Gopalan College
of Engineering and Management
ISO 9001 : 2008

APPROVED BY AICTE NEW DELHI, AFFILIATED TO VTU BELGAUM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

NETWORKS LAB MANUAL-10CSL77

2016-2017

SEMESTER-VII

Prepared by:	Reviewed by:	Approved by:
Aparna N,	**N S Saradha Devi**	**Dr. A A Powly Thomas**
Asst. Professor	**Head of the Department**	**Principal**
Dept. of CSE,	**Dept. of CSE**	**GCEM**
GCEM	**GCEM**	

181/1, 182/1, Hoodi Village, Sonnenahalli, K.R. Puram, Bengaluru, Karnataka 560048

TABLE OF CONTENTS

Networks Laboratory
Subject Code: 10CSL77 I.A. Marks: 25
Hours/Week: 03 Exam Hours: 03
Total Hours: 42 Exam Marks: 50

Note : Student is required to solve one problem from PART-A and one problem from PART-B. The questions are allotted based on lots. Both Questions carry equal marks. .

PART A – Simulation Exercises

The following experiments shall be conducted using either NS228/OPNET or any other simulators.

1. Simulate a three nodes point-to-point network with duplex links between them. Set the queue size vary the bandwidth and find the number of packets dropped.
2. Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets by TCP/UDP.
3. Simulate the transmission of ping messaged over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
4. Simulate an Ethernet LAN using N-nodes(6-10), change error rate and data rate and compare the throughput.
5. Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and plot congestion window for different source/destination.
6. Simulate simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

PART B

The following experiments shall be conducted using C/C++.

7. Write a program for error detecting code using CRC-CCITT (16-bits).
8. Write a program for distance vector algorithm to find suitable path for transmission.
9. Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.
10. Implement the above program using as message queues or FIFOs as IPC channels.
11. Write a program for simple RSA algorithm to encrypt and decrypt the data.
12. Write a program for congestion control using Leaky bucket algorithm.

Note:
In the examination, a combination of one problem has to be asked from Part A for a total of 25 marks and one problem from Part B has to be asked for a total of 25 marks. The choice must be based on random selection from the entire lots.

COURSE OBJECTIVE

Upon successful completion of this Lab the student will be able to:

- Understand the simulation using NCTU/NS.
- Simulate a three nodes point – to – point network with duplex links between them.
- Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
- Simulate an Ethernet LAN using n nodes.
- Understand the error detecting using cyclic redundancy check (**CRC).**
- Understand how the distance vector algorithm can be used to find the shortest path.
- The client-Server communication by message queue or FIFO.
- Control congestion using Leaky bucket algorithm.

COURSE OUTCOME

- Acquire knowledge of using simulators for different connections.
- Able to do error detection using CRC.
- Able to find the shortest path in the network using distance vector algorithm.
- Able to do interprocess communication and encryption and decryption of data will be clear.

COMPUTER LAB DO'S AND DON'T

DO'S

1. Know the location of the fire extinguisher and the first aid box and how to use them in case of an emergency.
2. Read and understand how to carry out an activity thoroughly before coming to the laboratory.
3. Report fires or accidents to your lecturer/laboratory technician immediately.
4. Report any broken plugs or exposed electrical wires to your lecturer/laboratory technician immediately.

DON'TS

1. Do not eat or drink in the laboratory.
2. Avoid stepping on electrical wires or any other computer cables.
3. Do not open the system unit casing or monitor casing particularly when the power is turned on. Some internal components hold electric voltages of up to 30000 volts, which can be fatal.
4. Do not insert metal objects such as clips, pins and needles into the computer casings. They may cause fire.
5. Do not remove anything from the computer laboratory without permission.
6. Do not touch, connect or disconnect any plug or cable without your lecturer/laboratory technician's permission.
7. Do not misbehave in the computer laboratory.

LIST OF EXPERIMENTS

PART – A

SIMULATION-INTRODUCTION

Network simulation is an important tool in developing, testing and evaluating network protocols. Simulation can be used without the target physical hardware, making it economical and practical for almost any scale of network topology and setup. It is possible to simulate a link of any bandwidth and delay, even if such a link is currently impossible in the real world. With simulation, it is possible to set each simulated node to use any desired software. This means that meaning deploying software is not an issue. Results are also easier to obtain and analyze, because extracting information from important points in the simulated network is as done by simply parsing the generated trace files.

Simulation is only of use if the results are accurate, an inaccurate simulator is not useful at all. Most network simulators use abstractions of network protocols, rather than the real thing, making their results less convincing. S.Y. Wang reports that the simulator OPNET uses a simplified finite state machine to model complex TCP protocol processing. [19] NS-2 uses a model based on BSD TCP, it is implemented as a set of classes using inheritance. Neither uses protocol code that is used in real world networking.

Setting up the environment

A user using the NCTUns in single machine mode, needs to do the following steps before he/she starts the GUI program:

1. Set up environment variables:
   Before the user can run up the dispatcher, coordinator, or NCTUns GUI

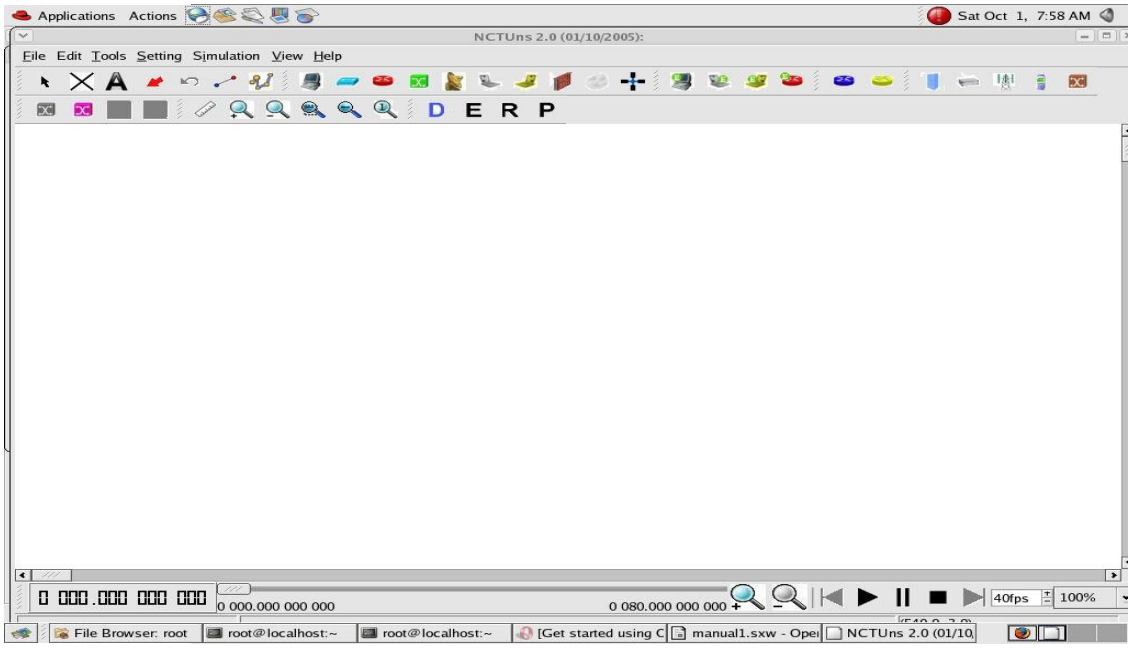
   program he/she must set up the NCTUNSHOME environment variable.

2. Start up the dispatcher on terminal 1.

3. Start up the coordinator on terminal 2.

4. Start up the nctuns client on terminal 3.

After the above steps are followed, the starting screen of NCTUns disappears and the user is presented with the working window as shown below:



Drawing A Network Topology

To draw a new network topology, a user can perform the following steps:

Choose Menu->File->Operating Mode-> and make sure that the “Draw Topology” mode is checked. This is the default mode of NCTUns when it is launched. It is only in this mode that a user can draw a new network topology or change an existing simulation topology. When a user switches the mode to the next mode “Edit Property”, the simulation network topology can no longer be changed.

1. Move the cursor to the toolbar.
2. Left-Click the router icon on the toolbar.
3. Left-Click anywhere in the blank working area to add a router to the current network topology. In the same way we can add switch, hub,WLAN access point,WLAN mobile node , wall (wireless signal obstacle) etc.
4. Left-Click the host icon on the toolbar. Like in step 4, add the required number of hosts to the current topology.

5. To add links between the hosts and the router, left-click the link icon on the toolbar to select it.

6. Left-Click a host and hold the mouse button. Drag this link to the router and then release the mouse left button on top of the router. Now a link between the selected host and the router has been created.

7. Add the other, required number of links in the same way. This completes the creation of a simple network topology.

8. Save this network topology by choosing Menu->File->Save. It is saved with a .tpl extension.

9. Take the snapshot of the above topology.

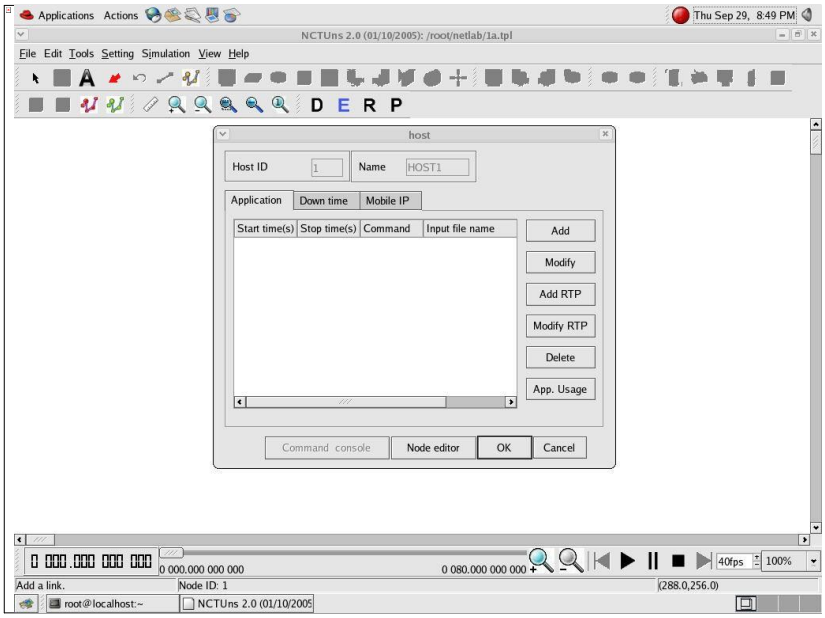
Editing Node's Properties

1. A network node (device) may have many parameters to set. For example, we may have to set the maximum bandwidth, maximum queue size etc to be used in a network interface. For another example, we may want to specify that some application programs (traffic generators) should be run on some hosts or routers to generate network traffic.

2. Before a user can start editing the properties of a node, he/she should switch the mode from the "Draw Topology" to "Edit Property" mode. In this mode, topology changes can no longer be made. That is, a user cannot add or delete nodes or links at this time.

3. The GUI automatically finds subnets in a network and generates and assigns IP and MAC addresses to layer 3 network interfaces.

4. A user should be aware that if he/she switches the mode back to the "Draw Topology" mode when he/she again switches the mode back to the "Edit Topology" mode, node's IP and MAC addresses will be regenerated and assigned to layer 3 interfaces.



Therefore the application programs now may use wrong IP addresses to communicate with their partners.

Running the Simulation

When a user finishes editing the properties of network nodes and specifying application programs to be executed during a simulation, he/she can start running the simulation.

1. In order to do so, the user must switch mode explicitly from "Edit Property" to "Run Simulation". Entering this mode indicates that no more changes can (should) be made to the simulation case, which is reasonable. This simulation is about to be started at this moment; of course, any of its settings should be fixed.

2. Whenever the mode is switched to the " Run Simulation" mode, the many simulation files that collectively describe the simulation case will be exported. These simulation files will be transferred to the (either remote or local) simulation server for it to execute the simulation. These files are stored in the " main File Name.sim" directory, where main Filename is the name of the simulation case chosen in the "Draw Topology" mode.

Playing Back the Packet Animation Trace

After the simulation is finished, the simulation server will send back the simulation result files to the GUI program after receiving these files, the GUI program will store these files in the "results directory" .It will then automatically switch to "play back mode".

1. These files include a packet animation trace file and all performance log files that the user specifies to generate. Outputting these performance log files can be specified by checking some output options in some protocol modules in the node editor. In addition to this, application programs can generate their own data files.

2. The packet animation trace file can be replayed later by the packet animation player. The performance curve of these log files can be plotted by the performance monitor.

Post Analysis

1. When the user wants to review the simulation results of a simulation case that has been finished before, he /she can run up the GUI program again and then open the case's topology file.

2. The user can switch the mode directly to the "Play Back" mode. The GUI program will then automatically reload the results (including the packet animation trace file and performance log file.

3. After the loading process is finished, the user can use the control buttons located at the bottom of the screen to view the animation.

Simulation Commands

- **Run**: Start to run simulation.
- **Pause**: Pause the currently -running simulation.
- **Continue**: Continue the simulation that was just paused.
- **Stop**: Stop the currently -running simulation
- **Abort**: Abort the currently running simulation. The difference between "stop" and "abort" is that a stopped simulation job's partial results will be

transferred back to GUI files.

- **Reconnect**: The Reconnect command can be executed to reconnect to a simulation job that was previously disconnected. All disconnected jobs that have not finished their simulations or have finished their simulations but the results have not been retrieved back to be a GUI program by the user will appear in a session table next to the "Reconnect" command. When executing the reconnect command, a user can choose a disconnected job to reconnect from this session table.
- **Disconnect**: Disconnect the GUI from the currently running simulation job. The GUI now can be used to service another simulation job. A disconnected simulation will be given a session name and stored in a session table.

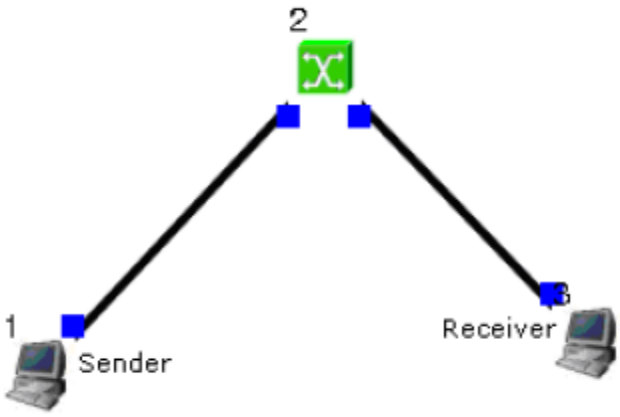
PART A Programs

Exp.No:1

Simulate to Find the Number of Packets Dropped

AIM:

Simulate a three-node point-to-point network with a duplex link between them. Set the queue size and vary the bandwidth and find the number of packets dropped.



Sender:-

stcp –p 2000 –l 1024 1.0.1.2

Receiver:-

rtcp –p 2000 –l 1024

Parameters:-

Drop Packets and Collision Packets.

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place a HOST1 on the editor.
   Repeat the above procedure and place another host "HOST2" on the editor.

2. Select/click the HUB icon on the toolbar and click the left mouse button on the editor, to place HUB1 on the editor.

3. Click on the LINK icon on the toolbar and connect HOST1 to HUB1 and HUB1 to HOST2

4. Click on the "E" icon on the toolbar to save the current topology
   e.g: file1.tpl
   (Look for the *******.tpl extension.)

NOTE: Changes cannot / (should not) be done after selecting the "E" icon.

Step2: Configuration

1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

   stg –u 1024 100 1.0.1.2

3. Click OK button on the command window to exit and once again click on the OK button on the HOST window to exit.

4. Double click the left mouse button while cursor is on HOST2 to open the HOST window.

5. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

   rtg –u –w log1

6. Click OK button on the command window to exit.

7. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

8. Select LOG STATISTICS and select checkboxes for Number of Drop Packet and Number of Collisions in the MAC window

9. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

Note: To set QUEUE size

1. Double click the left mouse button while cursor is on HOST2 to open the HOST window.

2. Click NODE EDITOR Button on the HOST window and select the FIFO tab from the modal window that pops up.

3. Change Queue size (Default 50).

4. Click OK button on the FIFO window to exit and once again click on the OK button on the HOST window to exit.

Step3: Simulate

i. Click "R" icon on the tool bar

ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.

iii. venTo start playback select "►" icon located at the bottom right corner of the editor.

iv. To view results, Open up new TERMINAL window, move to file1.results folder and open collision and drop log files in separate TERMINAL window.

Caution: file1 is the hypothetical name given to this simulation.

(Refer Step 1.4)

Changing configurations

Change 1

1. Open the above file,
2. Do not change the topology or any other configuration,
3. Select E icon on the toolbar
4. Reduce the bandwidth at link2 by double clicking the left mouse button while cursor is on link2 .(Change bandwidth on both tabs Uplink/Downlink)
5. Repeat Step3 (Simulate)

Change 2

1. Open the above file,
2. Remove HUB and replace it with SWITCH.
3. Do not change anything in the configuration

Repeat Step3(Simulate)

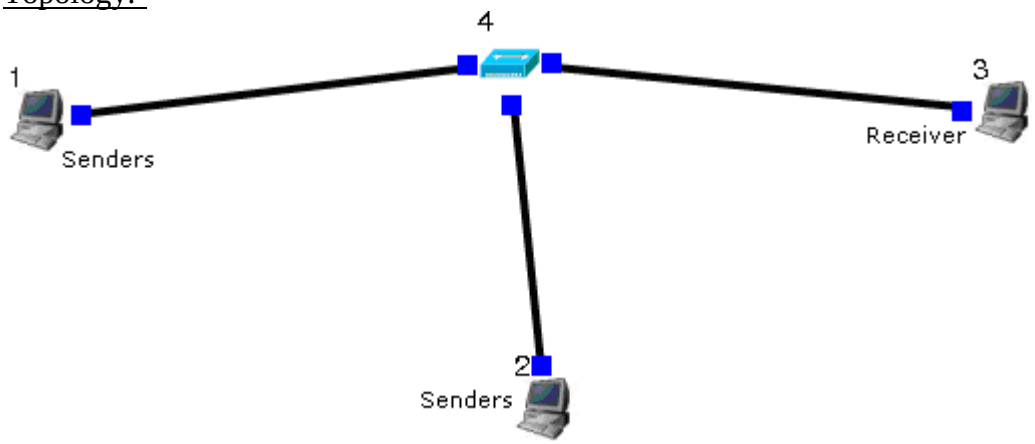
Exp.No:2

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

AIM:

Simulate a four-node point-to-point network and connect the link as follows: Apply a TCP agent between n0 to n3 and apply a UDP agent between n1 and n3. Apply relevant applications over TCP and UDP agents changing the parameters and determine the number of packets sent by two agents.

Topology:-



Sender:-

stcp –p 3000 –l 1024 1.0.1.3
stg –u 1024 1.0.1.3

Receiver:-

rtcp –p 3000 –l 1024
rtg –u 3000

Parameters:-

Throughput of incoming and outgoing Packets

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place a host on the editor.
   Repeat the above procedure and place two other hosts “HOST2” and “HOST3” on the editor.

2. Select/click the HUB (or SWITCH) icon on the toolbar and click the left mouse button on the editor, to place a HUB (or SWITCH) on the editor.

3. Click on the LINK icon on the toolbar and connect HOST1 to HUB, HOST2 to HUB and HUB to HOST3

4. Click on the "E" icon on the toolbar to save the current topology **e.g:** file2.tpl
   (Look for the ******.tpl extension.)

NOTE: Changes cannot / (should not) be done after selecting the "E" icon.

Step2: Configuration

1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

   stcp –p 21 –l 1024 1.0.1.3

3. Click OK button on the command window to exit

4. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

5. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

6. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

7. Double click the left mouse button while cursor is on HOST2 to open the HOST window.

8. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

   stg –u 1024 100 1.0.1.3

9. Click OK button on the command window to exit

10. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

11. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

12. Click OK button on the MAC window to exit and once again click on the OK button

on the HOST window to exit.

13. Double click the left mouse button while cursor is on HOST3 to open the HOST window.

14. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.
    rtcp –p 21 –l 1024

15. Click OK button on the command window to exit.

16. Also add the following command on HOST3
    rtg –u –w log1

17. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

18. Select LOG STATISTICS and select checkbox for input and output throughput in the MAC window

19. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

Step3: Simulate

i. Click "R" icon on the tool bar

ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.

iii. To start playback select "►" icon located at the bottom right corner of the editor.

iv. To view results, Open up new TERMINAL window, move to file2.results folder and open input and output throughput log files in separate TERMINAL window.

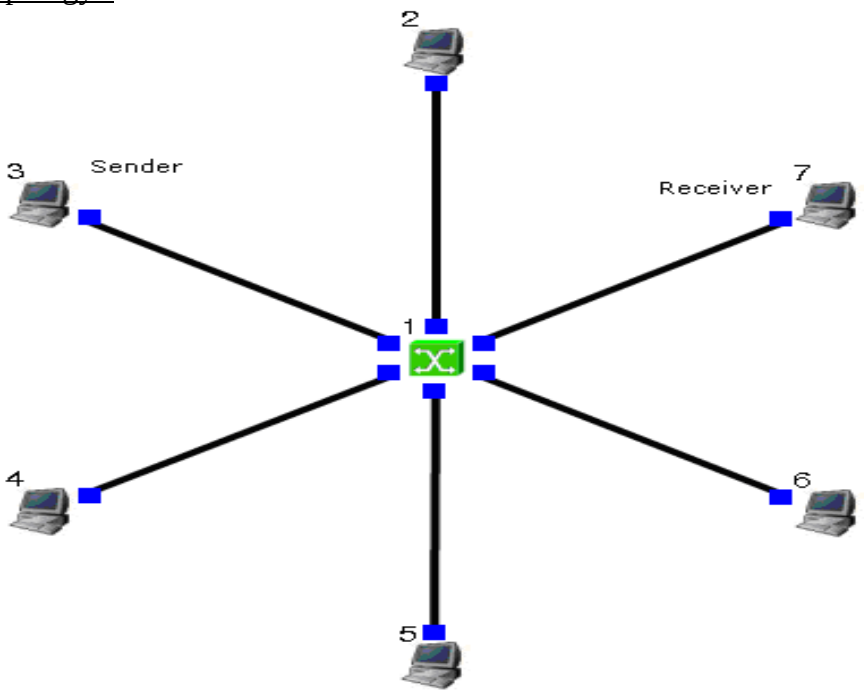
Caution: file2 is the hypothetical name given to this simulation.
(Refer Step 1.4)

Exp.No:3

Simulate to Find the Number of Packets Dropped due to Congestion

AIM:
Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

Topology:-



Sender:-

stcp –p 2000 –l 1024 1.0.1.4

Receiver:-

rtcp –p 2000 –l 1024

Command Console:-

Goto tools-> simulation time and change Simulation time to 100. During run mode, double click host 2 and then click command console. And execute the following command.

ping 1.0.1.4

Parameters:-

Drop Packets and Collision Packets.

Step1: Drawing topology

1. Select/click the SUBNET icon on the toolbar and click the left mouse button on the editor, to place a SUBNET on the editor.

2. A pop up window appears requesting the number of nodes and radius for the subnet
   Set number of nodes=6;
   Set radius of subnet >150

3. Click on the "E" icon on the toolbar to save the current topology **e.g:** file4.tpl
   (Look for the ******.tpl extension.)

NOTE: Changes cannot / (should not) be done after selecting

Step2: Configuration

4. Double click the left mouse button while cursor is on a HOST to open the HOST window.

5. Click NODE EDITOR Button on the HOST window and select the INTERFACE tab (1st tab) from the modal window that pops up.

6. Determine the IP address of the selected host.

7. Click OK button on the INTERFACE window to exit and once again click on the OK button on the HOST window to exit.

8. Repeat the above step for 2 other HOSTS

9. Also click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

10. Select LOG STATISTICS and select checkbox for drop and collision log statistics in the MAC window

11. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

12. Repeat steps 6 to 9 for the other hosts selected at step 5.

13. Select G_Setting from the menu bar and select Simulation from the drop down list
    Set simulation time>600sec

Step3: Simulate

i. Click "R" icon on the tool bar
ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
iii. During simulation, open a new terminal window.

iv. Type **ping IP address** of a host in the subnet at the command prompt.

v. To view results, Open up new TERMINAL window, move to file4.results folder and open drop and collision log files in separate TERMINAL window.

Caution: file4 is the hypothetical name given to this simulation.

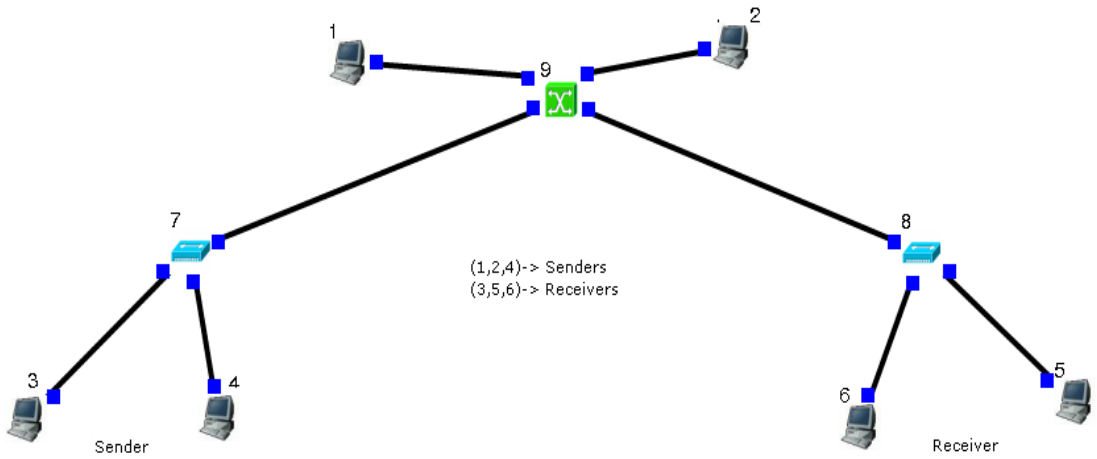
(Refer Step 1.3)

Exp.No:4

Simulate to Compare Data Rate & Throughput

AIM:
Simulate an Ethernet LAN using N nodes (6-10), change error rate and data rate and compare throughput.

Topology:-



Sender:-

stcp –p 2000 –l 1024 1.0.1.4

Receiver:-

rtcp –p 2000 –l 1024

Double click on receiver link and change BER to 0.000001, Run Again.

Parameters:-

Throughput of outgoing Packets

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place HOST1 on the editor.
   Repeat the above procedure and place 5 other hosts “HOST2”, “HOST3”, “HOST4”, “HOST5”, and “HOST6”on the editor.

2. Select/click the HUB icon on the toolbar and click the left mouse button on the editor, to place HUB1 on the editor.
   Repeat the above procedure and place another host “HUB2” on the editor

3. Click on the LINK icon on the toolbar and connect HOST1, HOST2 and HOST3 to

HUB1, HOST4, HOST5 and HOST6 to HUB2.

4. Select/click the SWITCH icon on the toolbar and click the left mouse button on the editor, to place SWITCH1 on the editor.

5. Click on the LINK icon on the toolbar and connect HUB1 to SWITCH1 and HUB2 to SWITCH1.

6. Click on the "E" icon on the toolbar to save the current topology **e.g:** file5.tpl
(Look for the ******.tpl extension.)

NOTE: Changes cannot / (should not) be done after selecting the "E" icon.

Step2: Configuration

1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

stcp –p 21 –l 1024 1.0.1.4

3. Click OK button on the command window to exit and once again click on the OK button on the HOST window to exit.

4. Repeat this step at HOST 2 and HOST3, but use different commands

stcp –p 21 –l 1024 1.0.1.5 at HOST2
stcp –p 21 –l 1024 1.0.1.6 at HOST3

5. Double click the left mouse button while cursor is on HOST4 to open the HOST window.

6. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

rtcp –p 21 –l 1024

7. Click OK button on the command window to exit.

8. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

9. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

10. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

11. Repeat this step at HOST 5 and HOST6, but use different commands
    rtcp –p 21 –l 1024 at HOST5
    rtcp –p 21 –l 1024 at HOST6

12. Double click the left mouse button while cursor is on HOST5 to open the HOST window.

13. Click NODE EDITOR Button on the HOST5 window and select the PHYSICAL tab from the modal window that pops up.

14. Change Bit Error Rate

15. Click OK button on the PHYSICAL window to exit and once again click on the OK button to return to the HOST window

16. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

17. Select LOG STATISTICS and select checkbox for output throughput in the MAC window

18. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

19. Repeat this step HOST6, Change Bandwidth this time while undoing the change in Bit Error Rate, also select the output throughput at HOST6.

Step3: Simulate

i. Click "R" icon on the tool bar

ii. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.

iii. To start playback select "►" icon located at the bottom right

iv. To view results, Open up new TERMINAL window, move to file5.results folder and open output throughput log files in separate TERMINAL window.

Caution: file5 is the hypothetical name we gave to this simulation
(Refer Step 1.7)

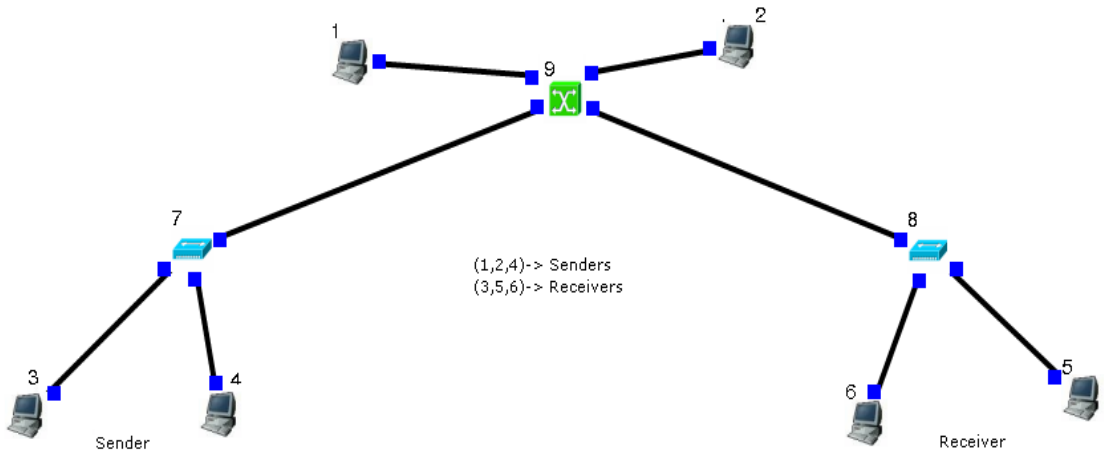
Exp.No:5

Simulate to Plot Congestion for Different Source/Destination

AIM:

Simulate an Ethernet LAN using N nodes and set multiple traffic nodes and plot congestion window for different source/destination.

Topology:-



Sender:-

stcp –p 2000 –l 1024 1.0.1.4

Receiver:-

rtcp –p 2000 –l 1024

Parameters:-

Receiver side Collision Packets and Drop Packets

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place HOST1 on the editor.
   Repeat the above procedure and place 3 other hosts “HOST2”, “HOST3”, “HOST4”, “HOST5”, and “HOST6”on the editor.

2. Select/click the HUB icon on the toolbar and click the left mouse button on the editor, to place HUB1 on the editor.
   Repeat the above procedure and place another host “HUB2” on the editor

3. Click on the LINK icon on the toolbar and connect HOST1, HOST2 and HOST3 to HUB1, HOST4, HOST5 and HOST6 to HUB2.

4. Select/click the SWITCH icon on the toolbar and click the left mouse button on the

editor, to place SWITCH1 the editor.

5. Click on the LINK icon on the toolbar and connect HUB1 to SWITCH1 and HUB2 to SWITCH1.

6. Click on the "E" icon on the toolbar to save the current topology **e.g:** file7.tpl
(Look for the ******.tpl extension.)
NOTE: Changes cannot / (should not) be done after selecting the "E" icon.

Step2: Configuration

1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

stcp –p 21 –l 1024 1.0.1.4

3. Click OK button on the command window to exit and once again click on the OK button on the HOST window to exit.

4. Repeat this step at HOST 2 and HOST3, but use different commands

stcp –p 23 –l 1024 1.0.1.5 at HOST2
stcp –p 25 –l 1024 1.0.1.6 at HOST3

5. Double click the left mouse button while cursor is on HOST4 to open the HOST window.

6. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

rtcp –p 21 –l 1024

7. Click OK button on the command window to exit.

8. Click NODE EDITOR Button on the HOST window and select the MAC tab from the modal window that pops up.

9. Select LOG STATISTICS and select checkbox for Number of drop and collisions packets in the MAC window

10. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

11. Repeat this step at HOST 5 and HOST6, but use different commands

rtcp –p 23 –l 1024 at HOST5
rtcp –p 25 –l 1024 at HOST6

12. Double click the left mouse button while cursor is on HOST5 to open the HOST window.

13. Click NODE EDITOR Button on the HOST5 window and select the MAC tab from the modal window that pops up.

14. Select LOG STATISTICS and select checkbox for Number of drop and collisions packets in the MAC window

15. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

16. Also select the drop and collisions at HOST6.

Step3: Simulate

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

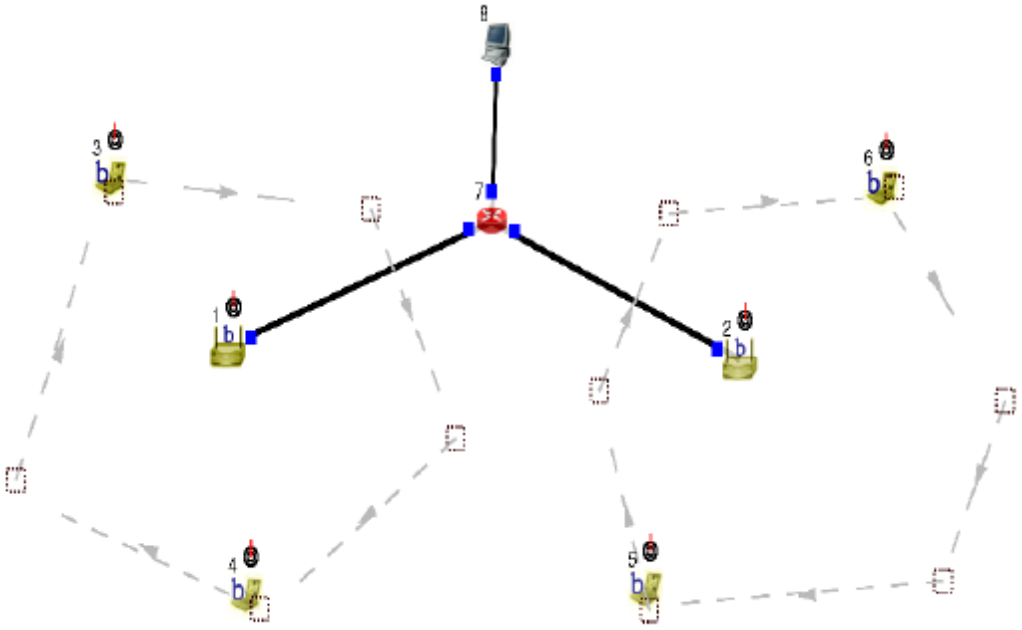
Caution: file7 is the hypothetical name given to this simulation.

Exp.No:6

Simulate to Determine the Performance with respect to Transmission of Packets

AIM:
Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

Topology:-



Click on “access point”. Goto wireless interface and tick on “show transmission range and then click OK.

Double click on Router -> Node Editor and then
Left stack -> throughput of Incoming packets
Right stack -> throughput of Outgoing packets

Select mobile hosts and access points then click on.
Tools -> WLAN mobile nodes-> WLAN Generate infrastructure.
Subnet ID: Port number of router (2)
Gateway ID: IP address of router

Mobile Host 1
ttcp –t –u –s –p 3000 1.0.1.1

Mobile Host 1
ttcp –t –u –s –p 3001 1.0.1.1

Host(Receiver)

ttcp –r –u –s –p 3000
ttcp –r –u –s –p 3001

Run and then play to plot the graph.

Step1: Drawing topology

1. Select/click the HOST icon on the toolbar and click the left mouse button on the editor, to place HOST1 on the editor.

2. Select/click the ROUTER icon on the toolbar and click the left mouse button on the editor, to place ROUTER1 on the editor.

3. Select/click the WIRELESS ACCESS POINT(802.11b) icon on the toolbar and click the left mouse button on the editor, to place ACCESS POINT 1 on the editor.
   Repeat this procedure and place ACCESS POINT 2 on the editor.

4. Select/click the MOBILE NODE (infrastructure mode) icon on the toolbar and click the left mouse button on the editor, to place MOBILE NODE 1 on the editor.
   Repeat this procedure and place MOBILE NODE 2, MOBILE NODE3 and MOBILE NODE 4 on the editor.

5. Click on the LINK icon on the toolbar and connect ACCESS POINT1 to ROUTER1 and ACCESS POINT2 to ROUTER1

6. Click on the "Create a moving path" icon on the toolbar and draw moving path across MOBILE NODE 1 and 2, Repeat for MOBILE NODE 3and 4 (Accept the default speed value 10 and close the window, Click the right mouse button to terminate the path).

To create Subnet

7. Select wireless subnet icon in the toolbar now select MOBILE NODE1, MOBILE NODE2 and ACCESS POINT1 by clicking on left mouse button, and clicking right mouse button will create a subnet.

8. Repeat the above step for MOBILE NODE3, MOBILE NODE4 and ACCESS POINT2.

9. Click on the "E" icon on the toolbar to save the current topology **e.g:** file8.tpl
   (Look for the ******.tpl extension.)
   NOTE: Changes cannot / (should not) be done after selecting the "E" icon.

Step2: Configuration

1. Double click the left mouse button while cursor is on HOST1 to open the HOST window.

2. Select Add button on the HOST window to invoke the command window and provide the following command in the command textbox.

   ttcp –r –u –s –p 8001

3. Click OK button on the command window to exit

4. Repeat this step and add the following commands at HOST1

   ttcp –r –u –s –p 8002
   ttcp –r –u –s –p 8003
   ttcp –r –u –s –p 8004

5. Click NODE EDITOR Button on the HOST1 window and select the MAC tab from the modal window that pops up.

6. Select LOG STATISTICS and select checkbox for Input throughput in the MAC window

7. Click OK button on the MAC window to exit and once again click on the OK button on the HOST window to exit.

8. Double click the left mouse button while cursor is on MOBILE NODE 1 to open the MOBILE NODE window.

9. Select Application tab and select Add button to invoke the command window and provide the following command in the command textbox.

   ttcp –t –u –s –p 80011.0.2.2 (host's ip address)

10. Click NODE EDITOR Button on the MOBILE NODE1 window and select the MAC tab from the nodal window that pops up.

11. Select LOG STATISTICS and select checkbox for Output throughput in the MAC window

12. Click OK button on the MAC window to exit and once again click on the OK button on the MOBILE NODE1 window to exit.

13. Repeat the above steps (step 8 to step12) for the MOBILE NODE2,3 and 4 and add the following commands at
    MOBILE NODE2:- ttcp –t –u –s –p 8002 1.0.2.2
    MOBILE NODE3:- ttcp –t –u –s –p 8003 1.0.2.2
    MOBILE NODE4:- ttcp –t –u –s –p 8004 1.0.2.2

14. Double click the left mouse button while cursor is on ROTER1 to open the ROUTER window.

15. Click NODE EDITOR Button on the ROUTER1 window and you can see three stacks. two stacks for two ACCESS POINTS and another stack for HOST1 which is connected to the ROUTER1.

16. Select the MAC tab of ACCESS POINT1 and Select LOG STATISTICS and select checkbox for Input throughput in the MAC window. Click OK button on the MAC window to exit.

17. Select the MAC tab of ACCESS POINT2 and Select LOG STATISTICS and select checkbox for Input throughput in the MAC window. Click OK button on the MAC window to exit.

18. Select the MAC tab of HOST1 and Select LOG STATISTICS and select checkbox for Output throughput in the MAC window. Click OK button on the MAC window to exit.

Step3: Simulate

i. Click "R" icon on the tool bar
ii. Select Simulation in the menu bar and click/select RUN in the dropdown list to execute the simulation.
iii. To start playback select "►" icon located at the bottom right corner of the editor.
iv. MOBILE NODE's start moving across the paths already drawn.

Caution: file8 is the hypothetical name given to this simulation,

Part B Programs

Exp.No:7

CRC(Cyclic Redundancy Check)

AIM:
Write a program for error detecting code using CRC-CCITT (16-bits).

Theory
It does error checking via polynomial division. In general, a bit string

$$b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0$$

As

$$b_{n-1}X^{n-1} + b_{n-2}\,X^{n-2} + b_{n-3}\,X^{n-3} + \ldots b_2\,X^2 + b_1\,X^1 + b_0$$

Ex: -

10010101110

As

$$X^{10} + X^7 + X^5 + X^3 + X^2 + X^1$$

All computations are done in modulo 2

Algorithm:-

1. Given a bit string, append 0^S to the end of it (the number of 0^s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.
2. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.
3. Define T(x) = B(x) –R(x)

   (T(x)/G(x) => remainder 0)

4. Transmit T, the bit string corresponding to T(x).
5. Let T' represent the bit stream the receiver gets and T'(x) the associated polynomial. The receiver divides $T^1(x)$ by G(x). If there is a 0 remainder, the receiver concludes T = T' and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission.

Program

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#define N strlen(g)

char t[128], cs[128], g[]="10001000000100001";
int a, e, c;
```

```
void xor() {
  for(c=1;c<N;c++) cs[c]=((cs[c]==g[c])?'0':'1');
}
void crc() {
  for(e=0;e<N;e++) cs[e]=t[e];
  do {
    if(cs[0]=='1') xor();
    for(c=0;c<N-1;c++) cs[c]=cs[c+1];
    cs[c]=t[e++];
  }while(e<=a+N-1);
}
void main() {
  clrscr();
  printf("\nEnter poly : "); scanf("%s",t);
  printf("\nGenerating Polynomial is : %s",g);
  a=strlen(t);
  for(e=a;e<a+N-1;e++) t[e]='0';
  printf("\nModified t[u] is :  %s",t);
  crc();
  printf("\nChecksum is : %s",cs);
  for(e=a;e<a+N-1;e++) t[e]=cs[e-a];
  printf("\nFinal Codeword is : %s",t);
  printf("\nTest Error detection 0(yes) 1(no) ? : ");
  scanf("%d",&e);
  if(e==0) {
    printf("Enter position where error is to inserted : ");
    scanf("%d",&e);
    t[e]=(t[e]=='0')?'1':'0';
    printf("Errorneous data   : %s\n",t);
  }
  crc();
  for (e=0;(e<N-1)&&(cs[e]!='1');e++);
  if(e<N-1) printf("Error detected.");
  else printf("No Error Detected.");
  getch();
}
```

Sample Output

```
Enter poly : 1011101
Generating Polynomial is : 10001000000100001
Modified t[u] is :  10111010000000000000000
Checksum is : 1000101101011000
Final Codeword is : 10111010000101101011000
Test Error detection 0(yes) 1(no) ? : 0
Enter position where you want to insert error : 3
Errorneous data   : 1010111010000101101011000
Error detected.

Enter poly : 1011101
Generating Polynomial is : 10001000000100001
Modified t[u] is :  10111010000000000000000
Checksum is : 1000101101011000
Final Codeword is : 10111010000101101011000
Test Error detection 0(yes) 1(no) ? : 1
No Error Detected.
```

Exp.No:8

Distance Vector Routing

AIM:
Write a program for distance vector algorithm to find suitable path for transmission.

Theory

Routing algorithm is a part of network layer software which is responsible for deciding which output line an incoming packet should be transmitted on. If the subnet uses datagram internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the subnet uses virtual circuits internally, routing decisions are made only when a new established route is being set up. The latter case is sometimes called session routing, because a rout remains in force for an entire user session (e.g., login session at a terminal or a file).

Routing algorithms can be grouped into two major classes: adaptive and nonadaptive. Nonadaptive algorithms do not base their routing decisions on measurement or estimates of current traffic and topology. Instead, the choice of route to use to get from *I* to *J* (for all *I* and *J*) is compute in advance, offline, and downloaded to the routers when the network ids booted. This procedure is sometime called static routing.

Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes, or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

Two algorithms in particular, distance vector routing and link state routing are the most popular. Distance vector routing algorithms operate by having each router maintain a table (i.e., vector) giving the best known distance to each destination and which line to get there. These tables are updated by exchanging information with the neighbors.

The distance vector routing algorithm is sometimes called by other names, including the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the RIP and in early versions of DECnet and Novell's IPX. AppleTalk and Cisco routers use improved distance vector protocols.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in subnet. This entry contains two parts: the preferred out going line to use for that destination, and an estimate of the time or distance to that destination. The metric used might be number of hops, time delay in milliseconds, total number of packets queued along the path, or something similar.

The router is assumed to know the "distance" to each of its neighbor. If the metric is hops, the distance is just one hop. If the metric is queue length, the router simply examines each queue. If the metric is delay, the router can measure it directly with special ECHO packets hat the receiver just time stamps and sends back as fast as possible.

The Count to Infinity Problem.

Distance vector routing algorithm reacts rapidly to good news, but leisurely to bad news. Consider a router whose best route to destination *X* is large. If on the next exchange neighbor *A* suddenly reports a

short delay to *X*, the router just switches over to using the line to *A* to send traffic to *X*. In one vector exchange, the good news is processed.

To see how fast good news propagates, consider the five node (linear) subnet of following figure, where the delay metric is the number of hops. Suppose A is down initially and all the other routers know this. In other words, they have all recorded the delay to A as infinity.

A	B	C	D	E		A	B	C	D	E	
∞	∞	∞	∞	Initially		1	2	3	4	Initially
1	∞	∞	∞	After 1 exchange		3	2	3	4	After 1 exchange
1	2	∞	∞	After 2 exchange		3	3	3	4	After 2 exchange
1	2	3	∞	After 3 exchange		5	3	5	4	After 3 exchange
1	2	3	4	After 4 exchange		5	6	5	6	After 4 exchange
						7	6	7	6	After 5 exchange
						7	8	7	8	After 6 exchange
							:			
						∞	∞	∞	∞	

Many ad hoc solutions to the count to infinity problem have been proposed in the literature, each one more complicated and less useful than the one before it. The **split horizon** algorithm works the same way as distance vector routing, except that the distance to *X* is not reported on line that packets for *X* are sent on (actually, it is reported as infinity). In the initial state of right figure, for example, *C* tells *D* the truth about distance to *A* but *C* tells *B* that its distance to *A* is infinite. Similarly, *D* tells the truth to *E* but lies to *C*.

Program

```
#include <conio.h>
#include <iostream.h>

#define MAX 10
int n;

class router {
    char adj_new[MAX],   adj_old[MAX];
    int  table_new[MAX], table_old[MAX];

  public:
    router(){
      for(int i=0;i<MAX;i++) table_old[i]=table_new[i]=99;
    }

    void copy(){
      for(int i=0;i<n;i++) {
        adj_old[i]  =adj_new[i];
        table_old[i]=table_new[i];
      }
```

```
    }

    int equal() {
      for(int i=0;i<n;i++)
        if(table_old[i]!=table_new[i]||adj_new[i]!=adj_old[i])return 0;
      return 1;
    }

    void input(int j) {
      cout<<"Enter 1 if the corresponding router is adjacent to router"
          <<(char)('A'+j)<<" else enter 99: "<<endl<<"             ";
      for(int i=0;i<n;i++)
        if(i!=j) cout<<(char)('A'+i)<<" ";
      cout<<"\nEnter matrix:";
      for(i=0;i<n;i++) {
        if(i==j)
          table_new[i]=0;
        else
          cin>>table_new[i];
        adj_new[i]= (char)('A'+i);
      }
      cout<<endl;
    }

    void display(){
      cout<<"\nDestination Router: ";
      for(int i=0;i<n;i++) cout<<(char)('A'+i)<<" ";
      cout<<"\nOutgoing Line:      ";
      for(i=0;i<n;i++) cout<<adj_new[i]<<" ";
      cout<<"\nHop Count:          ";
      for(i=0;i<n;i++) cout<<table_new[i]<<" ";
    }

    void build(int j) {
      for(int i=0;i<n;i++)
        for(int k=0;(i!=j)&&(k<n);k++)
          if(table_old[i]!=99)
            if((table_new[i]+r[i].table_new[k])<table_new[k]) {
              table_new[k]=table_new[i]+r[i].table_new[k];
              adj_new[k]=(char)('A'+i);
            }
    }
} r[10];

void build_table() {
  int i=0, j=0;
  while(i!=n) {
    for(i=j;i<n;i++) {
      r[i].copy();
      r[i].build(i);
    }
    for(i=0;i<n;i++)
      if(!r[i].equal()) {
        j=i;
        break;
      }
```

```
  }
}

void main() {
  clrscr();
  cout<<"Enter the number the routers(<"<<MAX<<"): "; cin>>n;
  for(int i=0;i<n;i++) r[i].input(i);
  build_table();
  for(i=0;i<n;i++) {
    cout<<"Router Table entries for router "<<(char)('A'+i)<<":-";
    r[i].display();
    cout<<endl<<endl;
  }
  getch();
}
```

Sample Output

```
Enter the number the routers: 5
Enter 1 if the corresponding is adjacent to router A else enter 99:
            B C D E
Enter matrix:1 1 99 99
Enter 1 if the corresponding is adjacent to router B else enter 99:
            A C D E
Enter matrix:1 99 99 99
Enter 1 if the corresponding is adjacent to router C else enter 99:
            A B D E
Enter matrix:1 99 1 1
Enter 1 if the corresponding is adjacent to router D else enter 99:
            A B C E
Enter matrix:99 99 1 99
Enter 1 if the corresponding is adjacent to router E else enter 99:
            A B C D
Enter matrix:99 99 1 99

Router Table entries for router A
Destination Router: A B C D E
Outgoing Line:      A B C C C
Hop Count:          0 1 1 2 2
Router Table entries for router B
Destination Router: A B C D E
Outgoing Line:      A B A A A
Hop Count:          1 0 2 3 3
Router Table entries for router C
Destination Router: A B C D E
Outgoing Line:      A A C D E
Hop Count:          1 2 0 1 1
Router Table entries for router D
Destination Router: A B C D E
Outgoing Line:      C C C D C
Hop Count:          2 3 1 0 2
Router Table entries for router E
Destination Router: A B C D E
Outgoing Line:      C C C C E
Hop Count:          2 3 1 2 0
```

Exp.No:9

TCP Socket

AIM:

Using TCP/IP sockets, write a client-server program to make client sending the file name and the server to send back the contents of the requested file if present.

Algorithm (Client Side)

1. Start.
2. Create a socket using socket() system call.
3. Connect the socket to the address of the server using connect() system call.
4. Send the filename of required file using send() system call.
5. Read the contents of the file sent by server by recv() system call.
6. Stop.

Algorithm (Server Side)

1. Start.
2. Create a socket using socket() system call.
3. Bind the socket to an address using bind() system call.
4. Listen to the connection using listen() system call.
5. accept connection using accept()
6. Receive filename and transfer contents of file with client.
7. Stop.

Program

/*Server*/

```
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>

int main()
{
  int cont,create_socket,new_socket,addrlen,fd;
  int bufsize = 1024;
  char *buffer = malloc(bufsize);
  char fname[256];
  struct sockaddr_in address;

  if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
    printf("The socket was created\n");

  address.sin_family = AF_INET;
  address.sin_addr.s_addr = INADDR_ANY;
  address.sin_port = htons(15000);
```

```
  if (bind(create_socket,(struct sockaddr *)&address,sizeof(address)) == 0)
    printf("Binding Socket\n");
  listen(create_socket,3);
  addrlen = sizeof(struct sockaddr_in);
  new_socket = accept(create_socket,(struct sockaddr *)&address,&addrlen);

  if (new_socket > 0)
     printf("The Client %s is Connected...\n",
            inet_ntoa(address.sin_addr));
  recv(new_socket,fname, 255,0);
  printf("A request for filename %s Received..\n", fname);
  if ((fd=open(fname, O_RDONLY))<0)
    {perror("File Open Failed"); exit(0);}
  while((cont=read(fd, buffer, bufsize))>0) {
     send(new_socket,buffer,cont,0);
  }
  printf("Request Completed\n");
  close(new_socket);
  return close(create_socket);
}
```

/*Client*/

```
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>

int main(int argc,char *argv[])
{
  int create_socket;
  int bufsize = 1024;
  char *buffer = malloc(bufsize);
  char fname[256];
  struct sockaddr_in address;

  if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
    printf("The Socket was created\n");
  address.sin_family = AF_INET;
  address.sin_port = htons(15000);
  inet_pton(AF_INET,argv[1],&address.sin_addr);

  if (connect(create_socket,(struct sockaddr *) &address,
              sizeof(address)) == 0)
    printf("The connection was accepted with the server %s...\n",
           argv[1]);
  printf("Enter The Filename to Request : "); scanf("%s",fname);
  send(create_socket, fname, sizeof(fname), 0);
  printf("Request Accepted... Receiving File...\n\n");
  printf("The contents of file are...\n\n");
  while((cont=recv(create_socket, buffer, bufsize, 0))>0) {
    write(1, buffer, cont);
```

```
  }
  printf("\nEOF\n");
  return close(create_socket);
}
```

Sample Output (Server)

```
[root@localhost CN Lab] ./s.o
Socket Created..
Binding Socket..
Now Listening for Request..

The Client 127.0.0.1 is trying to connect…
A request for filename alpha received..
Requested completed..
[root@localhost CN Lab]
```

Sample Output (Client)

```
[root@localhost CN Lab] ./c.o 127.0.0.1
Socket Created..
Connetion is accepted by 127.0.0.1 ..
Enter File Name to Request : alpha
Requestion for file alpha.. Request accepted. Receiving file…

The contents of file are :-
This a demo of client server using Sockets
Just for trial.
Now End of file

EOF

[root@localhost CN Lab]
```

Exp.No:10

FIFO IPC

AIM:
Implement the above program using as message queues or FIFO as IPC channels.

Algorithm (Client Side)

1. Start.
2. Open well known server FIFO in write mode.
3. Write the pathname of the file in this FIFO and send the request.
4. Open the client specified FIFO in read mode and wait for reply.
5. When the contents of the file are available in FIFO, display it on the terminal
6. Stop.

Algorithm (Server Side)

1. Start.
2. Create a well known FIFO using mkfifo()
3. Open FIFO in read only mode to accept request from clients.
4. When client opens the other end of FIFO in write only mode, then read the contents and store it in buffer.
5. Create another FIFO in write mode to send replies.
6. Open the requested file by the client and write the contents into the client specified FIFO and terminate the connection.
7. Stop.

Program

/*Server*/

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666

char fname[256];

int main() {
  int readfd, writefd, fd;
  ssize_t n;
  char buff[512];
  if (mkfifo(FIFO1, PERMS)<0)
    printf("Cant Create FIFO Files\n");
  if (mkfifo(FIFO2, PERMS)<0)
    printf("Cant Create FIFO Files\n");
  printf("Waiting for connection Request..\n");
  readfd =open(FIFO1, O_RDONLY, 0);
```

```
  writefd=open(FIFO2, O_WRONLY, 0);
  printf("Connection Established..\n");
  read(readfd, fname, 255);
  printf("Client has requested file %s\n", fname);
  if ((fd=open(fname,O_RDWR))<0) {
    strcpy(buff,"File does not exist..\n");
    write(writefd, buff, strlen(buff));
  } else {
    while((n=read(fd, buff,512))>0)
      write(writefd, buff, n);
  }
  close(readfd);  unlink(FIFO1);
  close(writefd);  unlink(FIFO2);
}
```

/*Client*/

```
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>

#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666

char fname[256];

int main()
{
  ssize_t n;
  char buff[512];
  int readfd,writefd;
  printf("Trying to Connect to Server..\n");
  writefd = open(FIFO1, O_WRONLY, 0);
  readfd  = open(FIFO2, O_RDONLY, 0);
  printf("Connected..\n");
  printf("Enter the filename to request from server: ");
  scanf("%s",fname);
  write(writefd, fname, strlen(fname));
  printf("Waiting for Server to reply..\n");
  while((n=read(readfd,buff,512))>0)
    write(1,buff,n);
  close(readfd);
  close(writefd);
  return 0;
}
```

Sample Output (Server)

```
[root@localhost CN Lab] ./s.o
Waiting for connection Request..
Connection Established..
Client has requested file alpha
[root@localhost CN Lab]
```

Sample Output (Client)

```
[root@localhost CN Lab] ./c.o
Trying to Connect to Server..
Connected..
Enter the filename to request from server: alpha
Waiting for Server to reply..

This a demo of client server using Sockets
Just for trial.
Now End of file

[root@localhost CN Lab]
```

Exp.No:11

RSA Algorithm

AIM:
Write a program for simple RSA algorithm to encrypt and decrypt the data.

Theory

Cryptography has a long and colorful history. The message to be encrypted, known as the plaintext, are transformed by a function that is parameterized by a key. The output of the encryption process, known as the ciphertext, is then transmitted, often by messenger or radio. The enemy, or intruder, hears and accurately copies down the complete ciphertext. However, unlike the intended recipient, he does not know the decryption key and so cannot decrypt the ciphertext easily. The art of breaking ciphers is called **cryptanalysis** the art of devising ciphers (cryptography) and breaking them (cryptanalysis) is collectively known as cryptology.

There are several ways of classifying cryptographic algorithms. They are generally categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are as follows:

1. Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption. It is also known as symmetric cryptography.
2. Public Key Cryptography (PKC): Uses one key for encryption and another for decryption. It is also known as asymmetric cryptography.
3. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information

Public-key cryptography has been said to be the most significant new development in cryptography. Modern PKC was first described publicly by Stanford University professor Martin Hellman and graduate student Whitfield Diffie in 1976. Their paper described a two-key crypto system in which two parties could engage in a secure communication over a non-secure communications channel without having to share a secret key.

Generic PKC employs two keys that are mathematically related although knowledge of one key does not allow someone to easily determine the other key. One key is used to encrypt the plaintext and the other key is used to decrypt the ciphertext. The important point here is that it does not matter which key is applied first, but that both keys are required for the process to work. Because pair of keys is required, this approach is also called asymmetric cryptography.

In PKC, one of the keys is designated the public key and may be advertised as widely as the owner wants. The other key is designated the private key and is never revealed to another party. It is straight forward to send messages under this scheme.

The RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

Algorithm

1. Generate two large random primes, P and Q, of approximately equal size.
2. Compute $N = P \times Q$
3. Compute $Z = (P-1) \times (Q-1)$.
4. Choose an integer E, $1 < E < Z$, such that GCD $(E, Z) = 1$

5. Compute the secret exponent D, $1 < D < Z$, such that $E \times D \equiv 1 \pmod{Z}$
6. The public key is (N, E) and the private key is (N, D).

Note: The values of P, Q, and Z should also be kept secret.
The message is encrypted using public key and decrypted using private key.

An example of RSA encryption

1. Select primes P=11, Q=3.
2. $N = P \times Q = 11 \times 3 = 33$
   $Z = (P-1) \times (Q-1) = 10 \times 2 = 20$
3. Lets choose E=3
   Check GCD(E, P-1) = GCD(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1),
   and check GCD(E, Q-1) = GCD(3, 2) = 1
   therefore GCD(E, Z) = GCD(3, 20) = 1
4. Compute D such that $E \times D \equiv 1 \pmod{Z}$
   compute $D = E^{\wedge -1} \bmod Z = 3^{\wedge -1} \bmod 20$
   find a value for D such that Z divides (($E \times D$)-1)
   find D such that 20 divides 3D-1.
   Simple testing (D = 1, 2, ...) gives D = 7
   Check: ($E \times D$)-1 = 3.7 - 1 = 20, which is divisible by Z.
5. Public key = (N, E) = (33, 3)
   Private key = (N, D) = (33, 7).

Now say we want to encrypt the message m = 7,
Cipher code = $M^{\wedge E} \bmod N$
$= 7^{\wedge 3} \bmod 33$
$= 343 \bmod 33$
$= 13.$
Hence the ciphertext c = 13.

To check decryption we compute Message' = $C^{\wedge D} \bmod N$
$= 13^{\wedge 7} \bmod 33$
$= 7.$

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that a = bc mod n = (b mod n).(c mod n) mod n so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

Program

```
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <math.h>

int mult(unsigned int x, unsigned int y, unsigned int n) {
  unsigned long int k=1;
  int j;
  for (j=1; j<=y; j++) k = (k * x) % n;
  return (unsigned int) k;
```

```
}

void main () {
  char msg[100];
  unsigned int pt[100], ct[100], n, d, e, p, q, i;
  printf("Enter message : "); gets(msg);
  //strcpy(pt, msg);
  for(i=0;i<strlen(msg);i++)
   pt[i]=msg[i];
  n = 253; d = 17; e = 13;
  printf("\nCT = ");
  for(i=0; i<strlen(msg); i++) ct[i] = mult(pt[i], e,n);
  for(i=0; i<strlen(msg); i++) printf("%d ", ct[i]);
  printf("\nPT = ");
  for(i=0; i<strlen(msg); i++) printf("%c", pt[i]);
  for(i=0; i<strlen(msg); i++) pt[i] = mult(ct[i], d,n) ;
}
```

Sample Output

```
Enter message : alpha
CT = 113 3 129 213 113
PT = alpha
```

Exp.No:12

Leaky Bucket

AIM:
Write a program for congestion control using Leaky bucket algorithm.

Theory

The congesting control algorithms are basically divided into two groups: open loop and closed loop. Open loop solutions attempt to solve the problem by good design, in essence, to make sure it does not occur in the first place. Once the system is up and running, midcourse corrections are not made. Open loop algorithms are further divided into ones that act at source versus ones that act at the destination.

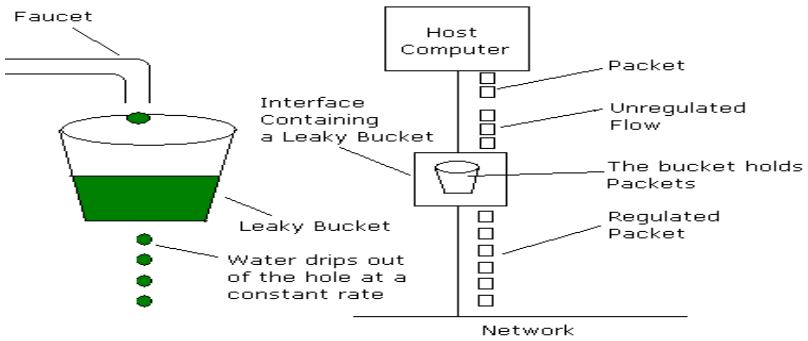
In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two sub categories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn the source. In implicit algorithm, the source deduces the existence of congestion by making local observation, such as the time needed for acknowledgment to come back.

The presence of congestion means that the load is (temporarily) greater than the resources (in part of the system) can handle. For subnets that use virtual circuits internally, these methods can be used at the network layer.

Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This approach to congestion management is widely used in ATM networks and is called **traffic shaping**.

The other method is the leaky bucket algorithm. Each host is connected to the network by an interface containing a leaky bucket, that is, a finite internal queue. If a packet arrives at the queue when it is full, the packet is discarded. In other words, if one or more process are already queued, the new packet is unceremoniously discarded. This arrangement can be built into the hardware interface or simulate d by the host operating system. In fact it is nothing other than a single server queuing system with constant service time.

The host is allowed to put one packet per clock tick onto the network. This mechanism turns an uneven flow of packet from the user process inside the host into an even flow of packet onto the network, smoothing out bursts and greatly reducing the chances of congestion.



Program

```
#include<iostream.h>
#include<dos.h>
#include<stdlib.h>
#define bucketSize 512

void bktInput(int a,int b) {
  if(a>bucketSize)
    cout<<"\n\t\tBucket overflow";
  else {
    delay(500);
    while(a>b){
      cout<<"\n\t\t"<<b<<" bytes outputted.";
      a-=b;
      delay(500);
    }
    if (a>0) cout<<"\n\t\tLast "<<a<<" bytes sent\t";
    cout<<"\n\t\tBucket output successful";
  }
}

void main() {
  int op, pktSize;
  randomize();
  cout<<"Enter output rate : "; cin>>op;
  for(int i=1;i<=5;i++){
    delay(random(1000));
    pktSize=random(1000);
    cout<<"\nPacket no "<<i<<"\tPacket size = "<<pktSize;
    bktInput(pktSize,op);
  }
}
```

Sample Output

```
Enter output rate : 100

Packet no 0  Packet size = 3
                Bucket output successful
                Last 3 bytes sent
Packet no 1  Packet size = 33
                Bucket output successful
                Last 33 bytes sent
Packet no 2  Packet size = 117
                Bucket output successful
                100 bytes outputted.
                Last 17 bytes sent
Packet no 3  Packet size = 95
                Bucket output successful
                Last 95 bytes sent
Packet no 4  Packet size = 949
                Bucket overflow
```

Viva Questions

1. What are functions of different layers?
2. Differentiate between TCP/IP Layers and OSI Layers
3. Why header is required?
4. What is the use of adding header and trailer to frames?
5. What is encapsulation?
6. Why fragmentation requires?
7. What is MTU?
8. Which layer imposes MTU?
9. Differentiate between flow control and congestion control.
10. Differentiate between Point-to-Point Connection and End-to-End connections.
11. What are protocols running in different layers?
12. What is Protocol Stack?
13. Differentiate between TCP and UDP.
14. Differentiate between Connectionless and connection oriented connection.
15. Why frame sorting is required?
16. What is meant by subnet?
17. What is meant by Gateway?
18. What is an IP address?
19. What is MAC address?
20. Why IP address is required when we have MAC address?
21. What is meant by port?
22. What are ephemerical port number and well known port numbers?
23. What is a socket?
24. What are the parameters of socket()?
25. Describe bind(), listen(), accept(),connect(), send() and recv().
26. What are system calls? Mention few of them.
27. What is IPC? Name three techniques.
28. Explain mkfifo(), open(), close() with parameters.
29. What is meant by file descriptor?
30. What is meant by traffic shaping?
31. How do you classify congestion control algorithms?
32. Differentiate between Leaky bucket and Token bucket.
33. How do you implement Leaky bucket?
34. How do you generate busty traffic?
35. What is the polynomial used in CRC-CCITT?
36. What are the other error detection algorithms?
37. What is difference between CRC and Hamming code?
38. Why Hamming code is called 7,4 code?
39. What is odd parity and even parity?
40. What is meant by syndrome?
41. What is generator matrix?
42. What is spanning tree?
43. Where Pirm's algorithm does finds its use in Networks?
44. Differentiate between Prim's and Kruskal's algorithm.
45. What are Routing algorithms?
46. How do you classify routing algorithms? Give examples for each.
47. What are drawbacks in distance vector algorithm?
48. How routers update distances to each of its neighbor?

49. How do you overcome count to infinity problem?
50. What is cryptography?
51. How do you classify cryptographic algorithms?
52. What is public key?
53. What is private key?
54. What are key, ciphertext and plaintext?
55. What is simulation?
56. What are advantages of simulation?
57. Differentiate between Simulation and Emulation.
58. What is meant by router?
59. What is meant by bridge?
60. What is meant by switch?
61. What is meant by hub?
62. Differentiate between route, bridge, switch and hub.
63. What is ping and telnet?
64. What is FTP?
65. What is BER?
66. What is meant by congestion window?
67. What is BSS?
68. What is incoming throughput and outgoing throughput?
69. What is collision?
70. How do you generate multiple traffics across different sender-receiver pairs?
71. How do you setup Ethernet LAN?
72. What is meant by mobile host?
73. What is meant by NCTUns?
74. What are dispatcher, coordinator and nctunsclient?
75. Name few other Network simulators
76. Differentiate between logical and physical address.
77. Which address gets affected if a system moves from one place to another place?
78. What is ICMP? What are uses of ICMP? Name few.
79. Which layer implements security for data?

Guidelines For installation of NCTUns Network Simulator

- Follow the following steps carefully,
- Please don't skip any steps that have been mentioned below

1. Install any Linux with kernel 2.6.9 (PCQ Linux 2004 is exception)
   Recommended → RED HAT LINUX ENTERPRISE EDITION
2. After installation Boot into Linux as root.
3. Copy the .tgz installation file of NCTUns that you got from college to the folder /*bin/local*
   Please don't change any folder name in this folder that is created after unzipping the above file. Don't even change the Case of the folder that is created
4. Now unzip the .tgz file by opening the terminal and changing the directory to */bin/local* by the command :-
   [root@localhost ~] cd /bin/local

   To unzip use the following command:-
   [root@localhost local] tar xvzf [the file name].tar

   (Note there is no '-' before xvzf)
   This will create a folder called NCTUns in the directory /bin/local...

5. Now disable the Secure Linux option by running the following command :-
   [root@localhost local] *vi /etc/selinux/config*

   When the file opens, there is a line similar to -- SELINUX=enforcing
   Change the "*enforcing*" to "*disabled*" (Note without quotes)

6. From the directory */bin/local* change the current working directory to *NCTUns* by following command :-
   [root@localhost local] *cd NCTUns*

7. Now from here execute the installation shell script that will do the required compilations and settings for you:-
   [root@localhost local] *./install.sh*

   During this part it will ask for installation of tunnel files. Please type yes and Enter to continue

8. If the installation is successful, it will display the success message at the end. Now restart your computer. You will find a new entry in GRUB Menu *"NCTUns kernel login*". Boot into Linux using this entry.
9. Log in as *root*. Now you have to modify any *.bash_profile* file
   [root@localhost ~] *vi .bash_profile*

   The Content of file should look like as follows.

```
# .bash_profile

# Get the aliases and function
```

```
if [ -t ~/.bashrc ]; then
      . ~/.bashrc
fi

# User specified environment variables and startup programs

PATH=$PATH:$HOME/bin:/usr/local/nctuns/bin
export LD_LIBRARY_PATH=/usr/local/nctuns/lib
export NCTUNSHOME=/usr/local/nctuns

export PATH
export USERNAME
```

If it's not like above, change it to look like above.

10. Now save this file and log off and then log on again.
11. Create another user account.
12. before using simulator, please execute the following command
    [root@localhost ~] *iptables -F*
13. Run the simulator using three commands where each command should be executed in different window.
    [root@localhost ~] *dispatcher*
    [root@localhost ~] *coordinator*
    [root@localhost ~] *nctunsclient*

14. In the NCTUns window Settings → Dispatcher. Provide the username and password of the user account u created in step 11. Then Click OK.