present here. See javadocs of especially report, routing and movement model classes for details. See also included settings files for examples.

Perhaps the most important settings are the following.

Scenario settings:

Scenario.name

Name of the scenario. All report files are by default prefixed with this.

Scenario.simulateConnections
Should connections be simulated. If you're only interested in movement modeling, you can disable this to get faster simulation. Usually you want this to be on.

Scenario.updateInterval
How many seconds are stepped on every update. Increase this to get faster simulation, but then you'll lose some precision. Values from 0.1 to 2 are good for simulations.

Scenario.endTime
How many simulated seconds to simulate.

Scenario.nrofHostGroups
How many hosts group are present in the simulation. Host group settings (used in Group or

GroupN namespace):

groupID
Group's identifier (a string or a character). Used as the prefix of host names that are shown in the GUI and reports. Host's full name is groupID+networkAddress.

nrofHosts
Number of hosts in this group.

transmitRange
Range (meters) of the hosts' radio devices.

transmitSpeed
Transmit speed of the hosts' radio devices (bytes per second).

movementModel
The movement model all hosts in the group use. Must be a valid class (one that is a subclass of MovementModel class) name from the movement package.

waitTime
Minimum and maximum (two comma-separated decimal values) of the wait time interval (seconds). Defines how long nodes should stay in the same place after reaching the destination of the current path. A new random value within the interval is used on every stop. Default value

is 0,0.

speed
Minimum and maximum (two comma-separated decimal values) of the speed interval (m/s). Defines how fast nodes move. A new random value is used on every new path. Default value is 1,1.

bufferSize
Size of the nodes' message buffer (bytes). When the buffer is full, node can't accept any more messages unless it drops some old messages from the buffer.

router
Router module which is used to route messages. Must be a valid class (subclass of Report class) name from routing package.

activeTimes
Time intervals (comma-separated simulated time value tuples: start1, end1, start2, end2, ...) when the nodes in the group should be active. If no intervals are defined, nodes are active all the time.

msgTtl
Time To Live (simulated minutes) of the messages created by this host group. Nodes (with active routing module) check every one minute whether some of their messages' TTLs have expired and drop such messages. If no TTL is defined, infinite TTL is used.

Group and movement model specific settings (only meaningful for certain movement models):

pois
Points Of Interest indexes and probabilities (comma-separated index-probability tuples: poiIndex1, poiProb1, poiIndex2, poiProb2, ...).
Indexes are integers and probabilities are decimal values in the range of 0.0-1.0. Setting defines the POI groups where the nodes in this host group can choose destinations from and the probabilities for choosing a certain POI group. For example, a (random) POI from the group defined in the POI file1 (defined with PointsOfInterest.poiFile1 setting) is chosen with the probability poiProb1. If the sum of all probabilities is less than 1.0, a probability of choosing

any random map node for the next destination is (1.0
- theSumOfProbabilities). Setting can be used only with ShortestPathMapBasedMovement -based movement models.

okMaps
Which map node types (refers to map file indexes) are OK for the group (comma-separated list of integers). Nodes will not travel trough map nodes that are not OK for them. As default, all map nodes are OK. Setting can be used with any MapBasedMovent -based movement model.

routeFile
If MapRouteMovement movement model is used, this setting defines the route file (path) where the route is read from. Route file should contain LINESTRING WKT directives. Each vertex in a LINESTRING represents one stop on the route.

routeType
If MapRouteMovement movement model is used, this setting defines the routes type. Type can be either circular (value 1) or ping-pong (value 2). See movement.map.MapRoute class for details.

Movement model settings:

MovementModel.rngSeed
The seed for all movement models' random number generator. If the seed and all the movement model related settings are kept the same, all nodes should move the same way in different simulations (same destinations and speed & wait time values are used).

MovementModel.worldSize
Size of the simulation world in meters (two comma separated values: width, height).

PointsOfInterest.poiFileN
For ShortestPathMapBasedMovement -based movement models, this setting defines the WKT files where the POI coordinates are read from. POI coordinates are defined using the POINT WKT directive. The "N" in the end of the setting must be a positive integer (i.e., poiFile1, poiFile2, ...).

MapBasedMovement.nrofMapFiles
How many map file settings to look for in the settings file.

MapBasedMovement.mapFileN
Path to the Nth map file ("N" must be a positive integer). There must be at least nrofMapFiles separate files defined in the configuration files(s). All map files must be WKT files with LINESTRING and/or MULTILINESTRING WKT directives. Map files can contain POINT directives too, but those are skipped. This way the same file(s) can be used for both POI and map data. By default the map coordinates are translated so that the upper left corner of the map is at coordinate point (0,0). Y-coordinates are mirrored before translation so that the map's north points up in the playfield view. Also all POI and route files are translated to match to the map data transformation.

Report settings:

Report.nrofReports
How many report modules to load. Module names are defined with settings "Report.report1", "Report.report2", etc. Following report settings can be defined for all reports (using Report name space) or just for certain reports (using ReportN name spaces).

Report.reportDir
Where to store the report output files. Can be absolute path or relative to the path where the simulation was started. If the directory doesn't exists, it is created.

Report.warmup
Length of the warm up period (simulated seconds from the start). During the warm up the report modules should discard the new events. The behavior is report module specific so check the (java)documentation of different report modules for details.

Event generator settings:

Events.nrof
How many event generators are loaded for the simulation. Event generator specific settings (see below) are defined in EventsN namespaces (so
Events1.settingName configures a setting for the 1st event generator etc.).

EventsN.class
Name of the generator class to load (e.g., ExternalEventsQueue or MessageEventGenerator). The class must be found from the input package.

For the ExternalEventsQueue you must at least define the path to the external events file (using setting "filePath"). See input.StandardEventsReader class' javadocs for information about different external events.

Other settings:
Optimization.randomizeUpdateOrder
Should the order in which the nodes' update method is called be randomized. Call to update causes the nodes to check their connections and also update their routing module. If set to false, node update order is the same as their network address order. With randomizing, the order is different on every time step.

GUI
The GUI's main window is divided into three parts. The main part contains the playfield view (where node movement is displayed) and simulation and GUI control and information. The right part is used to select nodes and the lower part is for logging and breakpoints.

The main part's topmost section is for simulation and GUI controls. The first field shows the current simulation time. Next field shows the simulation speed (simulated seconds per second). The following four buttons are used to pause, step, fast forward, and fast forward simulation to given time. Pressing step-button multiple times runs simulation step-by-step. Fast forward (FFW) can be used to skip uninteresting parts of simulation. In FFW, the GUI update speed is set to a large value. Next drop-down is used to control GUI update speed. Speed 1 means that GUI is updated on every simulated second. Speed 10 means that GUI is updated only on every 10th second etc. Negative values slow down the simulation. The following drop-down controls the zoom factor. The last button saves the current view as a png-image.

Middle section, i.e., the playfield view, shows the node placement, map paths, node identifiers, connections among nodes etc. All nodes are displayed as small rectangles and their radio range is shown as a green circle around the node. Node's group identifier and network address (a number) are shown next to each node. If a node is carrying messages, each message is represented by a green or blue filled rectangle. If node carries more than 10 messages, another column of rectangles is drawn for each 10 messages but every other rectangle is now red. You can center the view to any place by clicking with mouse button on the play field. Zoom factor

can also be changed using mouse wheel on top of the playfield view. The right part of main window is for choosing a node for closer inspection. Simply clicking a button shows the node info in main parts lower section. From there more information can be displayed by selecting one of the messages the node is carrying (if any) from the drop-down menu. Pressing the "routing info" button opens a new window where information about the routing module is displayed. When a node is chosen, the playfield view is also centered on that node and the current path the node is traveling is shown in red.

Logging (the lowest part) if divided to two sections, control and log. From the control part you can select what kind of messages are shown in the log. You can also define if simulation should be paused on certain type of event (using the check boxes in the "pause" column). Log part displays time stamped events. All nodes and message names in the log messages are buttons and you can get more information about them by clicking the buttons.

Toolkit

The simulation package includes a folder called "toolkit" that contains scripts for generating input and processing the output of the simulator. All (currently included) scripts are written with Perl (http://www.perl.com/) so you need to have it installed before running the scripts. Some post processing scripts use gnuplot (http://www.gnuplot.info/) for creating graphics. Both of the programs are freely available for most of the Unix/Linux and Windows environments. For Windows environment, you may need to change the path to the executables for some of the scripts.

getStats.pl

This script can be used to create bar-plots of various statistics gathered by the MessageStatsReport -report module. The only mandatory option is "-stat" which is used to define the name of the statistics value that should be parsed from the report files (e.g., "delivery_prob" for message delivery probabilities). Rest of the parameters should be MessageStatsReport output filenames (or paths). Script creates three output files: one with values from all the files, one with the gnuplot commands used to create the graphics and finally an image file containing the graphics. One bar is created to the plot for each input file. The title for each bar is parsed from the report filename using the regular expression defined with "-label" option. Run getStats.pl with "-help" option for more help.

ccdfPlotter.pl

Script for creating Complementary(/Inverse) Cumulative Distribution Function plots (using gluplot) from reports that contain time-hitcount-tuples. Output filename must be defined with the "-out" option and rest of the parameters should be (suitable) report filenames. "-label" option can be used for defining label extracting regular expression (similar to one for the getStats script) for the legend.

createCreates.pl

Message creation pattern for the simulation can be defined with external events file. Such a file can be simply created with any text editor but this script makes it easier to create a large amount of messages. Mandatory options are the number of messages ("-nrof"), time range ("-time"), host address range ("-hosts") and message size range ("-sizes"). The number of messages is simply an integer but the ranges are given with two integers with a colon (:) between them. If hosts should reply to the messages that they receive, size range of the reply messages can be defined with "-rsizes" option. If a certain random number generator seed should be used, that can be defined with "-seed" option. All random values are drawn from a uniform distribution with inclusive minimum value and exclusive maximum value. Script outputs commands that are suitable for external events file's contents. You probably want to redirect the output to some file.

Program-9

Configure 802.11 WLAN

Configure the 802.11b Interface Unit settings related to communication.

- This page is displayed when an 802.11b Interface Unit is installed.

IEEE 802.11b (Wireless LAN):
Shows whether the 802.11b Interface Unit is enabled or disabled.

Communication Mode:
Select a communication mode for the 802.11b Interface Unit from the drop-down menu.

- 802.11 Ad hoc mode:
  A method to communicate with other computers from peer to peer without having an access point. The same communication channel and SSID must be used for all computers.
- Ad hoc mode:
  A method to communicate with other computers from peer to peer without having an access point. The same communication channel must be used for all computers.
- Infrastructure mode:
  A method to communicate with other computers via an access point. The SSID must be the same as the one set for the access point.

Current SSID:
Shows the current SSID(Service Set ID).
This is called a Service Set ID and is used in the connection between the wireless LAN client and the access point. Only a wireless LAN client and an access point that have the same SSID can transmit to each other.

- This information appears if [802.11 Ad hoc mode] or [Infrastructure mode] is selected in the [Communication Mode] list.

Selected SSID:
Input an SSID using alphanumeric characters. Up to 32 characters can be used.
If the SSID is not set and the transmission mode is "802.11 Ad hoc mode", the string ASSID is used as the default SSID.

Channel:
If [Ad hoc mode] and [802.11 Ad hoc mode] is selected in the [Communication Mode] list, select a communication channel from the drop-down menu.The channel is a classification of radio frequency used for the wireless LAN. Computers using the same channel can communicate with each other.

WEP Setting:
Drop Down Title that controls WEP Setting Status Either "Enable" or "Disable" is displayed. WEP is an encryption method that protects wireless data communication. To communicate by encrypting data, both computers that are transmitting and receiving the data must be set with the same WEP Key.
If you want to enable WEB encryption, select [Enable], and then make the [WEP Key] settings as below.

WEP Key Status:
Displays WEP Key Status. Either "Not set" or "64 bit Key" or "128 bit Key" is displayed.

WEP (Encryption) Key and Confirm WEP (Encryption) Key:
To enter the WEP Key against for confirmation. WEP Key cannot be set without entering rhis item.

- 64 bit Key:
  To set 64 bit Key, entry of a 10-digit hexadecimal number required. Avoid starting each WEP Key with "0x". The number entered will be shown like "**********".
- 128 bit Key:
  To set 64 bit Key, entry of a 26-digit hexadecimal number required. The security is stronger than the security with 64 bit Key. Avoid starting each WEP Key with "0x". The number entered will be shown like "**********".

Program-10

Implement and simulate various types of routing algorithm.

a) Distance Vector routing protocol ROUTING

```
set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf

  set tr [open
  out.tr w]
  $ns
  trace-a
  $tr

proc finish {} {        global nf ns tr        $ns flush-trace
        close $tr
        exec nam out.nam &
          ex
  0

  }

set n0 [$ns node]
  set n1 [$n
  node]
  set n2 [$n
  node]
set n3 [$ns node]

  $ns duplex-link $n0 $n1 10Mb 10m
  DropTail
  $ns duplex-link $n1 $n3 10Mb 10m
  DropTail
$ns duplex-link $n2 $n1 10Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right-down
$ns duplex-link-op $n1 $n3 orient right

$ns duplex-link-op $n2 $n1 orient right-up

set tcp [new Agent/TCP] $ns attach-agent $n0 $tcp
```

```
set ftp [new Application/FTP]
$ftp attach-agent $tcp

set sink [new
Agent/TCPSink]
$ns attach-agen
$n3 $sink

set udp [new Agent/UDP] $ns attach-agent $n2 $udp

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

set null [new Agent/Null] $ns attach-agent $n3 $null

$ns connect $tc l
$sink
$ns connect $u$ns rtmodel-at 1.0 down $n1 $n3
$nul          $ns rtmodel-at 2.0 up $n1 $n3

$ns rtproto DV

$ns at 0.0 "$ft
start"        "
$ns at 0.0 "$c
start         $ns at 5.0 "finish"

$ns run
```

a) Link Sate Routing protocol

Step-1: Initializing the network :
The first step is to initialize the network simulator, and we do so by creating a network simulator object. After that, we initialize **rtproto** (routing protocol) to Link State (**LS**).

```
Set ns [new Simulator]

$ns rtproto LS
```

Step-2: Creating number of nodes :
We next create a random number of nodes, let's say 7. We use the **node** instance to create these nodes as follows.

```
set node1 [$ns node]
```

```
set node2 [$ns node]

set node3 [$ns node]

set node4 [$ns node]

set node5 [$ns node]

set node6 [$ns node]

set node7 [$ns node]
```

Step-3: Creating the trace file :
Our next step is to create the trace file and nam file. The nam file is used to view simulator output whereas the trace file traces all the routing information in the process. For this we create trace file and nam file objects and then open the files in write mode. The **trace-all** instance is used to trace all routing information into the trace file and similarly **namtrace-all** for the nam file.

```
set tf [open out.tr w]

$ns trace-all $tf

set nf [open out.nam w]

$ns namtrace-all $nf
```

Step-4: Labeling the nodes :
In the next step we can label the nodes if we wish to. Here we are labeling them from node 0 to node 6. We can also customize the labels by assigning different colors to them and thus viewing the simulation much more clearly. Here we will use red and blue.

```
$node1 label "node 1"

$node1 label "node 2"

$node1 label "node 3"

$node1 label "node 4"

$node1 label "node 5"

$node1 label "node 6"

$node1 label "node 7"

$node1 label-color blue

$node2 label-color red

$node3 label-color red

$node4 label-color blue

$node5 label-color blue

$node6 label-color blue

$node7 label-color blue
```

Step-5:Creating duplex links :
The next step is to create duplex links between the nodes forming a ring in the end. This can be achieved by using the duplex-link instance along with specifying three parameters: data rate (1.5Mb), delay (10ms) and kind of queue (**DropTail**).

```
$ns duplex-link $node1 $node2 1.5Mb 10ms DropTail

$ns duplex-link $node2 $node3 1.5Mb 10ms DropTail

$ns duplex-link $node3 $node4 1.5Mb 10ms DropTail

$ns duplex-link $node4 $node5 1.5Mb 10ms DropTail

$ns duplex-link $node5 $node6 1.5Mb 10ms DropTail

$ns duplex-link $node6 $node7 1.5Mb 10ms DropTail

$ns duplex-link $node7 $node1 1.5Mb 10ms DropTail
```

Step-6: Orient the links between the nodes :
Now we need to orient the links between the nodes appropriately to obtain proper alignment. The **duplex-link-op** instance is used for the same.

```
$ns duplex-link-op $node1 $node2 orient left-down

$ns duplex-link-op $node2 $node3 orient left-down

$ns duplex-link-op $node3 $node4 orient right-down

$ns duplex-link-op $node4 $node5 orient right

$ns duplex-link-op $node5 $node6 orient right-up

$ns duplex-link-op $node6 $node7 orient left-up

$ns duplex-link-op $node7 $node1 orient left-up
```

Step-7: Attaching TCP agents :
The next step is to attach TCP agents (using attach-agent) at two nodes let's say node 1 and node 4. We can do this creating the source and sink objects and connecting them using connect instance.

```
set tcp2 [new Agent/TCP]

$ns attach-agent $node1 $tcp2

set sink2 [new Agent/TCPSink]

$ns attach-agent $node4 $sink2

$ns connect $tcp2 $sink2
```

Step-8: Creating FTP traffic :
Our next step is to create FTP traffic and attach to TCP source. The traffic then flows across node 1 and node 4. We can do this by creating an FTP agent and attaching it to tcp2.

```
set traffic_ftp2 [new Application/FTP]

$traffic_ftp2 attach-agent $tcp2
```

Step-9: Adding a finish procedure :
The next step is to add a finish procedure to flush all data into trace file and then and then run the nam file.

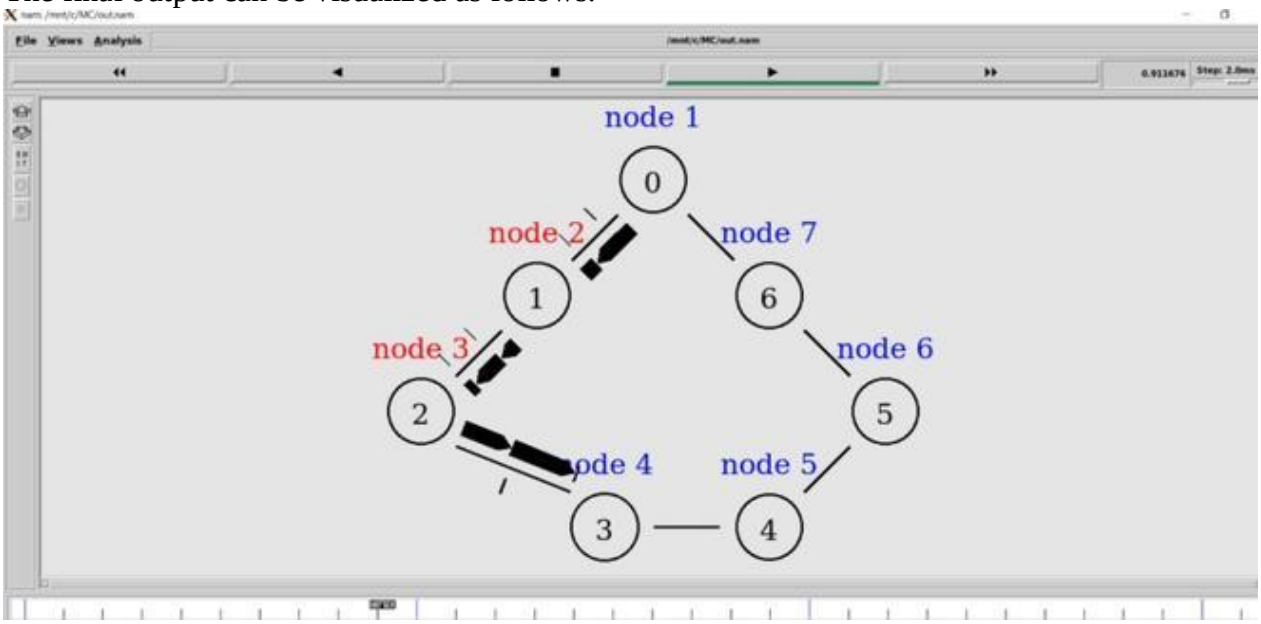
```
proc finish{} {

global ns nf

$ns flush-trace

close $nf

exec nam out.nam &

exit 0

}
```

Step-10: Scheduling the FTP :
The final step is to schedule the FTP traffic at the required time intervals. We can also disable the link between any pair of nodes at a certain timestamp using **rtmodel-at** instance and then enable it after a certain time. This is majorly done for testing purposes. Here we have disabled the link between nodes 2 and 3. The program ends with the run command.

```
$ns at 0.5 "traffic_ftp2 start"

$ns rtmodel-at 1.0 down $node2 $node3

$ns rtmodel-at 2.0 up $node2 $node3

$ns at 3.0 "traffic_ftp2 start"

$ns at 4.0 "traffic_ftp2 stop"

$ns at 5.0 "finish"

$ns run
```

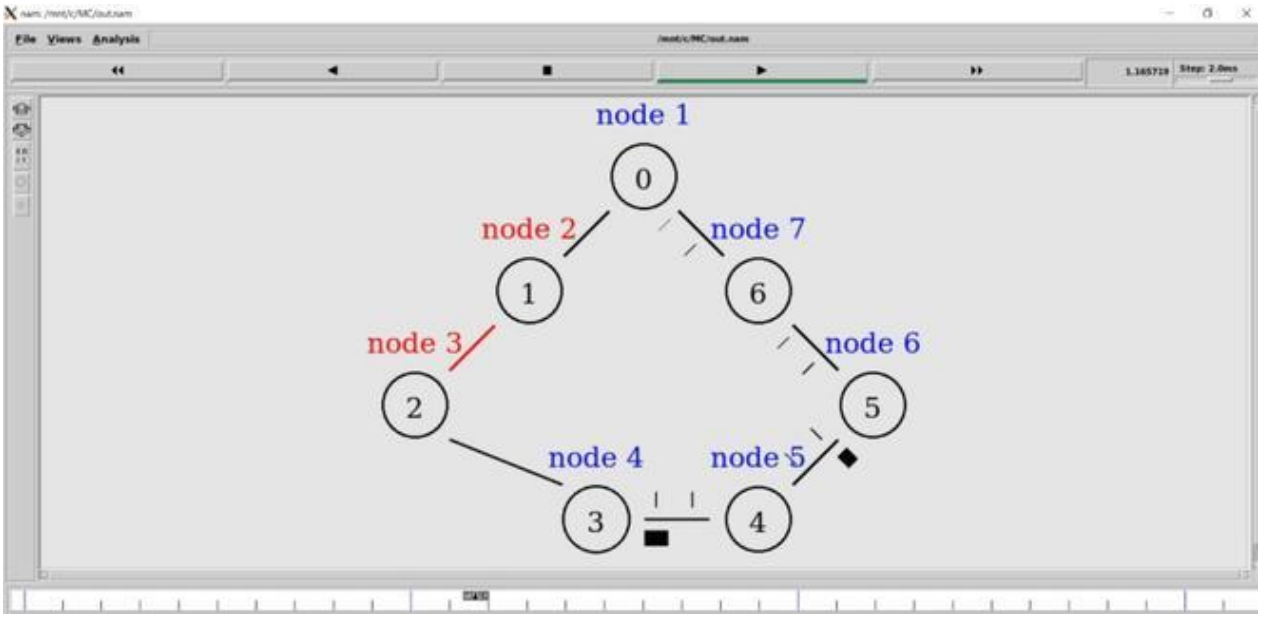
Output :

The final output can be visualized as follows.



Simulation before disabling the link

As shown in the figure above there is a normal flow of packets from node 1 to node 4. We now disable the link between nodes 2 and 3 at time 1 and the simulations changes as shown below. The disabled link is shown in red and the packet flow this time changes direction rather than the original route to node 4.



Program-11

Study and Simulation of MAC protocols like ALOHA, CSMA, CSMA/CD and CSMA/CA using standard network simulators

Static Channel Allocation uses Frequency Division Modulation (FDM) technique in which channel is divided in several portions and each user is assigned one to use for communication. This scheme works fine with limited number of users, as each one of the user has its private frequency to transmit and receive. In case of large number of users many of them will be denied permission due to lack of available bandwidth. Other approach is Dynamic Channel Allocation technique. This technique suggests use of single channel for every user by blocking others' working while some station is transmitting or receiving. Like token ring topology, to pass a special token for using the shared channel. This do not holds frue in case of wireless communication because users are mobile which can create conflicts between users for possession of token. Many algorithms are developed to allocate a multiple access to single channel with contention and collision models.

ALOHA:
A professor at University of Hawaii, Norman Abramson developed this protocol in 1970. ALOHA method was a pioneering computer networking system. ALOHA works fine in any wireless system, but as the communication gets heavier collision problem becomes worse. There are two versions of ALOHA: Pure and Slotted.

- **Pure ALOHA:**

This is the basic idea of the ALOHA, it allows any user to transmit whenever the data is ready to send. There will be collision if more than one station has transmitted at the same time, and frame or data sent gets grabbled. The user listens to the channel for any collisions, if there is any collision then transmitter station waits for random interval of time and resends the frame.
If the population is generating frames according to a Poisson distribution with mean 'N' and 'G' is the probability of 'k' transmission attempts per time frame, then throughput of the network can be given by.
$S=Ge^{2G}$

- **Slotted ALOHA:**

Roberts published a method of increasing the performance of Pure ALHOA in 1972. He proposed to divide the continuous time in Pure ALOHA in small discrete intervals and so it is called as Slotted ALOHA. This scheme practically doubled the throughput performance of the ALOHA. User is not permitted to send frames whenever ready but the user has to till next time slot.The network throughput is:$S = G\,e^{-G}$

Carrier Sense Multiple Access Protocols (CSMA):

Slotted ALOHA has increased network performance with an extent, but as stations keeps on sending frames more frames have to suffer collisions. Solution to. this problem is to sense the channel before sending frame and act accordingly.
Persistent and non-persistent CSMA: There are several different protocols which sense the carrier before transmitting frames. I-persistent CSMA, whenever user has data to send it first senses medium. If the carrier is idle then it transmits the frame otherwise waits for random interval of time (Tanenbaum, 2003). Station transmits data with probability of 1 whenever it finds channel idle so called as I-persistent CSMA. P-persistent CSMA: In this scheme whenever the station finds channel idle it transmits the frame with probability 'P' or differs transmission by probability 'q', where q = 1 — p (Tanenbaum, 2003). This process is repeated until successful transmission or some-other station starts transmitting.
Non-persistent CSMA: Working in this protocol is same as I-persistent protocol, but it does not keep on sensing the channel continuously instead it waits for random interval of time and repeats the algorithm (Tanenbaum, 2003).

CSMA with Collision Detection (CSMA-CD):

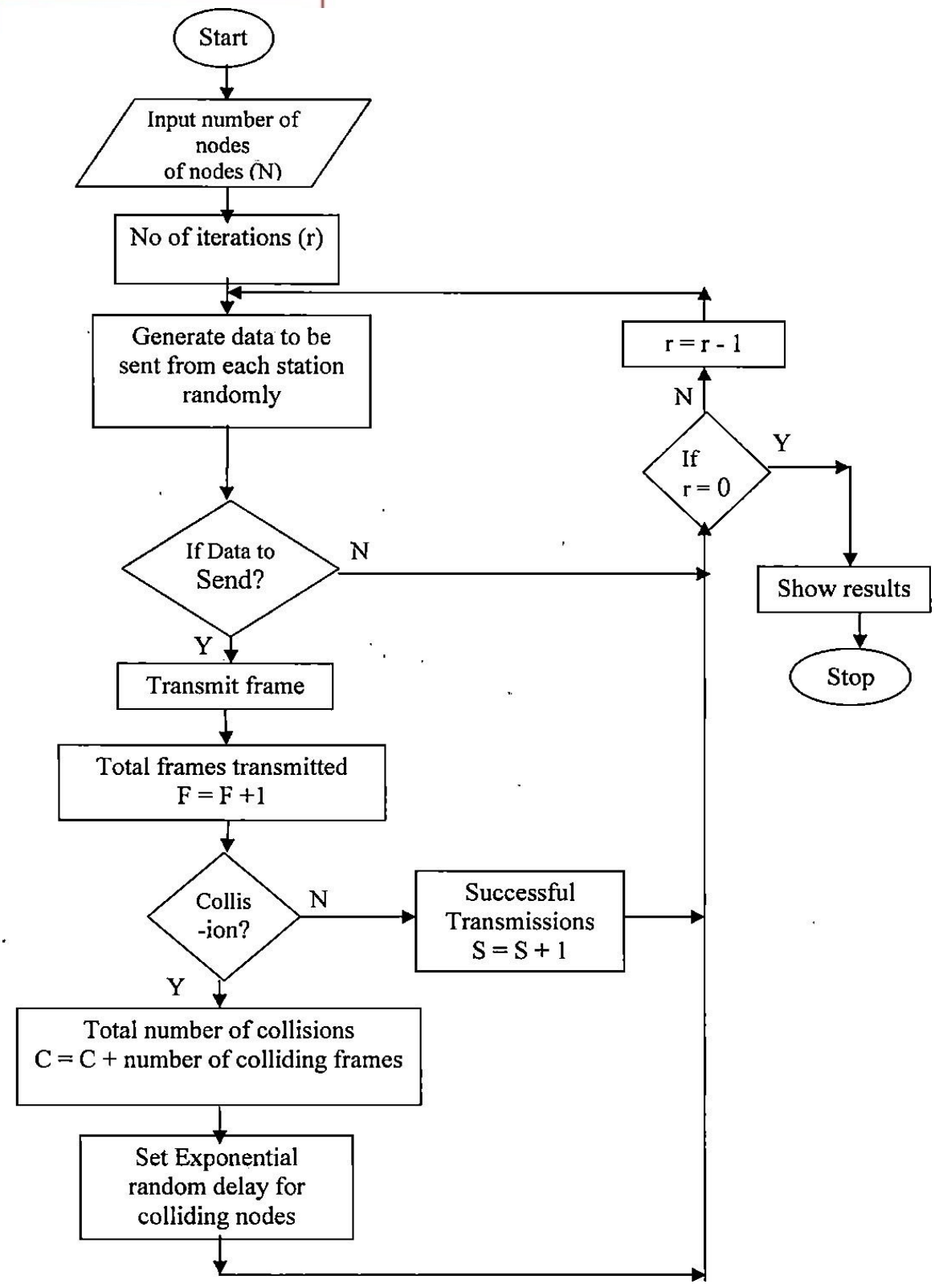
CSMA-CD is widely used in Local Area Networks (LAN) i.e. Ethernet. In this scheme if a station senses the channel to idle it starts transmitting, but they stops transmitting the frame if the detect any collision almost immediately. Then it waits for random interval of time and senses the channel again and repeats the algorithm (Tanenbaum, 2003). This quick termination of transmission saves time and bandwidth of the overall network. This protocol is inherently a half-duplex system cause each station continuously scans the channel for collisions which keeps the receiving logic busy for whole entire time of transmission preventing from receiving frames at the same time of transmitting.
Simulation Program now for ALOHA protocol:

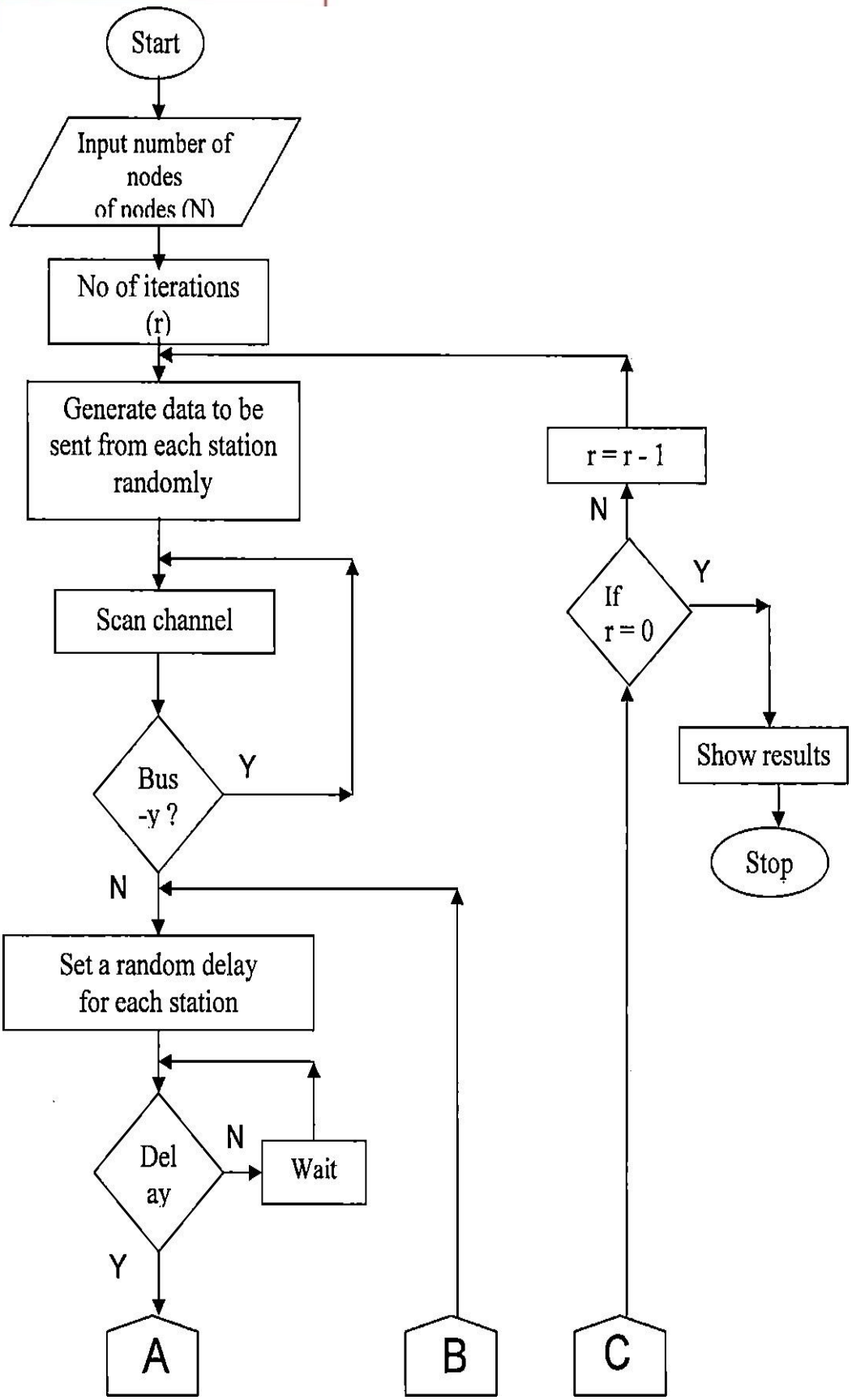
1. Input number of nodes in the network
2. Generate random number of data frames to transmit from each node
3. If data available for transmission and backoff delay expires (if any) transmit
4. Count total number of frames Transmitted
5. Scan the channel for Collision
6. If collision occurs, set random interval of delay for each colliding node
7. If collision does not occur, go back to Step 2
8. Show outputs.

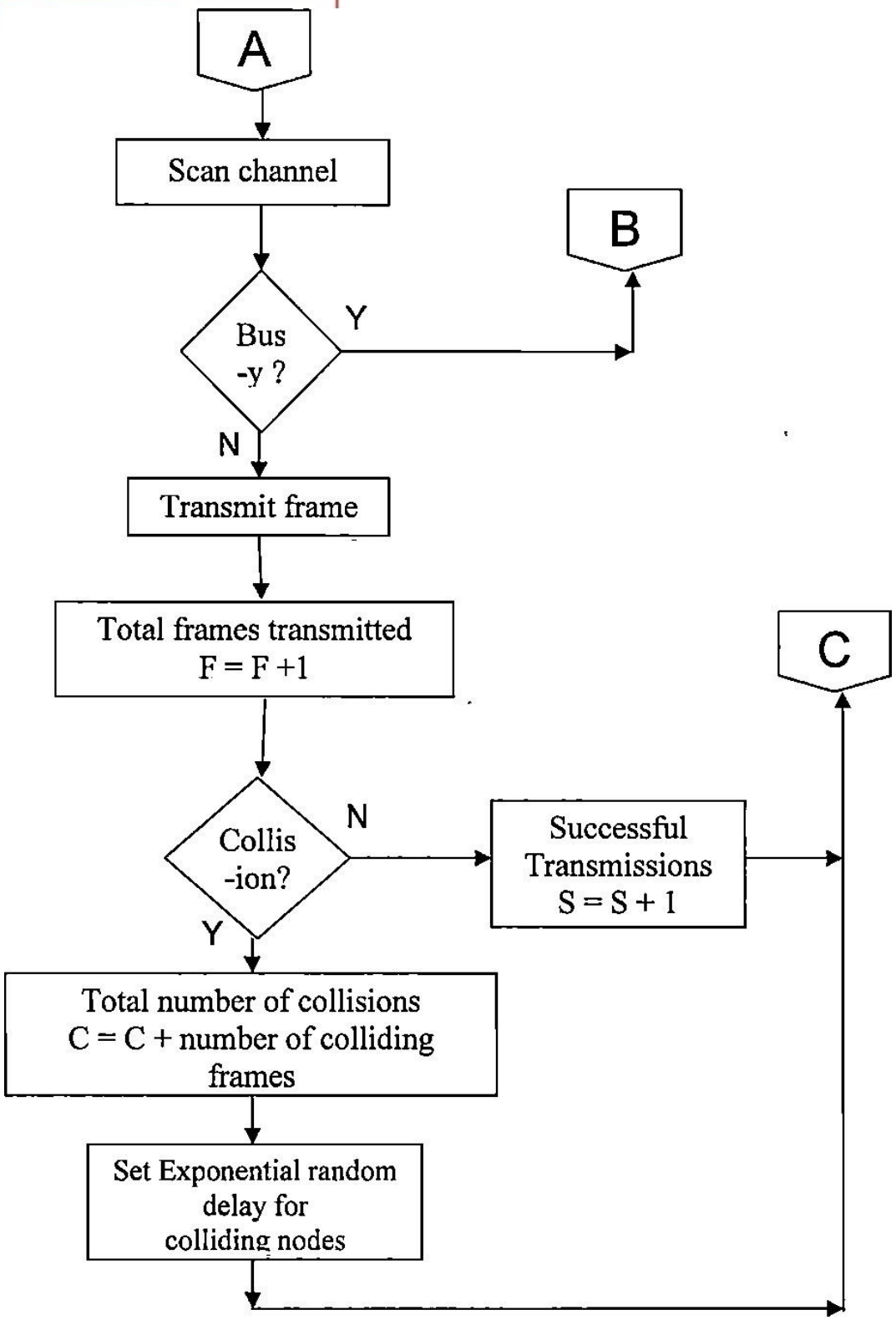
Simulation Program flow for CSMA/CA protocol:

1. Input number of nodes in the network
2. Generate random number of data frames to transmit from each node
3. If data available for transmission, scan the channel
4. If channel idle and backoff delay expires, transmit; else wait for longer interval of time, and transmit whenever channel is idle and delay expires

5. Count total number of frames Transmitted
6. Scan the channel for Collision
7. If collision occurs, set Exponential random interval of delay for each colliding node
8. If collision does not occur, go back to Step 2
9. Show outputs.


Start
Input number of nodes of nodes (N)
No of iterations (r)
Generate data to be sent from each station randomly
If Data to Send?
N
Y
Transmit frame
Total frames transmitted F = F +1
Collis -ion?
N
Successful Transmissions S = S + 1
Y
Total number of collisions C = C + number of colliding frames
Set Exponential random delay for colliding nodes
r = r - 1
N
If r = 0
Y
Show results
Stop


Start
Input number of nodes of nodes (N)
No of iterations (r)
Generate data to be sent from each station randomly
Scan channel
Bus -y ?
Y
N
Set a random delay for each station
Del ay
N
Wait
Y
A
B
C
r = r - 1
N
If r = 0
Y
Show results
Stop


A
Scan channel
B
Bus
-y ?
Y
N
Transmit frame
Total frames transmitted
F = F +1
C
Collis
-ion?
N
Y
Successful
Transmissions
S = S + 1
Total number of collisions
C = C + number of colliding
frames
Set Exponential random
delay for
colliding nodes

Program-12

Study of Application layer protocols-DNS, HTTP, HTTPS, FTP and TelNet

Domain Name System (DNS):

- To identify an entity, TCP/IP protocol uses the IP address which uniquely identifies the connection of a host to the Internet.
- DNS is a hierarchical system, based on a distributed database, that uses a hierarchy of Name Servers to resolve Internet host names into the corresponding IP addresses required for packet routing by issuing a DNS query to a name server.
- However, people refer to use names instead of address. Therefore, we need a system that can map a name to an address and conversely an address to name.
- In TCP/IP, this is the domain name system.
- DNS in the Internet: DNS is protocol that can be used in different platforms.

Hypertext Transfer Protocol (HTTP):

- This is a protocol used mainly to access data on the World Wide Web (www).
- The Hypertext Transfer Protocol (HTTP) the Web's main application-layer protocol although current browsers can access other types of servers.
- A repository of information spread all over the world and linked together.
- The HTIP protocol transfer data in the form of plain text, hyper text, audio, and video and so on.
- HTTP utilizes TCP connections to send client requests and server replies.

Hypertext Transfer Protocol Secure (HTTPS):

- Hypertext transfer protocol secure (HTTPS) is the secure version of HTTP.
- It is the primary protocol used to send data between a web browser and a website.
- HTTPS is encrypted in order to increase security of data transfer.
- This is particularly important when users transmit sensitive data, such as by logging into a bank account, email service, or health insurance provider.

File transfer Protocol (FTP):

- FTP is the standard mechanism provided by TCP/IP for copying a file from one host to another.
- FTP differs from other client-server applications because it establishes 2 connections between hosts. Two connections are: Data Connection and Control Connection.

- Data Connection uses PORT 20 for the purpose and control connection uses PORT 21 for the purpose.
- FTP is built on a client-server architecture and uses separate control and data connections between the client and the server.
- One connection is used for data transfer, the other for control information (commands and responses)

Telnet:

- TELNET is client-server application that allows a user to log onto remote machine and lets the user to access any application program on a remote computer.
- TELNET uses the NVT (Network Virtual Terminal) system to encode characters on the local system.
- On the server (remote) machine, NVT decodes the characters to a form acceptable to the remote machine.
- TELNET is a protocol that provides a general, bi-directional, eight-bit byte oriented communications facility.
- Many application protocols are built upon the TELNET protocol.