Computer Networks Laboratory (17ECL68)

LABORATORY MANUAL

VI Semester B.E.

(Academic Year: 2019-20)

Insert any equipment/lab related pictures

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management
Adyar, Mangaluru - 575007

STUDENT DETAILS

NAME	
USN	
BATCH	
SECTION	

STAFF DETAILS

NAME OF THE FACULTY	
NAME OF THE INSTRUCTOR	

EVALUATION

	MAXIMUM MARKS	MARKS OBTAINED
WEEKLY EVALUATION*		
LAB CIE		
TOTAL		

*OBE Sheet Evaluation

Signature of the Faculty

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management
Adyar, Mangaluru - 575007

Institutional Vision & Mission

VISION

To be a premier institution in Technology and Management by fostering excellence in education, innovation, incubation and values to inspire and empower the young minds.

MISSION

M1: Creating an academic ambience to impart holistic education focusing on individual growth, integrity, ethical values and social responsibility.

M2: Develop skill based learning through industry-institution interaction to enhance competency and promote entrepreneurship.

M3: Fostering innovation and creativity through competitive environment with state of-the-art infrastructure.

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575007

Department Vision & Mission

VISION

To establish the department as a center of excellence in creating globally competitive, socially responsible engineers to excel in the field of Electronics and Communication by transforming future challenges to sustainable opportunities.

MISSION

M1: Inculcating a distinctive teaching learning process to provide extensive knowledge of principles and solutions to challenges in the relevant domain.

M2: Nurturing the growth of every individual through inventive, dynamic and conductive learning environment using modern education techniques and industry-oriented pedagogy.

M3: Imparting leadership qualities with ethical values among students to cater societal and environmental needs.

Course Outcomes

COs	Description	Bloom's Level
C328.1	Use the network simulator for learning and practice of networking algorithms.	CL3
C328.2	Simulate the network with different configurations to measure the performance parameters.	CL3
C328.3	Implement HLDC frame using C programming	CL3
C328.4	Illustrate the operations of network protocols and algorithms using C programming	CL3
C328.5	Implement the routing protocols using C programming.	CL3

CO-PO-PSO Mapping

POs / COs	1	2	3	4	5	6	7	8	9	10	11	12	PSO1	PSO2
C328.1	3	3	1		2			1	1	1				2
C328.2	3	3	1		2			1	1	1				2
C328.3	3	3	1		2			1	1	1				2
C328.4	3	3	1		2			1	1	1				2
C328.5	3	3	1		2			1	1	1				2

Assessment Components (ACs)

Assessment Component (AC)	Assessment Component Description
AC1: Written Work	Observation and Record
AC2: Fundamental Knowledge to conduct experiment	Discussion on Pre requisites to conduct each experiment
AC3: Viva	Pre and post experimental questions asked to the individual students
AC4: Interaction during conduction of Experiment	Explanation regarding the concept of the individual experiment carried out
AC5: Punctuality	On time record submission and login timings of each laboratory session

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management
Adyar, Mangaluru - 575007

List of Experiments

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575007

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.

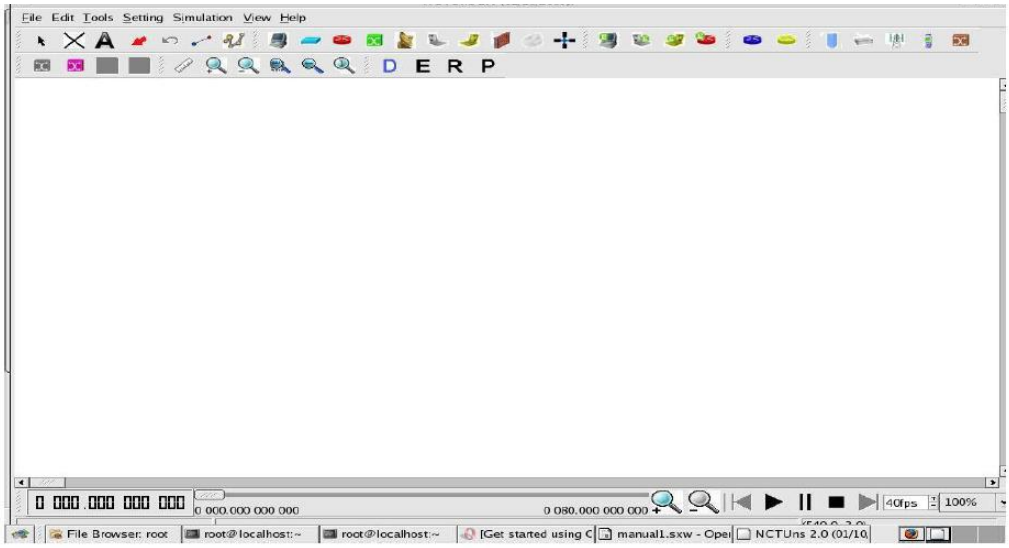
Simulation is only of use if the results are accurate, an inaccurate simulator is not useful at all. Most network simulators use abstractions of network protocols, rather than the real thing, making their results less convincing. S.Y. Wang reports that the simulator OPNET uses a simplified finite state 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 nctunsclient 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:

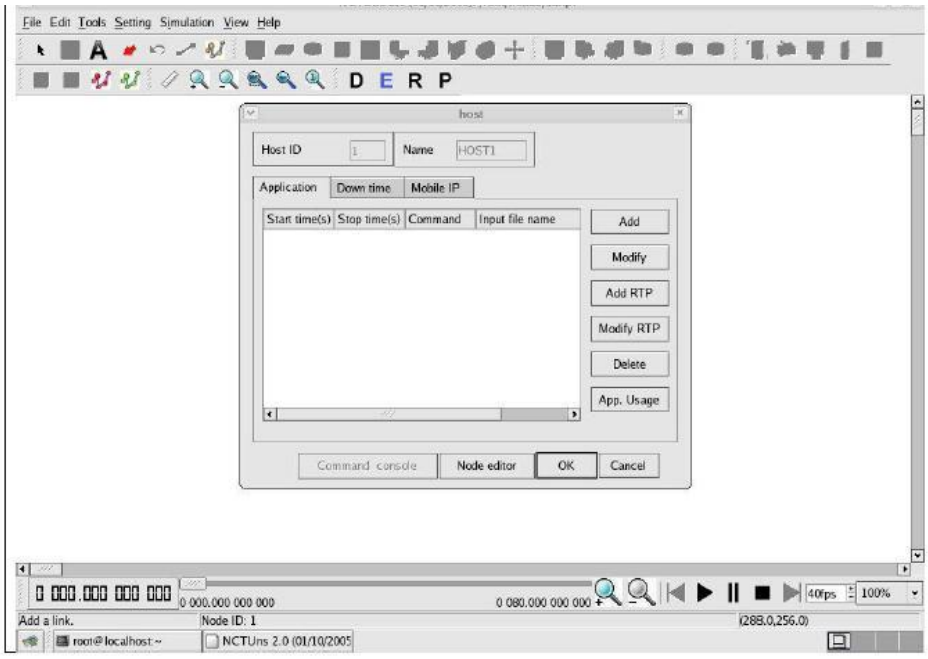
Choose Menu->File->Operating Mode-> and make sure that the "Draw Topology" mode is checked. This is the default mode of NCTUns when it is launched. It is only in this mode that a user can draw a new network topology or change an existing simulation topology. When a user switches the mode to the next 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.

EXPERIMENT No. 01: SIMULATE TO FIND THE NUMBER OF PACKETS DROPPED

1.1 Objectives:

Implement a point to point network with four nodes and duplex links between them. Analyze the network performance by setting the queue size and varying the bandwidth.

1.2 Apparatus Required:

NCTUns 5.0.

1.3 Pre-Requisite:

Computer networks, network topology.

1.4 Introduction:

Point-to-point networks are used to connect one location to another location. Point to Point topology is the simplest topology that connects two nodes directly together with a common link. The entire bandwidth of the common link is reserved for transmission between those two nodes. The point-to-point connections use an actual length of wire or cable to connect the two ends, but other options, such as satellite links, or microwaves are also possible. The typical applications include Internet, Intranet or Extranet configurations, ISP access networks, LAN to LAN applications.

The transfer of data in a point-to-point topology can be in multiple ways across the network: in a simplex, in full duplex, or half duplex.

- In Simplex mode of communication, signal flows in ONE direction and only one node transmit and the other receives.
- In Half duplex mode of communication, each node can transmit and receive but NOT at the same time.
- In Full-duplex mode of communication, both stations transmit and receive simultaneously.

1.5 Procedure:

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.
   stcp –p 2000 –l 1024 1.0.1.3
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.
   rtcp –p 2000 –l 1024
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 model 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

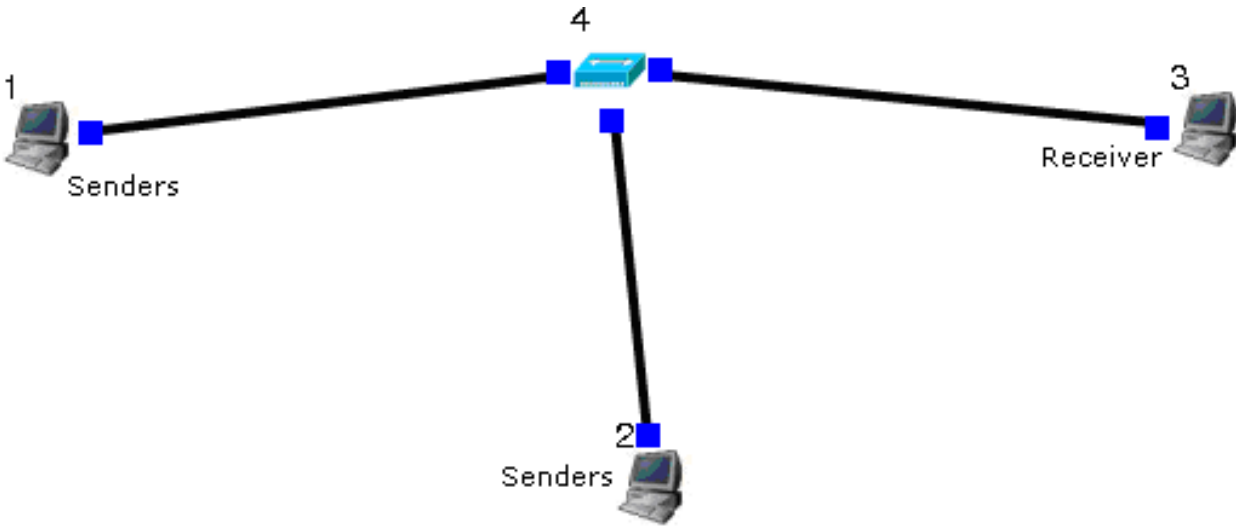
1. Click "R" icon on the tool bar.
2. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
3. To start playback select "►" icon located at the bottom right corner of the editor.
4. To view results, Open up new TERMINAL window, move to file1.results folder and open collision and drop log files in separate TERMINAL window.

Changing configurations

To change the Bandwidth

1. Open the above file.
2. Do not change the topology or any other configuration.
3. Select E icon on the toolbar.
4. Click NODE EDITOR Button on the HOST (Receiver) window and select on the Physical layer and change the bandwidth.
5. Repeat Step3 (Simulate).

1.6 Observations:



a) TCP

Case 1:

Sender 1:-

stcp –p 2000 –l 1024 1.0.1.3

Simulation time: 0 to 40 sec

Sender 2:-

stcp –p 3000 –l 1024 1.0.1.3

Simulation time: 30 to 60 sec

Receiver:-

rtcp –p 2000 –l 1024

Simulation time: 0 to 40 sec

rtcp –p 3000 –l 1024

Simulation time: 30 to 60 sec

Case 2: Change Bandwidth to 8Mbps

b) UDP

Case 1:

Sender 1:-

stg –u 2000 1024 1.0.1.3

Simulation time: 0 to 40 sec

Sender 2:-

stg –u 3000 1024 1.0.1.3

Simulation time: 30 to 60 sec

Receiver:-

rtg –u 2000

Simulation time: 0 to 40 sec

rtg –u 3000

Simulation time: 30 to 60 sec

Case 2: Change Bandwidth to 8Mbps

Parameters:-

Outthroughput, Inthroughput, and Drop Packets

1.7 Results :

1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. What do you mean by data communication?
2. What is simplex?
3. What is half-duplex?
4. What is full duplex?
5. What is a network?

1.10 Post – Experimentation Questions:

1. What is distributed processing?
2. What is point to point connection?
3. What is multipoint connection?
4. What is a topology?
5. Define LAN

EXPERIMENT No. 02: SIMULATE TO FIND THE NUMBER OF PACKETS DROPPED BY TCP/UDP

2.1 Objectives:

Implement a four node point to point network with links n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP between n1-n3. Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP.

2.2 Apparatus Required:

NCTUns 5.0.

2.3 Pre-Requisite:

TCP/IP, Transport Layer

2.4 Introduction:

TCP (Transmission Control Protocol):

TCP recovers data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver side sequence number is used to eliminate the duplicates as well as to order the segments in correct order since there is a chance of "out of order" reception. Therefore, in TCP no transmission errors will affect the correct delivery of data.

UDP (User Datagram Protocol):

UDP uses a simple transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. As this is normally IP over unreliable media, there is no guarantee of delivery, ordering or duplicate protection.

2.5 Procedure:

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.
   stcp –p 2000 –l 1024 1.0.1.3
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.
   rtcp –p 2000 –l 1024
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 model 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

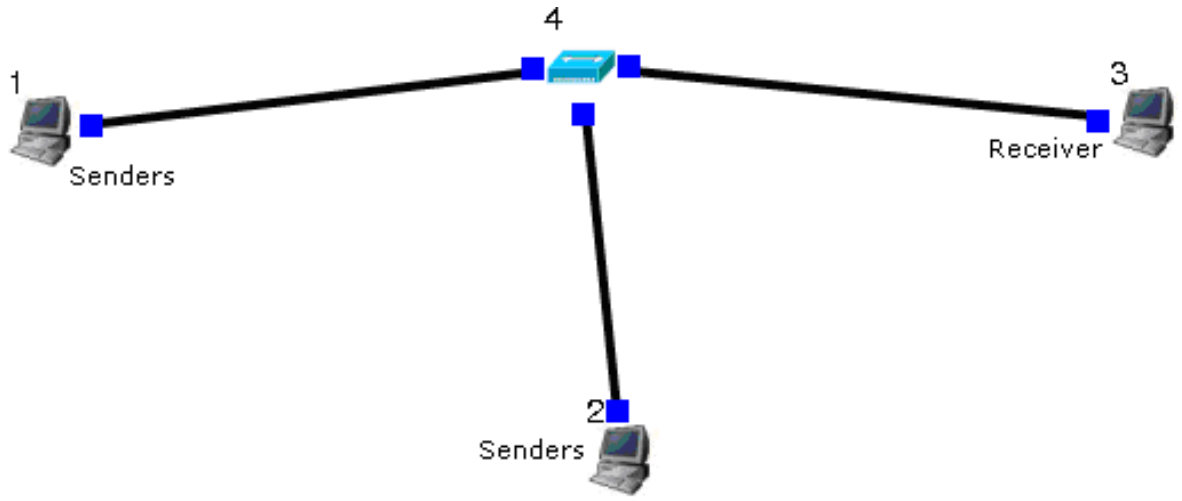
1. Click "R" icon on the tool bar.
2. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
3. To start playback select "►" icon located at the bottom right corner of the editor.
4. To view results, Open up new TERMINAL window, move to file1.results folder and open collision and drop log files in separate TERMINAL window.

Changing configurations

To change the Bandwidth

1. Open the above file.
2. Do not change the topology or any other configuration.
3. Select E icon on the toolbar.
4. Click NODE EDITOR Button on the HOST (Receiver) window and select on the Physical layer and change the bandwidth.
5. Repeat Step3 (Simulate).

2.6 Observations:



Sender 1:-

stcp –p 2000 –l 1024 IP address of receiver

Simulation time: 0 to 40 sec

Sender 2:-

stg –u 3000 1024 IP address of receiver

Simulation time: 30 to 60 sec

Receiver:-

rtcp –p 2000 –l 1024

Simulation time: 0 to 40 sec

rtg –u 3000

Simulation time: 30 to 60 sec

Note: If switch is used you can give same simulation time. If Hub is used give two different simulation time because in a hub only 2 nodes can be active at a time.

Parameters:-

Throughput of incoming and outgoing Packets

2.7 Results :

2.8 Discussions:

2.9 Pre – Experimentation Questions:

1. Define MAN
2. Define WAN
3. Define internet?
4. What is a protocol?
5. What is TCP/IP protocol model?

2.10 Post – Experimentation Questions:

1. Describe the functions of five layers in TCP/IP protocol?
2. What is ISO-OSI model?
3. What is multiplexing?
4. What is switching?
5. How data is transmitted over a medium?

EXPERIMENT No. 03: SIMULATE TO COMPARE DATA RATE & THROUGHPUT

3.1 Objectives:

Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate.

3.2 Apparatus Required:

NCTUns 5.0.

3.3 Pre-Requisite:

Ethernet LAN, error rate, data rate, throughput.

3.4 Introduction:

Bit error rate (BER):

The bit error rate or bit error ratio is the number of bit errors divided by the total number of transferred bits during a studied time interval i.e. BER=Bit errors/Total number of bits

Packet Error Rate (PER):

The PER is the number of incorrectly received data packets divided by the total number of received packets. A packet is declared incorrect if at least one bit is erroneous.

Data Rate:

Data Rate is the speed at which data can be transmitted from one device to another. It is often measured in megabits (million bits) per second.

3.5 Procedure:

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.

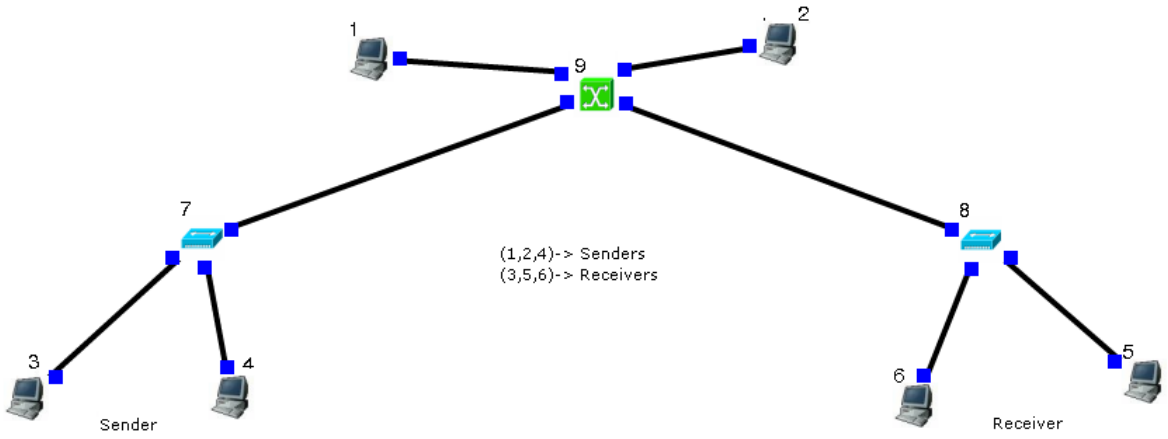
2. Select/click the HUB icon on the toolbar and click the left mouse button on the editor, to place HUB1 on the editor.
3. Click on the LINK icon on the toolbar and connect HOST1 to HUB1 and HUB1 to HOST2.
4. Click on the "E" icon on the toolbar to save the current topology. **e.g:** file1.tpl (Look 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.
   stcp –p 2000 –l 1024 1.0.1.3
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.
   rtcp –p 2000 –l 1024
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.

3.6 Observations:



Case 1:

a) Sender 1:

stcp –p 2000 –l 512 ip address of receiver 1

Sender 2:

stcp –p 3000 –l 512 ip address of receiver 2

Sender 3:

stcp –p 4000 –l 512 ip address of receiver 3

Simulaton Time: 0 to 60 sec.

Reciever 1:

rtcp –p 2000 –l 512

Reciever 2:

rtcp –p 3000 –l 512

Reciever 3:

rtcp –p 4000 –l 512

Simulaton Time: 0 to 60 sec.

Case 2:

b) Double click on all the receiver links and change BER to 0.0001, Run Again.

Parameters:-

Throughput of outgoing Packets

3.7 Results :

3.8 Discussions:

3.9 Pre – Experimentation Questions:

1. Compare analog and digital signals?
2. Define bandwidth?
3. What are the factors on which data rate depends?
4. Define bit rate and bit interval?
5. What is Nyquist bit rate formula?

3.10 Post – Experimentation Questions:

1. Compare analog and digital signals?
2. Define bandwidth?
3. What are the factors on which data rate depends?
4. Define bit rate and bit interval?
5. What is Nyquist bit rate formula?

EXPERIMENT No. 04: SIMULATE TO PLOT CONGESTION FOR DIFFERENT SOURCE/DESTINATION

4.1 Objectives:

Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.

4.2 Apparatus Required:

NCTUns 5.0.

4.3 Pre-Requisite:

Media access control, collision detection, Carrier-sense multiple access.

4.4 Introduction:

Carrier Sense Multiple Access Collision Detection (CSMA/CD) - Working of the truncated binary back off algorithm

In Ethernet networks, the CSMA/CD is commonly used to schedule retransmissions after collisions. The retransmission is delayed by an amount of time derived from the slot time and the number of attempts to retransmit.

After c collisions, a random number of slot times between 0 and 2^c - 1 is chosen. For the first collision, each sender will wait 0 or 1 slot times. After the second collision, the senders will wait anywhere from 0 to 3 slot times (inclusive). After the third collision, the senders will wait anywhere from 0 to 7 slot times (inclusive), and so forth. As the number of retransmission attempts increases, the number of possibilities for delay increases exponentially. The 'truncated' simply means that after a certain number of increases, the exponentiation stops; i.e. the retransmission timeout reaches a ceiling, and thereafter does not increase any further. For example, if the ceiling is set at i = 10 (as it is in the IEEE 802.3 CSMA/CD standard), then the maximum delay is 1023 slot times.

Because these delays cause other stations that are sending to collide as well, there is a possibility that, on a busy network, hundreds of people may be caught in a single collision set. Because of this possibility, after 16 attempts at transmission, the process is aborted.

4.5 Procedure:

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.
   stcp –p 2000 –l 1024 1.0.1.3
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.
   rtcp –p 2000 –l 1024
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 model 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

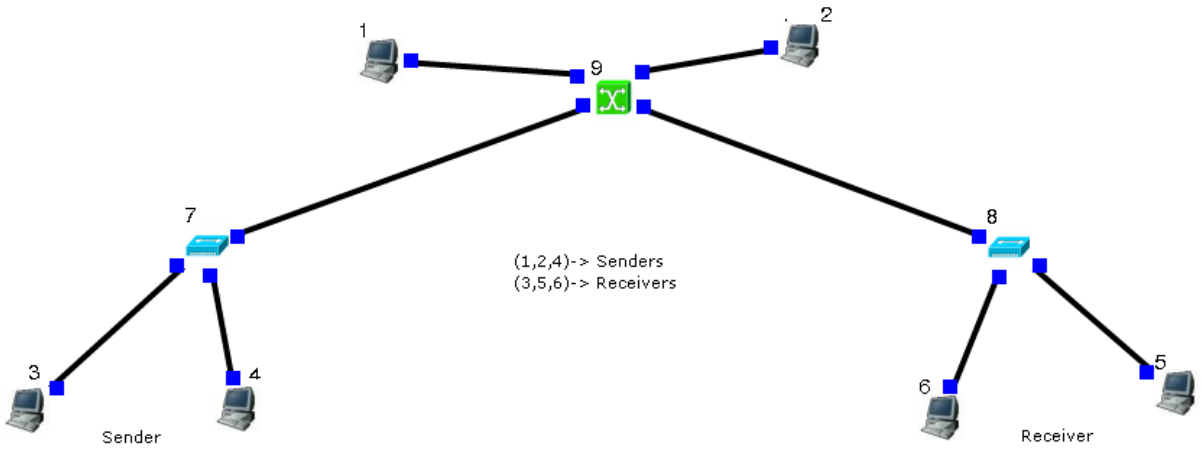
1. Click "R" icon on the tool bar.
2. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
3. To start playback select "►" icon located at the bottom right corner of the editor.
4. To view results, Open up new TERMINAL window, move to file1.results folder and open collision and drop log files in separate TERMINAL window.

Changing configurations

To change the Bandwidth

1. Open the above file.
2. Do not change the topology or any other configuration.
3. Select E icon on the toolbar.
4. Click NODE EDITOR Button on the HOST (Receiver) window and select on the Physical layer and change the bandwidth.
5. Repeat Step3 (Simulate).

4.6 Observations:



a) Sender 1:

stcp –p 2000 –l 512 ip address of receiver 1

stcp –p 2000 –l 512 ip address of receiver 2

Sender 2:

stcp –p 3000 –l 512 ip address of receiver 2

stcp –p 3000 –l 512 ip address of receiver 3

Sender 3:

stcp –p 4000 –l 512 ip address of receiver 1

stcp –p 4000 –l 512 ip address of receiver 2

stcp –p 4000 –l 512 ip address of receiver 3

Simulaton Time: 0 to 60 sec.

Reciever 1:

rtcp –p 2000 –l 512

rtcp –p 4000 –l 512

Reciever 2:

rtcp –p 3000 –l 512

rtcp –p 4000 –l 512

Reciever 3:

rtcp –p 4000 –l 512

rtcp –p 2000 –l 512

rtcp –p 3000 –l 512

Simulaton Time: 0 to 60 sec.

Parameters:-

Receiver side Collision Packets and Drop Packets

4.7 Results :

4.8 Discussions:

4.9 Pre – Experimentation Questions:

1. Define Shannon Capacity?
2. What is sampling?
3. What are the modes of data transmission?
4. What is Asynchronous mode of data transmission?
5. What is Synchronous mode of data transmission?

4.10 Post – Experimentation Questions:

1. What are the different types of multiplexing?
2. What are the different transmission media?
3. What are the different Guided Media?
4. What do you mean by wireless communication?
5. What are the switching methods?

EXPERIMENT No. 05: SIMULATE TO DETERMINE THE PERFORMANCE WITH RESPECT TO TRANSMISSION OF PACKETS

5.1 Objectives:

Implement ESS with transmission nodes in Wireless LAN and obtain the performance parameters.

5.2 Apparatus Required:

NCTUns 5.0.

5.3 Pre-Requisite:

Wireless LAN, transmission nodes, ESS

5.4 Introduction:

Wireless LAN is basically a LAN that transmits data over air, without any physical connection between devices. The transmission medium is a form of electromagnetic radiation. Wireless LAN is ratified by IEEE in the IEEE 802.11 standard. In most of the WLAN products on the market based on the IEEE 802.11b technology the transmitter is designed as a Direct Sequence Spread Spectrum Phase Shift Keying (DSSS PSK) modulator, which is capable of handling data rates of up to 11 Mbps. The underlying algorithm used in Wireless LAN is known as the CSMA/CA – Carrier Sense Multiple Access/Collision Avoidance algorithm.

5.5 Procedure:

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.
   stcp –p 2000 –l 1024 1.0.1.3
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.
   rtcp –p 2000 –l 1024
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 model 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

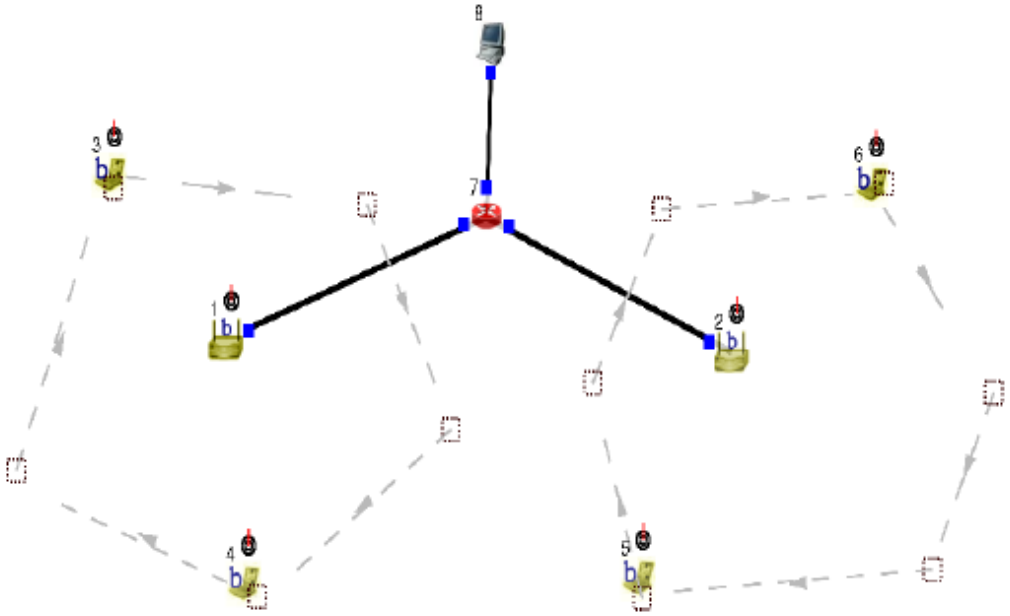
1. Click "R" icon on the tool bar.
2. Select Simulation in the menu bar and click/ select RUN in the dropdown list to execute the simulation.
3. To start playback select "►" icon located at the bottom right corner of the editor.
4. To view results, Open up new TERMINAL window, move to file1.results folder and open collision and drop log files in separate TERMINAL window.

Changing configurations

To change the Bandwidth

1. Open the above file.
2. Do not change the topology or any other configuration.
3. Select E icon on the toolbar.
4. Click NODE EDITOR Button on the HOST (Receiver) window and select on the Physical layer and change the bandwidth.
5. Repeat Step3 (Simulate).

5.6 Observations:



Mobile Host 1:

ttcp –t –u –s –p 8000 ip address of receiver

ttcp –t –u –s –p 8001 ip address of receiver

Host (Receiver):

ttcp –r –u –s –p 8000

ttcp –r –u –s –p 8001

Parameters:-

Throughput of incoming and outgoing packets

5.7 Results :

5.8 Discussions:

5.9 Pre – Experimentation Questions:

1. What are the duties of data link layer?
2. What are the types of errors?
3. What do you mean by redundancy?
4. Define parity check
5. What do you mean by flow control?

5.10 Post – Experimentation Questions:

1. What are functions of different layers?
2. Differentiate between TCP/IP Layers and OSI Layers
3. What is the use of adding header and trailer to frames?
4. What is encapsulation?
5. Why fragmentation requires?

PART – B

C Programs

Try a simple C program:

Program to Display "Hello, World!"

```
#include <stdio.h>
int main()
{
// printf() displays the string inside quotation
printf("Hello, World!");
return 0;
}
```

EXPERIMENT No. 01:
Write a program for a HLDC frame to perform the following.
1. Bit stuffing.
2. Character stuffing.

1.1 Objectives:

Write a program for a HLDC frame to perform the following.

1. Bit stuffing
2. Character stuffing.

1.2 Apparatus Required:

NCTUns 5.0.

1.3 Pre-Requisite:

Data link layer, Connection oriented connection.

1.4 Introduction:

HDLC is based on IBM's SDLC protocol, which is the layer 2 protocol for IBM's Systems Network Architecture (SNA). It was extended and standardized by the ITU as LAP (Link Access Procedure), while ANSI named their essentially identical version ADCCP.

The HDLC specification does not specify the full semantics of the frame fields. This allows other fully compliant standards to be derived from it, and derivatives have since appeared in innumerable standards. It was adopted into the X.25 protocol stack as LAPB, into the V.42 protocol as LAPM, into the Frame Relay protocol stack as LAPF and into the ISDN protocol stack as LAPD.

The original ISO standards for HDLC are the following:

ISO 3309-1979 – Frame Structure

ISO 4335-1979 – Elements of Procedure

ISO 6159-1980 – Unbalanced Classes of Procedure

ISO 6256-1981 – Balanced Classes of Procedure

ISO/IEC 13239:2002, the current standard, replaced all of these specifications.

HDLC was the inspiration for the IEEE 802.2 LLC protocol, and it is the basis for the framing mechanism used with the PPP on synchronous lines, as used by many servers to connect to a WAN, most commonly the Internet.

A similar version is used as the control channel for E-carrier (E1) and SONET multichannel telephone lines. Cisco HDLC uses low-level HDLC framing techniques but adds a protocol field to the standard HDLC header.

1.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

1.6 Observations:

1. Bit stuffing

```
#include<stdio.h>
#include<string.h>
int si,di;
char src[30],dst[30],flag[30]="01111110";
int main()
{
  di=strlen(flag);
  printf("Enter message string\n");
  scanf("%s",&src);
  strcpy(dst,flag);
  while(src[si]!='\0')
```

```
  if(src[si]=='1'&&src[si+1]=='1'&&src[si+2]=='1'&&src[si+3]=='1'&&src[si+4]=='1')
        {
            dst[di]='1' , dst[di+1]='1' , dst[di+2]='1' , dst[di+3]='1' , dst[di+4]='1' ,

            dst[di+5]='0';
            di+=6;
            si+=5;
        }
        else
            dst[di++]=src[si++];
  dst[di++]='\0';
  printf("string is %s \nstuffed string is %s \n",src,strcat(dst,flag));
}
```

```
  Enter message string      1    1    1    1
  String is                 1    1    1    1
  Stuffed string is         01111110111101111110
```

2. Character stuffing

```
#include<stdio.h>
#include<string.h>
int si,di;
char src[30],dst[30],flag1[30]="DLESTX",flag2[30]="DLEETX";
int main()
{
  di=strlen(flag1);
  printf("Enter message string\n");
  scanf("%s",&src);
  strcpy(dst,flag1);
  while(src[si]!='\0')
        if(src[si]=='D'&&src[si+1]=='L'&&src[si+2]=='E')
```

```
            {
                dst[di]='D' , dst[di+1]='L' , dst[di+2]='E' , dst[di+3]='D' ,
                dst[di+4]='L' , dst[di+5]='E';
                di+=6;
                si+=3;
            }
            else
                dst[di++]=src[si++];
    dst[di++]='\0';
    printf("string is %s \nstuffed string is %s \n",src,strcat(dst,flag2));
}
```

Enter message string sahyadri

String is sahyadri

Stuffed string is DLESTXsahyadriDLEETX

1.7 Results :

1.8 Discussions:

1.9 Pre – Experimentation Questions:

1. Define TCP?
2. Define UDP?
3. Define IP?
4. .What do you mean by client server model?
5. What is the information that a computer attached to a TCP/IP internet must possesses?

1.10 Post – Experimentation Questions:

1. Differentiate between TCP and UDP?
2. Differentiate between bit stuffing & character stuffing?
3. Advantages of bit stuffing?
4. Advantages of character stuffing?
5. Differentiate between Point-to-Point Connection and End-to-End connections.

EXPERIMENT No. 02:
Write a program for distance vector algorithm to find suitable path for transmission.

2.1 Objectives:

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

2.2 Apparatus Required:

NCTUns 5.0.

2.3 Pre-Requisite:

Data network, routers, routing tables of routers

2.4 Introduction:

Distance Vector Routing is one of the routing algorithms used in a Wide Area Network for computing shortest path between source and destination. The router is one of the main devices used in a wide area network. The main task of the router is routing. It forms the routing table and delivers the packets depending upon the routes in the table – either directly or via an intermediate device (perhaps another router). Each router initially has information about its all neighbors (i.e., it is directly connected). After a period of time, each router exchanges its routing table among its neighbors. After certain number of exchanges, all routers will have the full routing information about the area of the network. After each table exchange, router re-computes the shortest path between the routers. The algorithm used for this routing is called Distance Vector Routing.

Algorithm:

Repeat the following steps until there is no change in the routing table for all routers.

- Take the Next Router routing table and its neighbor routing table.
- Add the router entry that is not in your own routing table, but exists in any one of the other routing tables. If the new router entry exists in more than one neighbor, then find the

minimum cost among them. The minimum cost value details are taken as a new entry: such as source router, intermediate router, destination router and cost value, etc.

- Update the source router routing table cost value if both the destination router and the intermediate router field value have the same value as any one of the neighbors" routing entry.
- Update the source router's routing table entry with the new advertised one if the intermediate router value in the source table is not same, but the cost value is greater than the its neighbor's entry.
- Write the next stage of routing table into the file.

Repeat steps 1 to 5 for all routers. Check whether any changes are made in any of the routers. If yes, then repeat the above steps, otherwise, quit the process.

2.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

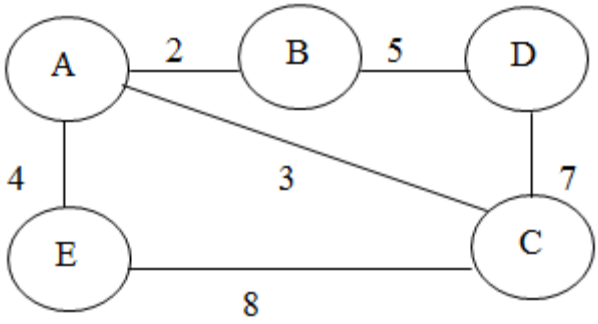
By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

2.6 Observations:

```
#include<stdio.h>
int dist[10][10],nextnode[10][10],cost[10][10],n;
void distvector()
{
  int i,j,k;
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
    {
```

```
      dist[i][j]=cost[i][j];
      nextnode[i][j]=j;
    }
    for(i=0;i<n;i++)
      for(j=0;j<n;j++)
        for(k=0;k<n;k++)
          if(dist[i][j]>cost[i][k]+dist[k][j])
          {
            dist[i][j]=dist[i][k]+dist[k][j];
            nextnode[i][j]=k;
          }
}
int main()
{
  int i,j;
  printf("\nenter the no of vertices:\t");
  scanf("%d",&n);
  printf("\nenter the cost matrix\n");
  for(i=0;i<n;i++)
    for(j=0;j<n;j++)
      scanf("%d",&cost[i][j]);
  distvector();
  for(i=0;i<n;i++)
  {
    printf("\nstate value for router %c \n",i+65);
    printf("\ndestnode\tnextnode\tdistance\n");
    for(j=0;j<n;j++)
    {
      if(dist[i][j]==99)
        printf("%c\t\t-\t\tinfinite\n",j+65);
      else
        printf("%c\t\t%c\t\t%d\n",j+65,nextnode[i][j]+65,dist[i][j]);
    } }
}
```



Enter no fo vertices 5

Enter the cost matrix

99	2	3	99	4
2	99	99	5	99
3	99	99	7	8
99	5	7	99	99
4	99	8	99	99

State value for router A

destnode	nextnode	distance
A	B	4
B	B	2
C	C	3
D	B	7
E	E	4

State value for router B

destnode	nextnode	distance
A	A	2
B	A	4
C	A	5
D	D	5
E	A	6

State value for router C

destnode	nextnode	distance
A	A	3
B	A	5
C	A	6
D	D	7
E	A	7

State value for router D

destnode	nextnode	distance
A	B	7
B	B	5
C	C	7
D	B	10
E	B	11

State value for router E

destnode	nextnode	distance
A	A	4
B	A	6
C	A	7
D	A	11
E	A	8

2.7 Results :

2.8 Discussions:

2.9 Pre – Experimentation Questions:

1. What are Routing algorithms?
2. How do you classify routing algorithms? Give examples for each.
3. What are drawbacks in distance vector algorithm?
4. How routers update distances to each of its neighbor?
5. How do you overcome count to infinity problem?

2.10 Post – Experimentation Questions:

1. What are the design issue of layers?
2. What are the protocols in application layer?
3. What is domain name system (DNS)?
4. Differentiate between Connectionless and connection oriented connection.
5. What are protocols running in different layers?

EXPERIMENT No. 03:
Implement Dijkstras algorithm to compute the shortest routing path.

3.1 Objectives:

Implement Dijkstras algorithm to compute the shortest routing path.

3.2 Apparatus Required:

NCTUns 5.0.

3.3 Pre-Requisite:

Fundamentals of routing algorithms.

3.4 Introduction:

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent, for example, road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956 and published three years later.

The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph, producing a shortest-path tree.

For a given source node in the graph, the algorithm finds the shortest path between that node and every other. It can also be used for finding the shortest paths from a single node to a single destination node by stopping the algorithm once the shortest path to the destination node has been determined. For example, if the nodes of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities. A widely used application of shortest path algorithm is network routing protocols.

The Dijkstra algorithm uses labels that are positive integers or real numbers, which are totally ordered. It can be generalized to use any labels that are partially ordered, provided the subsequent labels (a subsequent label is produced when traversing an edge) are monotonically non-decreasing. This generalization is called the generic Dijkstra shortest-path algorithm.

3.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

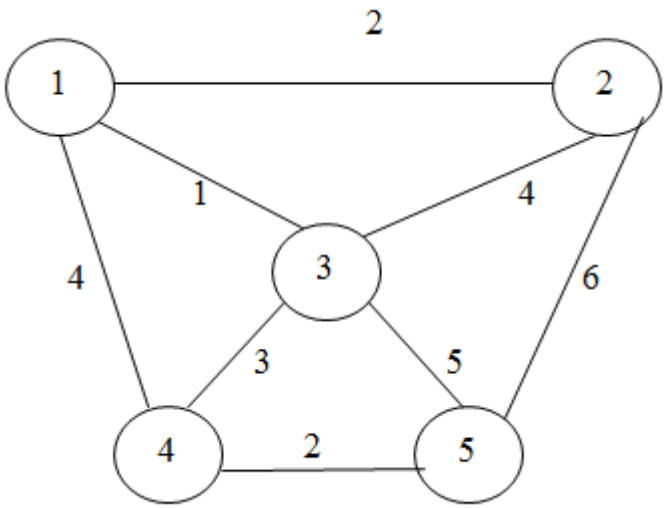
In terminal type the command **# ./a.out.**

3.6 Observations:

```
#include<stdio.h>

int p[10][10];

int main()

{

  int i,j,k,n,t,g;

  int m[10][10],w[10][10];

  char r;

  int path (int a,int b);

  printf("enter the number of nodes\n");

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

```
printf("enter the node connection matrix(to indicate no connection enter 100)\n" );
for(i=1;i<=n;i++)
{
        for(j=1;j<=n;j++)
        {
                scanf("%d",&w[i][j]);
                m[i][j]=w[i][j];
                p[i][j]=0;
        }
}
for(k=1;k<=n;k++)
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)
                        if(m[i][k]+m[k][j]<m[i][j])
                        {
                                m[i][j]=m[i][k]+m[k][j];
                                p[i][j]=k;
                        }
                        for(g=0;g<5;g++)
                        {
                            printf("\n enter the source and destination node\n");
                            scanf("%d%d",&i,&j);
                            printf("\nthe weight is %d",m[i][j]);
                            printf("\n the path is");
                            printf("%d---->",i);
```

```
                    path(i,j);
                    printf("%d",j);
                }
}
int path(int i,int j)
{
  int k;
  k=p[i][j];
  if(k!=0)
  {
        path(i,k);
        printf("%d-->",k);
        path(k,j);
  }
}
```



Enter the number of nodes 5

Enter the node connection matrix (to indicate no connection enter 100)

0	2	1	100	100
2	0	4	100	100
1	4	0	3	5
100	100	3	0	2
100	100	5	2	0

Enter the source and destination node

1 5

The weight is 6

The path is **1----->3------->5**

3.7 Results :

3.8 Discussions:

3.9 Pre – Experimentation Questions:

1. What are drawbacks in Dijkstra algorithm?
2. Why is Dijkstra algorithm important?
3. What is the time complexity of Dijkstra algorithm?
4. What are the design issue of layers?
5. What are the protocols in application layer ?

3.10 Post – Experimentation Questions:

1. Explain Dijkstra algorithm?
2. Differentiate Dijkstra algorithm and distance vector algorithm?
3. How routers update distances to each of its neighbor?
4. How do you overcome count to infinity problem?
5. What is Dynamic routing?

EXPERIMENT No. 04:
For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the program for the cases
1. without error
2. with error

4.1 Objectives:

For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the program for the cases

1. without error
2. with error

4.2 Apparatus Required:

NCTUns 5.0.

4.3 Pre-Requisite:

Error detection techniques**,** algebraic polynomial

4.4 Introduction:

Cyclic redundancy check (CRC) is a type of function that takes a data stream of any length as input, and produces an output. Bit strings are treated as representation of polynomials with coefficients of "0" and "1". The Sender and Receiver must agree upon the Generator Polynomial in advance. Both the high and low order of the generator must be "1". The size of data must be greater than the size of the Generator polynomial to compute the checksum. The computed checksum is appended to the transmitting frame. If the receiver gets the frame it tries dividing it by Generator polynomial, if there is a reminder there has been a transmission error, else no error.

4.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

4.6 Observations:

```
#include<stdio.h>
#include<string.h>
#define N strlen(g)
char t[28],cs[28],g[]="10001000000100001";
int a,i,j;
void xor()
{
for(j = 1;j < N; j++)
cs[j] = (( cs[j] == g[j])?'0':'1');
}
void crc()
{
for(i=0;i<N;i++)
cs[i]=t[i];
do
{
if(cs[0]=='1')
xor();
for(j=0;j<N-1;j++)
cs[j]=cs[j+1];
```

```
cs[j]=t[i++];
}while(i<=a+N-1);
}
int main()
{
printf("\nEnter data : ");
scanf("%s",t);
printf("\n---------------------------------------");
printf("\nGeneratng polynomial : %s",g);
a=strlen(t);
for(i=a;i<a+N-1;i++)
t[i]='0';
printf("\n---------------------------------------");
printf("\nModified data is : %s",t);
printf("\n---------------------------------------");
crc();
printf("\nChecksum is : %s",cs);
for(i=a;i<a+N-1;i++)
t[i]=cs[i-a];
printf("\n---------------------------------------");
printf("\nFinal codeword is : %s",t);
printf("\n---------------------------------------");
printf("\nEnter received message ");
scanf("%s",t);
crc();
for(i=0;(i<N-1) && (cs[i]!='1');i++);
if(i<N-1)
printf("\nError detected\n\n");
else
printf("\nNo error detected\n\n");
printf("\n---------------------------------------\n");
return 0;
}
```

Enter data : 1011101
Generating polynomial : 10001000000100001
Modified data is : 10111010000000000000000
Checksum is : 1000101101011000
Final codeword is : 10111011000101101011000
Enter received message : 10111011000101101011000
No error detected

Enter data : 1011101
Generating polynomial : 10001000000100001
Modified data is : 10111010000000000000000
Checksum is : 1000101101011000
Final codeword is : 10111011000101101011000
Enter received message : 10111011000101101011001
Error detected

4.7 Results :

4.8 Discussions:

4.9 Pre – Experimentation Questions:

1. What are the types of errors?
2. What is Error Detection? What are its methods?
3. What is Redundancy?
4. What is CRC?
5. What is Checksum?

4.10 Post – Experimentation Questions:

1. What is the polynomial used in CRC-CCITT?
2. What are the other error detection algorithms?
3. What is difference between CRC and Hamming code?
4. What is odd parity and even parity?
5. Why Hamming code is called 7,4 code?

EXPERIMENT No. 05:
Implementation of Stop and Wait Protocol and Sliding Window Protocol.

5.1 Objectives:

Implementation of Stop and Wait Protocol and Sliding Window Protocol.

5.2 Apparatus Required:

NCTUns 5.0.

5.3 Pre-Requisite:

Connection Oriented (Point to Point) Transmission, Data Link and Transport Layers

5.4 Introduction:

STOP AND WAIT PROTOCOL:

Stop and Wait is a reliable transmission flow control protocol. This protocol works only in Connection Oriented (Point to Point) Transmission. The Source node has window size of ONE. After transmission of a frame the transmitting (Source) node waits for an Acknowledgement from the destination node. If the transmitted frame reaches the destination without error, the destination transmits a positive acknowledgement. If the transmitted frame reaches the Destination with error, the receiver destination does not transmit an acknowledgement. If the transmitter receives a positive acknowledgement it transmits the next frame if any. Else if its acknowledgement receive timer expires, it retransmits the same frame.

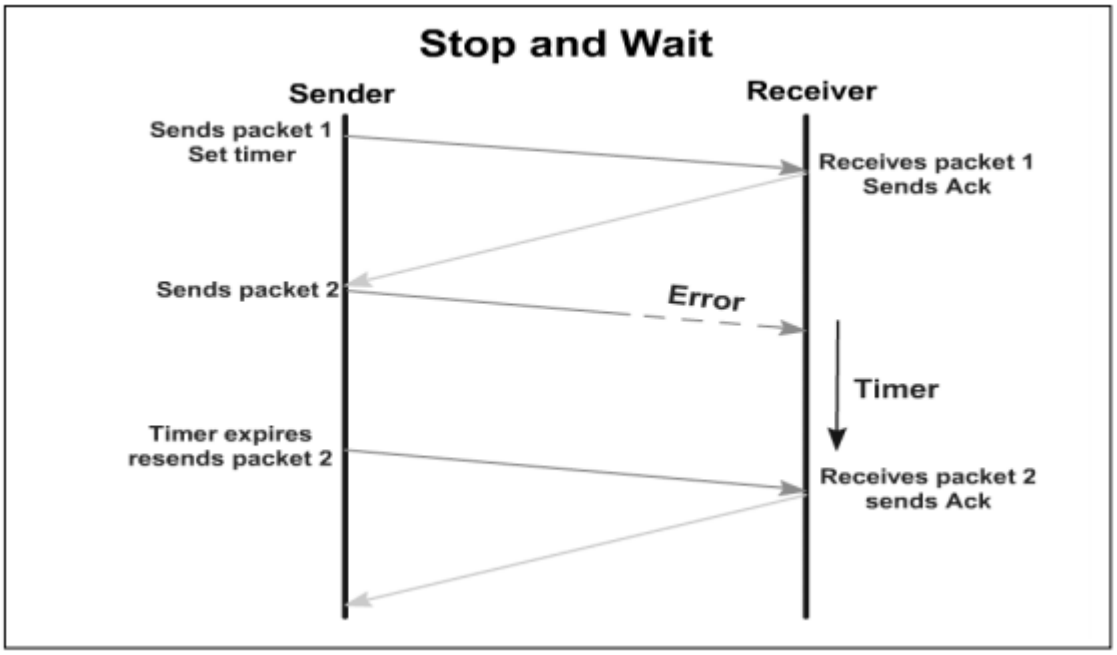
1. Start with the window size of 1 from the transmitting (Source) node
2. After transmission of a frame the transmitting (Source) node waits for a reply (Acknowledgement) from the receiving (Destination) node.
3. If the transmitted frame reaches the receiver (Destination) without error, the receiver (Destination) transmits a Positive Acknowledgement.
4. If the transmitted frame reaches the receiver (Destination) with error, the receiver (Destination) do not transmit acknowledgement.

5. If the transmitter receives a positive acknowledgement it transmits the next frame if any. Else if the transmission timer expires, it retransmits the same frame again.

6. If the transmitted acknowledgment reaches the Transmitter (Destination) without error, the Transmitter (Destination) transmits the next frame if any.

7. If the transmitted frame reaches the Transmitter (Destination) with error, the Transmitter (Destination) transmits the same frame.

8. This concept of the Transmitting (Source) node waiting after transmission for a reply from the receiver is known as STOP and WAIT.



SLIDING WINDOW PROTOCOL:

A sliding window protocol is a feature of packet-based data transmission protocols. Sliding window protocols are used where reliable in-order delivery of packets is required, such as in the Data Link Layer (OSI model) as well as in the Transmission Control Protocol (TCP).

Conceptually, each portion of the transmission (packets in most data link layers, but bytes in TCP) is assigned a unique consecutive sequence number, and the receiver uses the numbers to place received packets in the correct order, discarding duplicate packets and identifying missing ones. The problem with this is that there is no limit on the size of the sequence number that can be required.

By placing limits on the number of packets that can be transmitted or received at any given time, a sliding window protocol allows an unlimited number of packets to be

communicated using fixed-size sequence numbers. The term "window" on the transmitter side represents the logical boundary of the total number of packets yet to be acknowledged by the receiver. The receiver informs the transmitter in each acknowledgment packet the current maximum receiver buffer size (window boundary). The TCP header uses a 16 bit field to report the receive window size to the sender. Therefore, the largest window that can be used is 216 = 64 kilobytes. In slow-start mode, the transmitter starts with low packet count and increases the number of packets in each transmission after receiving acknowledgment packets from receiver. For every ack packet received, the GRT INSTITUTE OF ENGINEERING AND TECHNOLOGY window slides by one packet (logically) to transmit one new packet. When the window threshold is reached, the transmitter sends one packet for one ack packet received. If the window limit is 10 packets then in slow start mode the transmitter may start transmitting one packet followed by two packets (before transmitting two packets, one packet ack has to be received), followed by three packets and so on until 10 packets. But after reaching 10 packets, further transmissions are restricted to one packet transmitted for one ack packet received. In a simulation this appears as if the window is moving by one packet distance for every ack packet received. On the receiver side also the window moves one packet for every packet received. The sliding window method ensures that traffic congestion on the network is avoided. The application layer will still be offering data for transmission to TCP without worrying about the network traffic congestion issues as the TCP on sender and receiver side implement sliding windows of packet buffer. The window size may vary dynamically depending on network traffic.

For the highest possible throughput, it is important that the transmitter is not forced to stop sending by the sliding window protocol earlier than one round-trip delay time (RTT). The limit on the amount of data that it can send before stopping to wait for an acknowledgment should be larger than the bandwidth-delay product of the communications link. If it is not, the protocol will limit the effective bandwidth of the link.

5.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

5.6 Observations:

Stop and Wait Protocol

```
#include <stdio.h>
#include <stdlib.h>
#define RTT 4
#define TIMEOUT 4
#define TOT_FRAMES 7
enum {NO,YES} ACK;
int main()
{
 int wait_time,i=1;
 ACK=YES;
 for(;i<=TOT_FRAMES;)
 {
  if (ACK==YES && i!=1)
    printf("\nSENDER: ACK for Frame %d Received.\n",i-1);
  printf("\nSENDER: Frame %d sent, Waiting for ACK...\n",i);
  ACK=NO;
  wait_time= rand() % 4+1;
  if (wait_time==TIMEOUT)
```

```
      printf("SENDER: ACK not received for Frame %d=>TIMEOUT Resending
Frame...",i);
    else
    {
     sleep(RTT);
     printf("\nRECEIVER: Frame %d received, ACK sent\n\n",i);
     ACK=YES;
     i++;
    }
  }
  return 0;
}
```

SENDER: Frame1 sent, waiting for ACK….

SENDER: ACK not received for Frame1=>TIMEOUT Resending Frame…

SENDER: Frame1 sent, waiting for ACK…

RECEIVER: Frame1 received, ACK sent

SENDER: ACK for Frame1 received ….

SENDER: Frame2 sent, waiting for ACK……

RECEIVER: Frame2 received, ACK sent

SENDER: ACK for Frame2 received.

SENDER: Frame3 sent, waiting for ACK…

SENDER: ACK not received for Frame3=>TIMEOUT Resending Frame…

SENDER: Frame3 sent, waiting for ACK…

RECEIVER: Frame3 received, ACK sent

SENDER: ACK for Frame3 received

SENDER: Frame4 sent, waiting for ACK…

SENDER: ACK not received for Frame4=>TIMEOUT Resending Frame…

SENDER: Frame4 sent, waiting for ACK…

RECEIVER: Frame4 received, ACK sent

SENDER: ACK for Frame4 received

SENDER: Frame5sent, waiting for ACK…

RECEIVER: Frame5 received, ACK sent

SENDER: ACK for Frame5 received

SENDER: Frame6 sent, waiting for ACK…

RECEIVER: Frame6 received, ACK sent

SENDER: ACK for Frame6 received

SENDER: Frame7 sent, waiting for ACK…

RECEIVER: Frame7 received, ACK sent

Sliding Window Protocol

```
#include <stdio.h>
#include <stdlib.h>
#define RTT 5
int main()
{
  int window_size,i,j,frames[50];
  printf("Enter window size: ");
  scanf("%d",&window_size);
  printf("\nEnter number of frames to transmit: ");
  scanf("%d",&j);
  printf("\nEnter %d frames: ",j);
  for(i=1;i<=j;i++)
    scanf("%d",&frames[i]);
  printf("\nAfter sending %d frames at each stage sender waits for ACK",window_size);
  printf("\nSending frames in the following manner....\n\n");
```

```
  for(i=1;i<=j;i++)
  {
   if(i%window_size!=0)
     printf(" %d",frames[i]);
   else
   {
     printf(" %d\n",frames[i]);
     printf("SENDER: waiting for ACK...\n\n");
     sleep(RTT/2);
     printf("RECEIVER: Frames Received, ACK Sent\n\n");
     sleep(RTT/2);
     printf("SENDER:ACK received\n");
   }
  }
  if(j%window_size!=0)
  {
   printf("\nSENDER: waiting for ACK...\n");
   sleep(RTT/2);
   printf("\nRECEIVER:Frames Received, ACK Sent\n\n");
   sleep(RTT/2);
   printf("SENDER:ACK received.");
  }
  return 0;
}
```

```
  Enter Window Size                  :      3
  Enter number of frames to transmit :      8
  Enter 8 frames                     :      0
  1
  2
  3
  4
  5
  6
  7
```

After sending 3 frames at each stage sender waits for ACK

Sending frames in the following manner…

0 1 2

SENDER: Waiting for ACK…

RECEIVER: Frames Received, ACK sent

SENDER: ACK received

3 4 5

SENDER: Waiting for ACK…

RECEIVER: Frames Received, ACK sent

SENDER: ACK received

6 7

SENDER: Waiting for ACK..

RECEIVER: Frames Received, ACK sent

5.7 Results :

5.8 Discussions:

5.9 Pre – Experimentation Questions:

1. What is ARQ?
2. What is stop and wait protocol?
3. What is stop and wait ARQ?
4. What is usage of sequence number in reliable transmission?
5. What is sliding window?

5.10 Post – Experimentation Questions:

1. Define Go-Back-N ARQ?
2. Define Selective Repeat ARQ?
3. What do you mean by pipelining, is there any pipelining in error control?
4. What do you mean by line control protocol?
5. What do you mean by Authentication protocol?

EXPERIMENT No. 06:
Write a program for congestion control using leaky bucket algorithm.

6.1 Objectives:

Write a program for congestion control using leaky bucket algorithm.

6.2 Apparatus Required:

NCTUns 5.0.

6.3 Pre-Requisite:

Congestion control, network layer, traffic shaping

6.4 Introduction:

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.

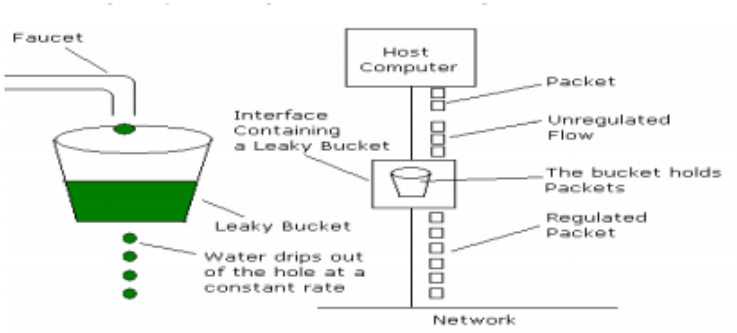
In contrast, closed loop solutions are based on the concept of a feedback loop if there is any congestion. Closed loop algorithms are also divided into two sub categories: explicit feedback and implicit feedback. In explicit feedback algorithms, packets are sent back from the point of congestion to warn 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.



6.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

6.6 Observations:

```
#include<stdio.h>
int trand(int a)
{
        int rn;
        rn=random()%10;
        rn=rn%a;
        if(rn==0)
                rn=1;
        return(rn);
}
int main()
{
        int i,packet[5],psz,bsz,pszrn=0,clk,ptime,premain,orate,flag=0;
        for(i=0;i<5;i++)
                packet[i]=trand(6)*10;
        printf("\nenter the o/p rate");
        scanf("%d",&orate);
        printf("\nenter the bucket size");
        scanf("%d",&bsz);
        for(i=0;i<5;i++)
        {
                if((packet[i]+pszrn)>bsz)
        printf("\n incoming packet size %d is greater than bucket size_reject",packet[i]);
                else
                {
                        for(;;)
                        {
                        premain=4-i;
                        psz=packet[i];
                        pszrn+=psz;
                        printf("\nincoming packet size is %d",psz);
                        printf("\n no.of packets waiting for transmission= %d",pszrn);
                        ptime=trand(4)*10;
```

```
                printf("\nnext packet will come at %d",ptime);
                for(clk=0;clk<=ptime;clk++)
                {
                    printf("\ntime left =%d",ptime-clk);
                    sleep(1);
                    if(pszrn)
                    {
                        if(pszrn==0)
                            printf("\nbucket is empty");
                        else
                        {
                            if(pszrn>=orate)

                              printf("%d bytes are tranmited",orate);
                          else
                          printf("\n %d bytes are transmitted",pszrn);
                    }
                    if(pszrn<=orate)
                        pszrn=0;
                    else
                        pszrn-=orate;
                    printf("\nbytes remaining %d ",pszrn);
                }
                else
                    printf("\nbytes remaing %d ",pszrn);
                }
                if(pszrn!=0)
                    flag=1;
                break;
               }
            }
        }
        printf("\n\n");
        return 0;
    }
```

Enter the o/p rate 10

Enter the bucket size 5

Incoming packet size 30 is greater than bucket size_reject

Incoming packet size 10 is greater than bucket size_reject

Incoming packet size 10 is greater than bucket size_reject

Incoming packet size 50 is greater than bucket size_reject

Incoming packet size 30 is greater than bucket size_reject

6.7 Results :

6.8 Discussions:

6.9 Pre – Experimentation Questions:

1. How do you classify congestion control algorithms?
2. Differentiate between Leaky bucket and Token bucket
3. How do you implement Leaky bucket?
4. How do you generate busty traffic?
5. What is Firewalls?

6.10 Post – Experimentation Questions:

1. What is token bucket algorithm?
2. What is Leaky bucket algorithm?
3. How do you implement token bucket algorithm?
4. How do you generate busty traffic?
5. What is Firewalls?

EXPERIMENT No. 07:
Write a program for a HLDC frame to perform the Bit Unstuffing

7.1 Objectives:

Write a program for a HLDC frame to perform the Bit Unstuffing.

7.2 Apparatus Required:

NCTUns 5.0.

7.3 Pre-Requisite:

Data link layer, Connection oriented connection.

7.4 Introduction:

HDLC is based on IBM's SDLC protocol, which is the layer 2 protocol for IBM's Systems Network Architecture (SNA). It was extended and standardized by the ITU as LAP (Link Access Procedure), while ANSI named their essentially identical version ADCCP.

The HDLC specification does not specify the full semantics of the frame fields. This allows other fully compliant standards to be derived from it, and derivatives have since appeared in innumerable standards. It was adopted into the X.25 protocol stack as LAPB, into the V.42 protocol as LAPM, into the Frame Relay protocol stack as LAPF and into the ISDN protocol stack as LAPD.

The original ISO standards for HDLC are the following:

ISO 3309-1979 – Frame Structure

ISO 4335-1979 – Elements of Procedure

ISO 6159-1980 – Unbalanced Classes of Procedure

ISO 6256-1981 – Balanced Classes of Procedure

ISO/IEC 13239:2002, the current standard, replaced all of these specifications.

HDLC was the inspiration for the IEEE 802.2 LLC protocol, and it is the basis for the framing mechanism used with the PPP on synchronous lines, as used by many servers to connect to a WAN, most commonly the Internet.

A similar version is used as the control channel for E-carrier (E1) and SONET multichannel telephone lines. Cisco HDLC uses low-level HDLC framing techniques but adds a protocol field to the standard HDLC header.

7.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

7.6 Observations:

```
#include<stdio.h>
int si,di;
char src[50],dst[50];
int main()
{
  printf("enter input frame(0's & 1's only):\n");
        scanf("%s",&src);
  while(src[si]!='\0')
  {
        if(src[si]=='0' && src[si+1]=='1' && src[si+2]=='1' && src[si+3]=='1' &&
        src[si+4]=='1' && src[si+5]=='1' && src[si+6]=='1' && src[si+7]=='0')
        {
          si+=8;
        }
        else if(src[si]=='1' && src[si+1]=='1' && src[si+2]=='1' && src[si+3]=='1'
        && src[si+4]=='1' && src[si+5]=='0')
```

```
            {
              dst[di]='1',dst[di+1]='1',dst[di+2]='1',dst[di+3]='1',dst[di+4]='1';
                    si+=6;
                    di+=5;
            }
            else
               dst[di++]=src[si++];
    }
    printf("After unstuffed the frame is %s\n",dst);
    return 0;
  }
```

```
Enter input frame:
011110
1       1       0
2       2       01
3       3       011
4       4       0111
5       5       01111
6       6       011110
After unstuffed the frame is 011110
```

7.7 Results :

7.8 Discussions:

7.9 Pre – Experimentation Questions:

1. Define TCP?
2. Define UDP?
3. Define IP?
4. What do you mean by client server model?
5. What is the information that a computer attached to a TCP/IP internet must possesses?

7.10 Post – Experimentation Questions:

1. Differentiate between TCP and UDP?
2. Differentiate between bit unstuffing ?
3. Application of bit unstuffing?
4. Advantages of character unstuffing?
5. Differentiate between Point-to-Point Connection and End-to-End connections.

EXPERIMENT No. 08:
Write a program for a HLDC frame to perform the Character Unstuffing

8.1 Objectives:

Write a program for a HLDC frame to perform the Character Unstuffing.

8.2 Apparatus Required:

NCTUns 5.0.

8.3 Pre-Requisite:

Data link layer, Connection oriented connection.

8.4 Introduction:

HDLC is based on IBM's SDLC protocol, which is the layer 2 protocol for IBM's Systems Network Architecture (SNA). It was extended and standardized by the ITU as LAP (Link Access Procedure), while ANSI named their essentially identical version ADCCP.

The HDLC specification does not specify the full semantics of the frame fields. This allows other fully compliant standards to be derived from it, and derivatives have since appeared in innumerable standards. It was adopted into the X.25 protocol stack as LAPB, into the V.42 protocol as LAPM, into the Frame Relay protocol stack as LAPF and into the ISDN protocol stack as LAPD.

The original ISO standards for HDLC are the following:

ISO 3309-1979 – Frame Structure

ISO 4335-1979 – Elements of Procedure

ISO 6159-1980 – Unbalanced Classes of Procedure

ISO 6256-1981 – Balanced Classes of Procedure

ISO/IEC 13239:2002, the current standard, replaced all of these specifications.

HDLC was the inspiration for the IEEE 802.2 LLC protocol, and it is the basis for the framing mechanism used with the PPP on synchronous lines, as used by many servers to connect to a WAN, most commonly the Internet.

A similar version is used as the control channel for E-carrier (E1) and SONET multichannel telephone lines. Cisco HDLC uses low-level HDLC framing techniques but adds a protocol field to the standard HDLC header.

8.5 Procedure:

To write C Program:

Open a terminal and enter the command **# GEDIT**

Type the C program and save the program with **filename.c (.c extension is necessary)**

Close the Gedit.

To Compile the program:

Go to terminal and type the command **# gcc filename.c**

To Check the output:

By default output will be stored in **"a.out" file.**

In terminal type the command **# ./a.out.**

8.6 Observations:

```
#include<stdio.h>
char src[50],dst[50];
int si,di;
int main()
{
  printf("enter input frame:\n");
  scanf("%s",&src);
  while(src[si]!='\0')
  {
        if(src[si]=='D' && src[si+1]=='L' && src[si+2]=='E' && src[si+3]=='S' &&
src[si+4]=='T' && src[si+5]=='X' )
        {
          si=si+6;
        }
        else if(src[si]=='D' && src[si+1]=='L' && src[si+2]=='E' && src[si+3]=='E' &&
src[si+4]=='T' && src[si+5]=='X' )
```

```
            {
               si=si+6;
            }
            else if(src[si]=='D' && src[si+1]=='L' && src[si+2]=='E' && src[si+3]=='D' &&
src[si+4]=='L' && src[si+5]=='E' )
            {
               si=si+6;
                           dst[di]='D',dst[di+1]='L' , dst[di+2]='E';
                           di=di+3;
            }
            else
            {
               dst[di++]=src[si++];
            }
            printf("%d\t%d\t%s\n",si,di,dst);
   }
   printf("After unstuffed the frame is %s\n",dst);
   return 0;
}
```

```
Enter input frame:
sahyadri
1       1       s
2       2       sa
3       3       sah
4       4       sahy
5       5       sahya
6       6       sahyad
7       7       sahyadr
8       8       sahyadri
After unstuffed the frame is sahyadri
```

8.7 Results :

8.8 Discussions:

8.9 Pre – Experimentation Questions:

1. Define TCP?
2. Define UDP?
3. Define IP?
4. What do you mean by client server model?
5. What is the information that a computer attached to a TCP/IP internet must possesses?

8.10 Post – Experimentation Questions:

1. Differentiate between TCP and UDP?
2. Differentiate between character unstuffing?
3. Application of bit unstuffing?
4. Advantages of character unstuffing?
5. Differentiate between Point-to-Point Connection and End-to-End connections.

Program Outcomes (POs) and Program Specific Outcomes (PSOs)

PO1. **Engineering Knowledge**: Apply knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2. **Problem Analysis**: Identify, formulate, research literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3. **Design/ Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, cultural, societal and environmental considerations.

PO4. **Conduct Investigations** of complex problems using research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of information to provide valid conclusions.

PO5. **Modern Tool Usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. **The Engineer and Society**: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to professional engineering practice.

PO7. **Environment and Sustainability**: Understand the impact of professional engineering solutions in societal and environmental contexts and demonstrate knowledge of and need for sustainable development.

PO8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams and in multi-disciplinary settings.

PO10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11. **Project Management and Finance**: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. **Life-long Learning**: Recognize the need for and have the preparation and ability to Engage in independent and lifelong learning in the broadest contest of technological change.

PSO1: Exhibit competency in Embedded Systems and VLSI Design

PSO2: Capability to comprehend the technological advancements in Signal Processing and Communication.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SAHYADRI

College of Engineering & Management

Adyar, Mangaluru - 575007