BANGALORE INSTITUTE OF TECHNOLOGY

K R Road, V V Pura, Bangalore-560004

DEPARTMENT OF
ELECTRONICS & COMMUNICATION ENGINEERING

COMPUTER NETWORKS LABORATORY MANUAL
18ECL76

VII SEMESTER

Faculty in-charge:

Dr. M N Sreerangaraju, Professor
Prof. Gahan A V, Assistant Professor
Prof. Hithaishi P, Assistant Professor
Prof. Manasa P, Assistant Professor

BANGALORE INSTITUTE OF TECHNOLOGY

K R Road, V V Pura, Bangalore-560004

DEPARTMENT OF ECE

Name of the Laboratory	**:**	**COMPUTER NETWORKS LAB**
Semester	**:**	**VII**
No. of Students / Batch	**:**	**20**
No. of Systems	**:**	**20**
Operating System	**:**	**Fedora**
Software	**:**	**NS-2, C Compiler**
Location	**:**	**4th Floor, 409B**
Faculty Lab in Charge	**:**	**Dr. M N Sreerangaraju,** Professor **Prof. Gahan A V,** Assistant Professor **Prof. Hithaishi P,** Assistant Professor **Prof. Manasa P,** Assistant Professor
Technical Staff	**:**	**Rakshitha K C,** Assistant Instructor **Suresh B M,** Assistant Instructor **Ganesh Gowda J H**, Mechanic **Dhali Gowda C,** Helper

BANGALORE INSTITUTE OF TECHNOLOGY

VISION OF THE INSTITUTE

To establish and develop the Institute as a centre of higher learning, ever abreast with expanding horizon of knowledge in the field of engineering and technology, with entrepreneurial thinking, leadership excellence for life-long success and solve societal problem.

MISSION OF THE INSTITUTE

1. Provide high quality education in the engineering disciplines from the undergraduate through doctoral levels with creative academic and professional programs.

2. Develop the Institute as a leader in Science, Engineering, Technology and management, Research and apply knowledge for the benefit of society.

3. Establish mutual beneficial partnerships with industry, alumni, local, state and central governments by public service assistance and collaborative research.

4. Inculcate personality development through sports, cultural and extracurricular activities and engage in the social, economic and professional challenges.

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

VISION

Imparting Quality Education to achieve Academic Excellence in Electronics and Communication Engineering for Global Competent Engineers.

MISSION

1. Create state of art infrastructure for quality education.
2. Nurture innovative concepts and problem solving skills.
3. Delivering Professional Engineers to meet the societal needs

PROGRAM EDUCATIONAL OBJECTIVES

PE01: Prepare graduates to be professionals, Practicing engineers an Entrepreneurs in the field of Electronics and communication Engineering.

PE02: To acquire sufficient knowledge base for innovative techniques in design and development of systems and tools.

PE03: Capable of competing globally in multidisciplinary field.

PE04: Achieve professional and personal success with awareness and commitment to ethical and social responsibilities as an individual as well as a team.

PE05: Graduates will maintain and improve technical competence through continuous learning process

PROGRAM SPECIFIC OUTCOMES

PSO1: The graduates will be able to apply the principles of Electronics and Communication in core areas.

PSO2: An ability to use latest hardware and software tools in Electronics and Communication engineering.

PSO3: Preparing Graduates to satisfy industrial needs and pursue higher studies with social-awareness and universal moral values.

Program Outcomes

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

SYLLABUS

Subject Code	18ECL76	CIE Marks	40
Number of Lecture Hours/Week	01Hr Tutorial (Instructions) + 02 Hours Laboratory = 03	SEE Marks	60
RBT Levels	L1, L2, L3	Exam Hours	03

CREDITS – 02

Course Objectives: This course will enable students to:

1. Choose suitable tools to model a network and understand the protocols at various OSI reference levels.
2. Design a suitable network and simulate using a Network simulator tool.
3. Simulate the networking concepts and protocols using C/C++ programming.
4. Model the networks for different configurations and analyze the results.

Laboratory Experiments

PART-A:
Simulation experiments using NS2/ NS3/ OPNET/ NCTUNS/ NetSim/ QuaINet or any other equivalent tool.

1. 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.
2. 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.
3. Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate.
4. Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.
5. Implement ESS with transmission nodes in Wireless LAN and obtain the performance parameters.
6. Implementation of Link state routing algorithm.

PART-B:

1. Write a program for a HLDC frame to perform the following.
   i. Bit stuffing
   ii. Character stuffing.
2. Write a program for distance vector algorithm to find suitable path for transmission.
3. Implement Dijkstra's algorithm to compute the shortest routing path.
4. For the given data, use CRC-CCITT polynomial to obtain CRC-code. Verify the program for the cases.

I. Without error.
II. With error.

5. Implementation of stop and wait protocol and sliding windows protocol.
6. Write a program for CONGESTION control using leaky (Bucket algorithm)

Graduate Attributes (as per NBA)

- Engineering knowledge.
- Problem analysis.
- Design / Development of solutions.

Conduction of Practical Examination:

- All laboratory experiments are to be included for Practical examination.
- Students are allowed to pick one experiment from the lot.
- Strictly follow the instruction printed on the cover page. On the cover page answer strip breakup of the marks.
- Change of experiment is allowed only once & 15 % marks allotted to the procedure part to be made zero.

BANGALORE INSTITUTE OF TECHNOLOGY

COMPUTER NETWORKS LABORATORY

(18ECL76)

B.E, VII Semester, Electronics & Communication Engineering

[As per Choice Based Credit System (CBCS) scheme]

PART-A: SIMULATION EXPERIMENTS USING NS2

CYCLE-I

1. 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.
2. 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.
3. Implementation of link state routing algorithm.

CYCLE-II

4. Implement Ethernet LAN using n (6-10) nodes. Compare the throughput by changing the error rate and data rate.
5. Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations.
6. Implement ESS with transmission nodes in wireless LAN and obtain the Performance Parameters.

PART-B : IMPLIEMENT THE FOLLOWING IN C PROGRAMS

CYCLE-III

7. Write a C program for a HLDC frame to perform Bit stuffing
   I. Bit stuffing
   II. Character stuffing

8. For the given data, use CRC-CCITT polynomial to obtain CRC code.
   I. Without error
   II. With error
9. Implementation of stop and wait protocol and Sliding Window Protocol.

CYCLE-IV

10. Write a C program for distance vector algorithm to find suitable path for transmission
11. Implement Dijkstra's algorithm to compute the shortest routing path.
12. Write a C Program for conjunction Control using Laky bucket algorithm.

COMPUTER NETWORKS LABORATORY

(18ECL76)

B.E, VII Semester, Electronics & Communication Engineering

[As per Choice Based Credit System (CBCS) scheme]

QUESTION BANK

1. a). 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 using NS2.

   b). For the given data, use CRC-CCITT polynomial to obtain CRC code. Verify the C program for the cases a). Without error b).With error

2. a).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 betweenn1-n3.Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP using NS2.
   b).Write a C program for distance vector algorithm to find suitable path for transmission.

3. a). Implement Ethernet LAN using n (6-10)nodes. Compare the throughput by changing the error rate and data rate using NS2.
   b). Write a C program for a HLDC frame to perform Bit stuffing.

4. a). Implement Ethernet LAN using n nodes and assign multiple traffic to the nodes and obtain congestion window for different sources/ destinations using NS2.
   b). Write a C program for a HLDC frame to perform character stuffing,

5. a) Implement ESS with transmission nodes in wireless LAN and obtain the Performance Parameters using NS2.

   b) Write a C program for implementation of stop and wait protocol.

6. a). Implementation of link state routing algorithm using NS2.
   b). Write a C program for implementation of sliding window protocol.

7. a). 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 using NS2.
   b). Write a C program for implement Dijkstra's algorithm to compute the shortest routing path

8. a). 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 betweenn1-n3.Apply relevant applications over TCP and UDP agents changing the parameter and determine the number of packets sent by TCP/UDP using NS2.
   b).Write a C Program for conjunction Control using Laky bucket algorithm.

9. a). Implementation of link state routing algorithm using NS2.
   b). Write a C program for a HLDC frame to perform Bit stuffing.

10. a). 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 using NS2.
    b). Write a C program for a HLDC frame to perform character stuffing,

11. a).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 using NS2.
    b).Write a C program for implementation of stop and wait protocol.

12. a). Implementation of link state routing algorithm using NS2.
    b). Write a C program for implementation of sliding window protocol.

COMPUTER NETWORKS LABORATORY
(18ECL76)

INTERNAL ASSEMENT EVALUATION

RUBRICS

Max Marks -40

Evaluation parameter	Marks	Observation parameters	Max Marks	Justification Levels	Justification
Record	20	Design (observation book)	5	5	Complete the Aim, network diagram and code.
				2-4	Any one of the above is missing.
				1-2	More than one is missing.
		VIVA	5	3-5	More than 50% of questions answered.
				1-3	Less than 50% of questions answered.
		Conduction	5	5	Successful conduction.
		Result &Evolution (Record)	5	5	Complete with Aim, Network diagram, code, Analysis, Calculations.
				3-5	Any one of the above missing.
				1-3	More than one is missing.
Test	20	Write up	5	5	Complete with Aim, Network diagram and code.
				1-5	Incomplete write up.
		Conduction	10	10	Successful conduction.
				6-9	Moderate performance.
				1-5	Poor performance.
		VIVA	5	3-5	More than 50% of questions answered.
				1-3	Less than 50% of questions answered.

316	18ECL76	COMPUTER NETWORKS LABORATORY

EXISTING COURSE OUTCOMES

18ECL76	COMPUTER NETWORKS LABORATORY	C406.1	Design a suitable network and simulate network element with TCP/IP and analyze network performance.
		C406.2	Designs a suitable Ethernet network using IEEE 802.3 Standard and determine through input analyze the performance of wireless LAN using IEE 802.11.
		C406.3	Design a suitable network and evaluate the performance of line state routing algorithm.
		C406.4	Demonstrate the working of HDLC, CRC, and stop & wait sliding window protocol.
		C406.5	Demonstrate the working of Dijkstra's, distance vector routing & analyze the congestion leaky bucket algorithm.

CO-PO MAPPING

			PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
18ECL76	COMPUTER NETWORKS LABORATORY	CO76.1	3	3	3	3	3								3	2	2
		CO76.2	3	3	3	3	3			1					3	2	2
		CO76.3	3	3	3	3	3								2	2	1
		CO76.4	3	3	2	1									2	1	1
		CO76.5	3	3	2	1									2	1	1
AVERAGE			3	3	2.6	2.2				1					2.4	1.6	1.4

JUSTIFICATION

			Justification	Coverage of PO's	Coverage of PSO's	Taxonomy Levels	Blooms Taxonomy keywords
18ECL76	COMPUTER NETWORKS LABORATORY	CO76.1	1. Design a point to point network with four nodes. 2. Design a four node point to point network. With links and apply TCP agent. 3. Analyze network performance by Varying bandwidth, queue size, data rate.	PO1-H PO2-H PO3-H PO4-H	PSO1-H PSO2-M PSO3-M	L6 L4	Design analyze
		CO76.2	1. Design an Ethernet LAN using IEEE standards 802.3 2. Design wireless Ethernet LAN Using IEEE standard 802.11. 3. Determine throughput, Congestion window.	PO1-H PO2-H PO3-H PO4-H PO8-L	PSO1-H PSO2-M PSO3-M	L6 L5	Design Determine
		CO76.3	1. Design a network 2. Evaluate the performance of link state routing algorithm.	PO1-H PO2-H PO3-H PO4-H	PSO1-M PSO2-M PSO3-L	L6 L4	Design Evaluate
		CO76.4	1. Demonstrate working of HDLC, CRC protocol. 2. Analyze stop and wait and sliding window protocols.	PO1-H PO2-H PO3-M PO4-L	PSO1-M PSO2-L PSO3-L	L4 L2	Demonstrate Analyze
		CO76.5	1. Demonstrate Dijkstra's, distance vector routing algorithm. 2. Analyze congestion control 3. Using leaky bucket algorithm	PO1-H PO2-M PO3-M PO4-L	PSO1-M PSO2-L PSO3-L	L4 L2	Demonstrate Analyze

DO'S AND DONT'S

Follow the schedule time, late comers will not be permitted.

1. Sign the Log book available in the Lab.
2. Leave the Footwear's in the specified stand before entering Lab.
3. Keep belongings in the specified place.
4. Students are expected to come prepared for experiments &VIVA.
5. Show the completed Observation Book and submit Record to the Teacher before the Lab session begins.
6. Cycle of experiments should be followed.
7. Follow all the safety measures as suggested by the Teacher/Lab Instructor.
8. Observe the instructions given by the Teacher/ Lab Instructor and strictly follow accordingly.
9. Report to the Teacher/Lab Instructor immediately in case of any software/hardware/ Electrical failure during working. Never try to fix it manually/individually.
10. Computers and any other experimental Kits should be switched OFF and chairs should be repositioned before leaving the Lab.
11. Get the observations verified/signed by the teacher before leaving the Lab.
12. Maintain Discipline & tidiness inside the Lab. Attend the Lab in formal attire.
13. Usage of Mobile Phones, Pen Drive and Electronics Gadgets are restricted during the Lab session.

CHAPTER - 1

INTRODUCTION TO COMPUTER NETWORKS

Let's start at the very beginning,
A very nice place to start,
When you sing, you begin with A, B, C,
When you simulate, you begin with the topology
. . .

Need for Computer Networking: A Computer Network is a set of computers connected together for the purpose of sharing resources. File sharing: Networking of computers helps the network users to share data files. Hardware sharing: Users can share devices such as printers, scanners, CD-ROM drives, hard drives etc.

A Computer Network comprises two or more computers that are connected—either by cables (wired) or WiFi (wireless)—with the purpose of transmitting, exchanging, or sharing data and resources.

It acts as basis of communication in Information Technology (IT). It is system of connected computing devices and shares information and resources between them. The devices in network are connected by communication links (wired/wireless) and share data by Data Communication System

The computers use common communication protocols over digital interconnections to communicate with each other. These interconnections are made up of telecommunication network technologies, based on physically wired, optical, and wireless radio-frequency methods that may be arranged in a variety of network topologies.

The nodes of a computer network may include personal computers, servers, networking hardware, or other specialised or general-purpose hosts. They are identified by hostnames and network addresses. Hostnames serve as memorable labels for the nodes, rarely changed after initial assignment. Network addresses serve for locating and identifying the nodes by communication protocols such as the Internet Protocol.

Computer networks may be classified by many criteria, including the transmission medium used to carry signals, bandwidth, communications protocols to organize network traffic, the network size, the topology, traffic control mechanism, and organizational intent.

Computer networks support many applications and services, such as access to the World Wide Web, digital video, digital audio, shared use of application and storage servers, printers, and fax machines, and use of email and instant messaging applications.

A computer network extends interpersonal communications by electronic means with various technologies, such as email, instant messaging, online chat, voice and video telephone calls, and video conferencing. A network allows sharing of network and computing resources. Users may access and use resources provided by devices on the network, such as printing a document on a shared network printer or use of a shared storage device. A network allows sharing of files, data, and other types of information giving authorized users the ability to access information stored on other computers on the network. Distributed computing uses computing resources across a network to accomplish tasks.

Network Packet

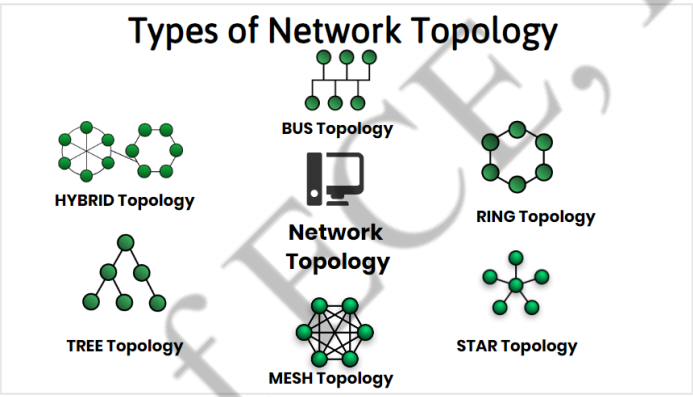
Most modern computer networks use protocols based on packet-mode transmission. A network packet is a formatted unit of data carried by a packet-switched network. The physical link technologies of packet network typically limit the size of packets to a certain maximum transmission unit (MTU). A longer message is fragmented before it is transferred and once the packets arrive, they are reassembled to construct the original message.

Packets consist of two types of data: control information and user data (payload). The control information provides data the network needs to deliver the user data, for example, source and destination network addresses, error detection codes, and sequencing information. Typically, control information is found in packet headers and trailers, with payload data in between.

With packets, the bandwidth of the transmission medium can be better shared among users than if the network were circuit switched. When one user is not sending packets, the link can be filled with packets from other users, and so the cost can be shared, with relatively little interference, provided the link isn't overused. Often the route a packet needs to take through a network is not immediately available. In that case, the packet is queued and waits until a link is free.

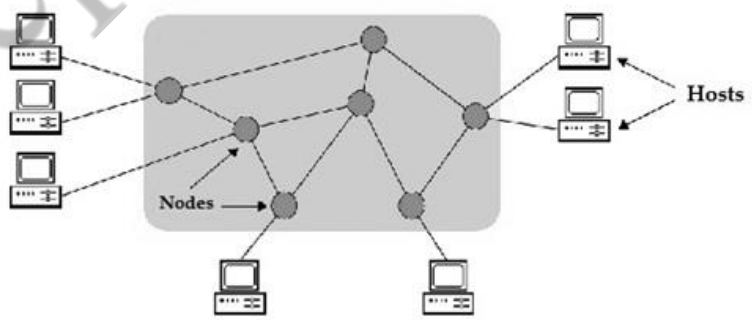
Network Topology

Network topology is the layout, pattern, or organizational hierarchy of the interconnection of network hosts, in contrast to their physical or geographic location. Typically, most diagrams describing networks are arranged by their topology. The network topology can affect throughput, but reliability is often more critical. With many technologies, such as bus or star networks, a single failure can cause the network to fail entirely. In general, the more interconnections there are, the more robust the network is; but the more expensive it is to install.



Network Nodes

Apart from any physical transmission media, networks are built from additional basic system building blocks, network interface controllers (NICs), repeaters, hubs, bridges, switches, routers, modems, and firewalls. Any particular piece of equipment will frequently contain multiple building blocks and so may perform multiple functions.



Repeaters and hubs

A repeater is an electronic device that receives a network signal, cleans it of unnecessary noise and regenerates it. The signal is retransmitted at a higher power level, or to the other side of obstruction so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable that runs longer than 100 meters. With fiber optics, repeaters can be tens or even hundreds of kilometres apart.

Repeaters work on the physical layer of the OSI model but still require a small amount of time to regenerate the signal. This can cause a propagation delay that affects network performance and may affect proper function. As a result, many network architectures limit the number of repeaters used in a network.

An Ethernet repeater with multiple ports is known as an Ethernet hub. In addition to reconditioning and distributing network signals, a repeater hub assists with collision detection and fault isolation for the network. Hubs and repeaters in LANs have been largely obsoleted by modern network switches.

Bridges and switches

Network bridges and network switches are distinct from a hub in that they only forward frames to the ports involved in the communication whereas a hub forwards to all ports. Bridges only have two ports but a switch can be thought of as a multi-port bridge. Switches normally have numerous ports, facilitating a star topology for devices, and for cascading additional switches.

Bridges and switches operate at the data link layer (layer 2) of the OSI model and bridge traffic between two or more network segments to form a single local network. Both are devices that forward frames of data between ports based on the destination MAC address in each frame. They learn the association of physical ports to MAC addresses by examining the source addresses of received frames and only forward the frame when necessary. If an unknown destination MAC is targeted, the device broadcasts the request to all ports except the source, and discovers the location from the reply.

Bridges and switches divide the network's collision domain but maintain a single broadcast domain. Network segmentation through bridging and switching helps break down a large, congested network into an aggregation of smaller, more efficient networks.

Routers

A router is an internetworking device that forwards packets between networks by processing the addressing or routing information included in the packet. The routing information is often processed in conjunction with the routing table. A router uses its routing table to determine where to forward packets and does not require broadcasting packets which is inefficient for very big networks.

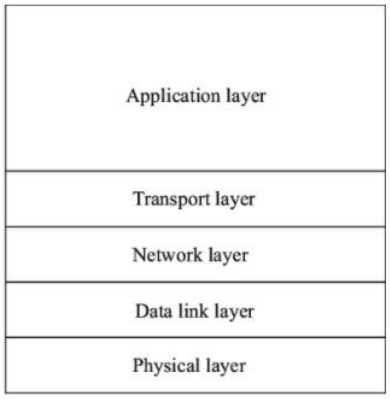
Modems

Modems (modulator-demodulator) are used to connect network nodes via wire not originally designed for digital network traffic, or for wireless. To do this one or more carrier signals are modulated by the digital signal to produce an analog signal that can be tailored to give the required properties for transmission. Early modems modulated audio signals sent over a standard voice telephone line. Modems are still commonly used for telephone lines, using a digital subscriber line technology and cable television systems using DOCSIS technology.

Firewalls

A firewall is a network device or software for controlling network security and access rules. Firewalls are inserted in connections between secure internal networks and potentially insecure external networks such as the Internet. Firewalls are typically configured to reject access requests from unrecognized sources while allowing actions from recognized ones. The vital role firewalls play in network security grows in parallel with the constant increase in cyber-attacks.

TCP/IP Model



The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as *TCP/IP*. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating via an IP network. Major internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP, which is part of the Transport Layer of the TCP/IP suite. SSL/TLS often runs on top of TCP. The TCP/IP model is based on a five-layer model for networking. From bottom (the link) to top (the user application), these are the physical, data link, net- work, transport, and application layers. Not all layers are completely defined by the model, so these layers are "filled in" by external standards and protocols.

Physical Layer

The physical network layer specifies the characteristics of the hardware to be used for the network. For example, physical network layer specifies the physical characteristics of the communications media. The physical layer of TCP/IP describes hardware standards such as IEEE 802.3, the specification for Ethernet network media, and RS-232, the specification for standard pin connectors.

Data-Link Layer

The data-link layer identifies the network protocol type of the packet, in this instance TCP/IP. The data-link layer also provides error control and "framing." Examples of data-link layer protocols are Ethernet IEEE 802.2 framing and Point-to-Point Protocol (PPP) framing.

Internet Layer

This layer, also known as the network layer, accepts and delivers packets for the network. This layer includes the powerful Internet Protocol (IP), the Address Resolution Protocol (ARP), and the Internet Control Message Protocol (ICMP).

IP Protocol

The IP protocol and its associated routing protocols are possibly the most significant of the entire TCP/IP suite. IP is responsible for the following:

- IP addressing – The IP addressing conventions are part of the IP protocol. Chapter 3, Planning Your TCP/IP Network (Task) describes IPv4 addressing in detail and Chapter 14, IPv6 (Overview) describes IPv6 addressing in detail.

- Host-to-host communications – IP determines the path a packet must take, based on the receiving host's IP address.

- Packet formatting – IP assembles packets into units that are known as IP datagrams. Datagrams are fully described in Internet Layer.

- Fragmentation – If a packet is too large for transmission over the network media, IP on the sending host breaks the packet into smaller fragments. IP on the receiving host then reconstructs the fragments into the original packet.

Previous releases of the Solaris operating environment implement version 4 of the Internet Protocol, which is abbreviated as IPv4. However, because of the rapid growth of the Internet, a new Internet Protocol was created. The new protocol increases address space. This new version, known as version 6, is abbreviated as IPv6. The Solaris operating environment supports both versions, which are described in this book. To avoid confusion when addressing the Internet Protocol, one of the following conventions is used:

- When the term IP is used in a description, the description applies to both IPv4 and IPv6.
- When the term IPv4 is used in a description, the description applies only to IPv4.
- When the term IPv6 is used in a description, the description applies only to IPv6.

ARP Protocol

The Address Resolution Protocol (ARP) conceptually exists between the data-link and Internet layers. ARP assists IP in directing datagrams to the appropriate receiving host by mapping Ethernet addresses (48 bits long) to known IP addresses (32 bits long).

ICMP Protocol

Internet Control Message Protocol (ICMP) detects and reports network error conditions. ICMP reports on the following:

- Dropped packets – Packets that arrive too fast to be processed
- Connectivity failure – A destination host that cannot be reached)
- Redirection – Redirecting a sending host to use another router

The ping Command contains more information on the operating system commands that use ICMP for error detection.

Transport Layer

The TCP/IP transport layer protocols ensure that packets arrive in sequence and without error, by swapping acknowledgments of data reception, and retransmitting lost packets. This type of communication is known as “end-to-end.” Transport layer protocols at this level are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

TCP Protocol

TCP enables applications to communicate with each other as though connected by a physical circuit. TCP sends data in a form that appears to be transmitted in a character-by-character fashion, rather than as discrete packets. This transmission consists of a starting point, which opens the connection, the entire transmission in byte order, and an ending point, which closes the connection.

TCP attaches a header onto the transmitted data. This header contains a large number of parameters that help processes on the sending machine connect to peer processes on the receiving machine.

TCP confirms that a packet has reached its destination by establishing an end-to-end connection between sending and receiving hosts. TCP is therefore considered a "reliable, connection-oriented" protocol.

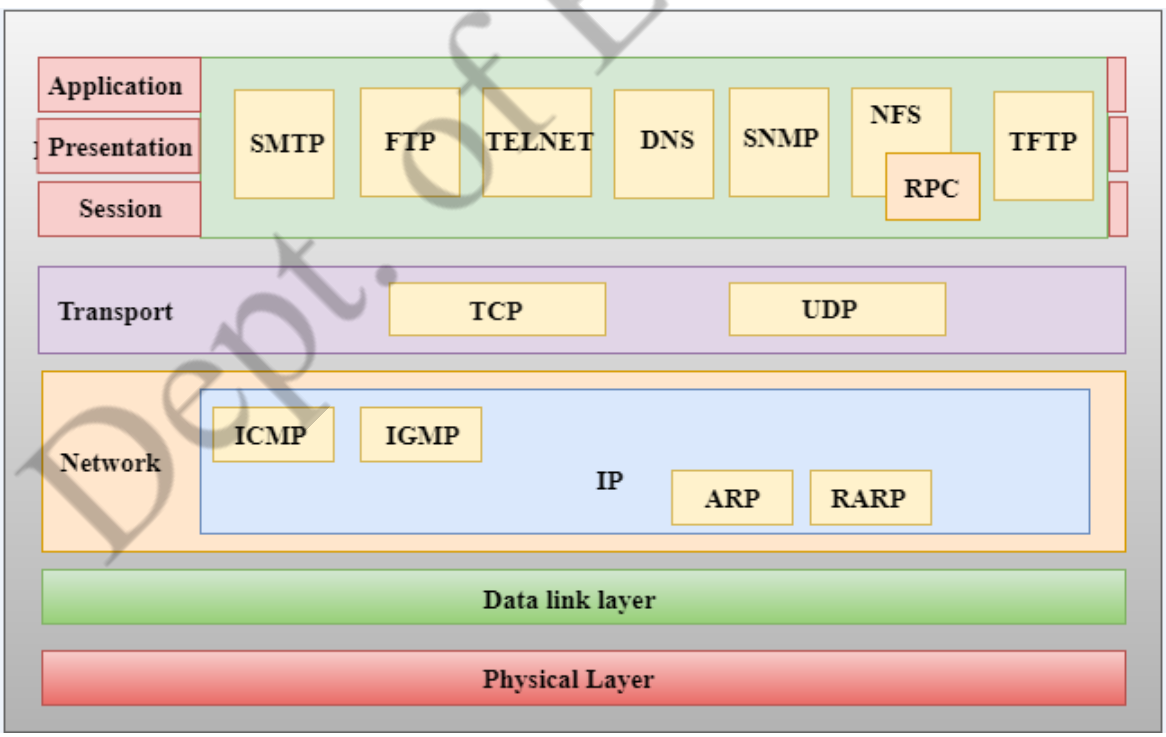
UDP Protocol

UDP, the other transport layer protocol, provides datagram delivery service. UDP does not verify connections between receiving and sending hosts. Because UDP eliminates the processes of establishing and verifying connections, applications that send small amounts of data use UDP rather than TCP.

Application Layer

The application layer defines standard Internet services and network applications that anyone can use. These services work with the transport layer to send and receive data. Many application layer protocols exist. The following list shows examples of application layer protocols:

- Standard TCP/IP services such as the ftp, tftp, and telnet commands
- UNIX "r" commands, such as rlogin and rsh
- Name services, such as NIS+ and domain name system (DNS)
- File services, such as the NFS service
- Simple Network Management Protocol (SNMP), which enables network management
- RIP and RDISC routing protocols



CHAPTER – 2

NETWORK SIMULATOR – 2

Simulation: "The process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and/or evaluating various strategies for the operation of the system."

What is the need of simulation? Network simulation offers an efficient, cost-effective way to assess how the network will behave under different operating conditions. Simulation results can be analysed to assess network performance, identify potential problems, understand the root cause, and resolve the issues prior to deployment. The main advantage of a simulator is to provide practical feedback to the users while designing real-world systems. They allow the designers of the system to study trouble at numerous abstraction levels.

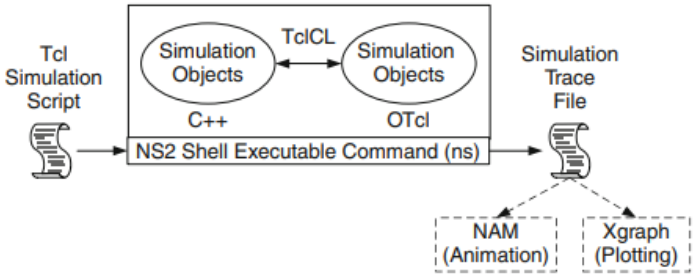
Network Simulator (Version 2), widely known as NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviours. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator -1, the foundation on which NS is invented. Since 1995 the Defence Advanced Research Projects Agency (DARPA) supported the development of NS through the Virtual Internetwork Testbed (VINT) project 10.2, currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of researchers and developers in the community are constantly working to keep NS2 strong and versatile.

Basic Architecture

The below figure shows the basic architecture of NS2. NS2 provides users with an executable command "ns" which takes one input argument, the name of a Tcl simulation scripting file. In most cases, a simulation trace file is created and is used to plot graph and/or to create animation. NS2 consists of two key languages: CCC and Object-oriented Tool Command Language (OTcl). While the CCC defines the internal mechanism (i.e., a backend) of the simulation, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The CCC and the OTcl are linked together using Tcl. Mapped to a CCC object, variables in the OTcl domains are sometimes referred to as handles. Conceptually, a handle is just a string (e.g., "_o10") in the OTcl domain and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped CCC object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects.

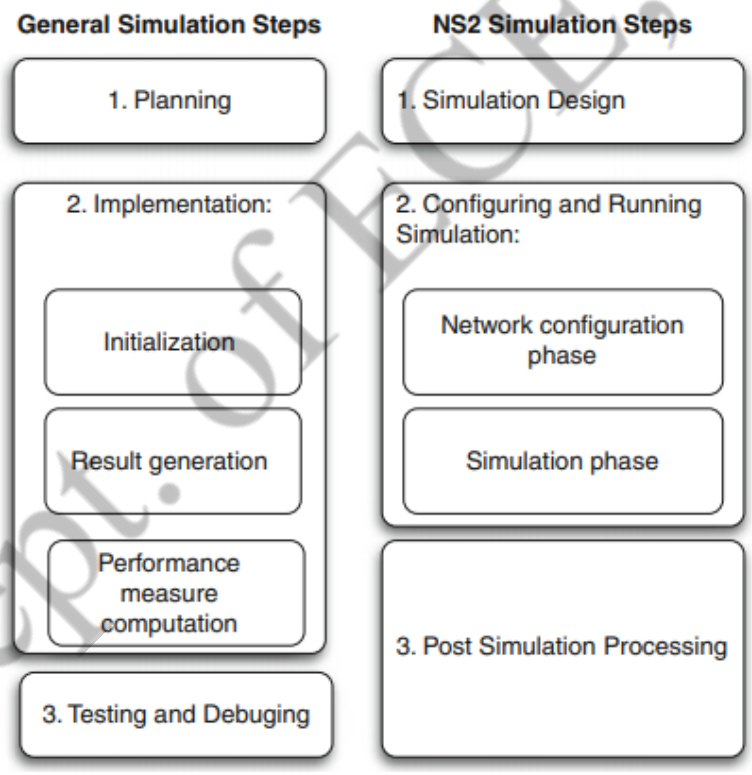
TCL: TCL stands for tool command language .TCL files are text files containing TCL scripts. TCL is an interpreted script language developed by Dr John Outshout at the University of California, Berkeley.

AWK: AWK is a scripting language suitable for text processing / data extraction and report formatting. It is interpreted line by line and produces the output in required format.



NS2 provides a large number of built-in CCC classes. It is advisable to use these CCC classes to set up a simulation via a Tcl simulation script. However, advance users may find these objects insufficient. They need to develop their own CCC classes and use a OTcl configuration interface to put together objects instantiated from these class. After simulation, NS2 outputs either text-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyse a particular behaviour of the network, users can extract a relevant subset of text-based data and transform it to a more conceivable presentation.

NS2 Simulation Steps



The key NS2 simulation steps include the following:

Step 1: Simulation Design The first step in simulating a network is to design the simulation. In this step, the users should determine the simulation purposes, network configuration.

Step 2: Configuring and Running Simulation This step implements the design in the first step. It consists of two phases:

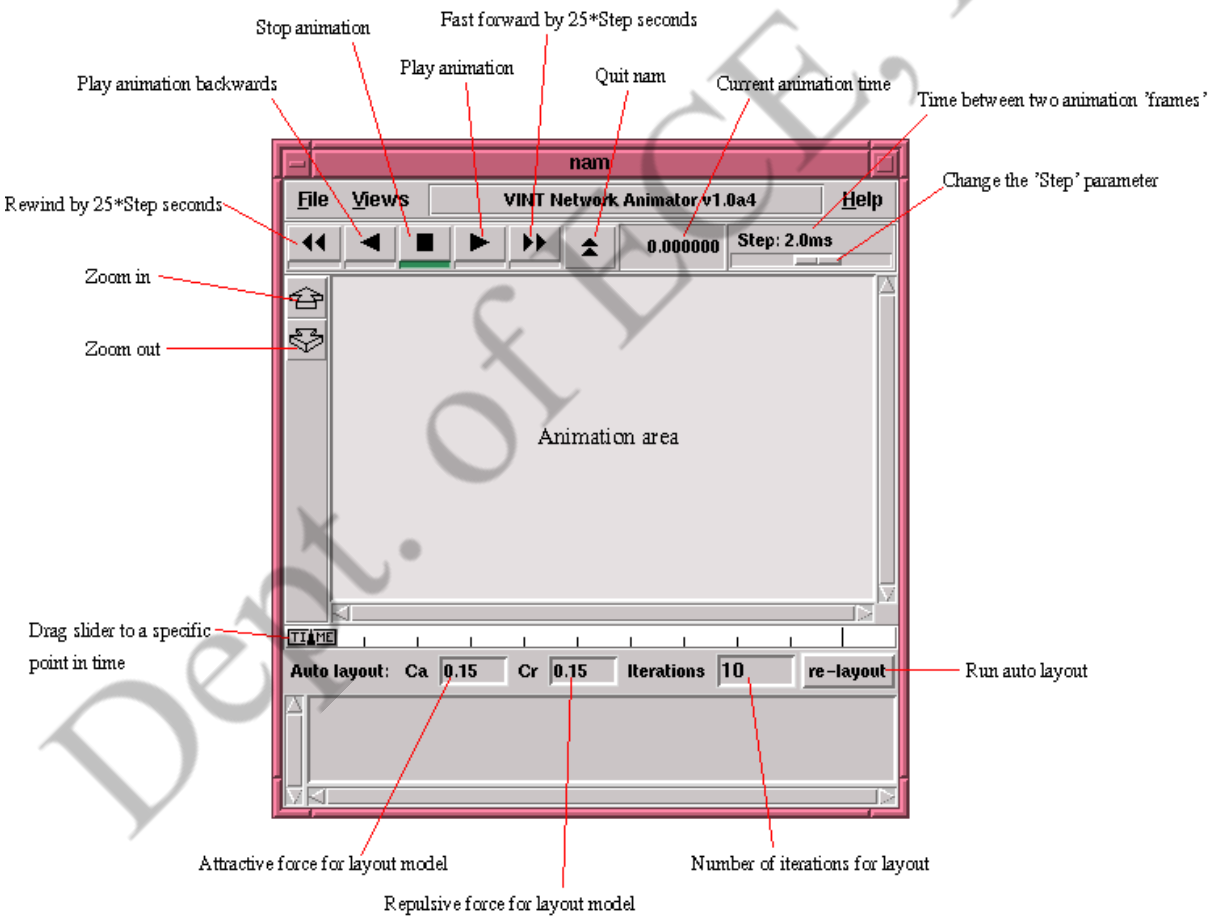
• Network configuration phase: In this phase, network components (e.g., node, TCP and UDP) are created and configured according to the simulation design. Also, the events such as data transfer are scheduled to start at a certain time.

• Simulation Phase: This phase starts the simulation which was configured in the Network Configuration Phase. It maintains the simulation clock and executes events chronologically. This phase usually runs until the simulation clock reaches a threshold value specified in the Network Configuration Phase. In most cases, it is convenient to define a simulation scenario in a Tcl scripting file and feed the file as an input argument of an NS2 invocation (e.g., executing "ns ").

Step 3: Post simulation processing the main tasks in this steps include verifying the integrity of the program and evaluating the performance of the simulated network. While the first task is referred to as debugging, the second one is achieved by properly collecting and compiling simulation results.

NAM Animator

NAM is a TCL based animation tool for viewing network simulation traces and real world packet traces. It supports topology layout, packet level animation and various data inspection tools.



Trace File

Text-based packet tracing records the detail of packets passing through network checkpoints (e.g., nodes and queues). The trace file format is shown below,

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)
+ : enqueue (at queue)                src_addr : node.port (3.0)
- : dequeue (at queue)                dst_addr : node.port (0.0)
d : drop    (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
- 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
```

In here, there is 12 fields and we can explain it as;

1. EVENT OR TYPE IDENTIFIER

+ :a packet enque event
- :a packet deque event
r :a packet reception event
d :a packet drop event
c :a packet collision at the MAC level

2. TIME: at which the packet tracing string is created.

3-4. SOURCE AND DESTINATION NODE: source and destination ID's of tracing objects.

5. PACKET NAME: Name of the packet type.

6. PACKET SIZE: Size of packet in bytes.

7. FLAGS: 7 digit flag string.

"-": disable
1st = "E": ECN (Explicit Congestion Notification) echo is enabled.
2nd = "P": the priority in the IP header is enabled.
3rd: Not in use
4th = "A": Congestion action
5th = "E": Congestion has occurred.
6th = "F": The TCP fast start is used.
7th = "N": Explicit Congestion Notification (ECN) is on.

8. FLOW ID

9. SOURCE AND DESTINATION ADDRESS: The format of these two fields is “a.b”, where “a" is the address and "b" is the port.

10. SEQUENCE NUMBER

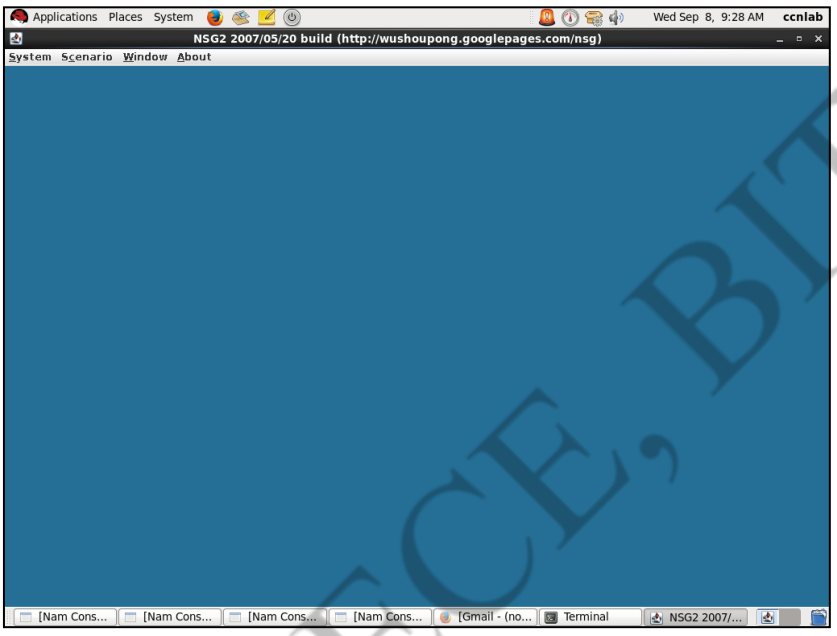
11. PACKET UNIQUE ID

Each trace line starts with an event (+, -, *d, r*) descriptor followed by the simulation time (*in seconds*) of that event, and from and to node, which identify the link on which the event occurred.

PART A

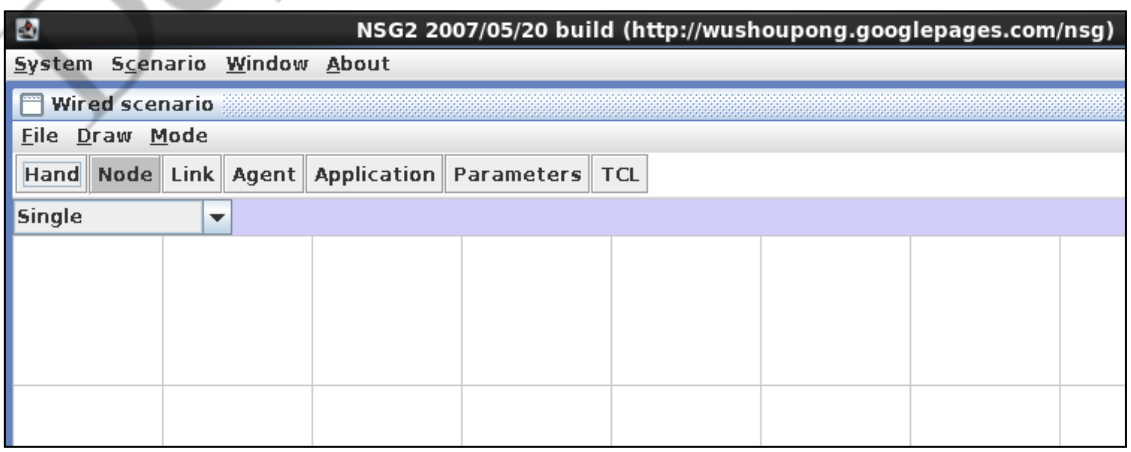
Execution steps:

1) Double Click on terminal
2) NSG2 windows opens

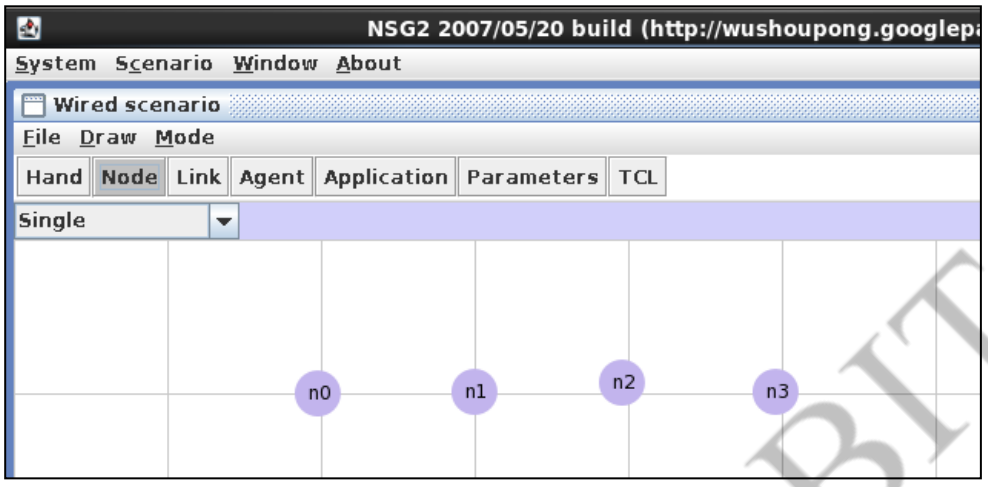


3) Click on scenario and select wired scenario

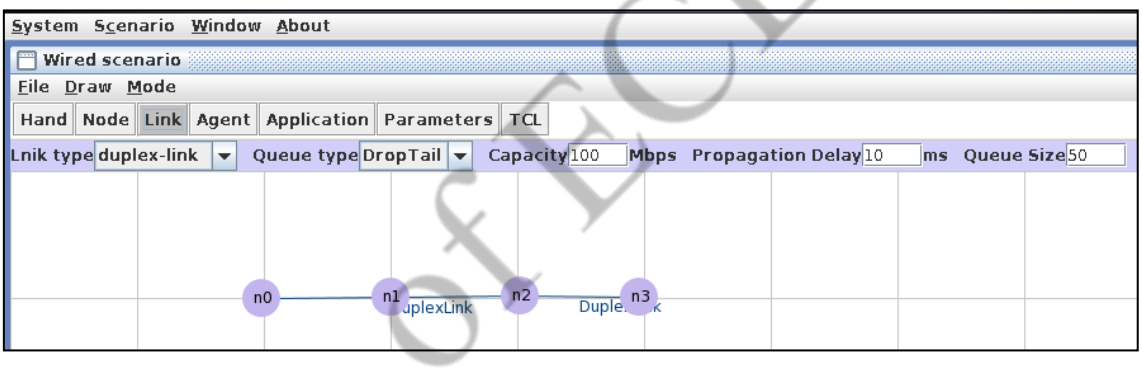
4) To draw network and place nodes, click on node



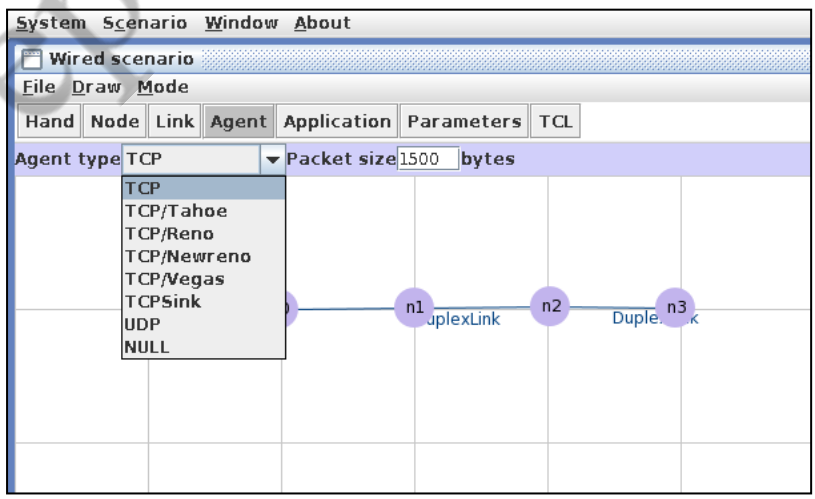
5) Place the nodes on drawing sheet



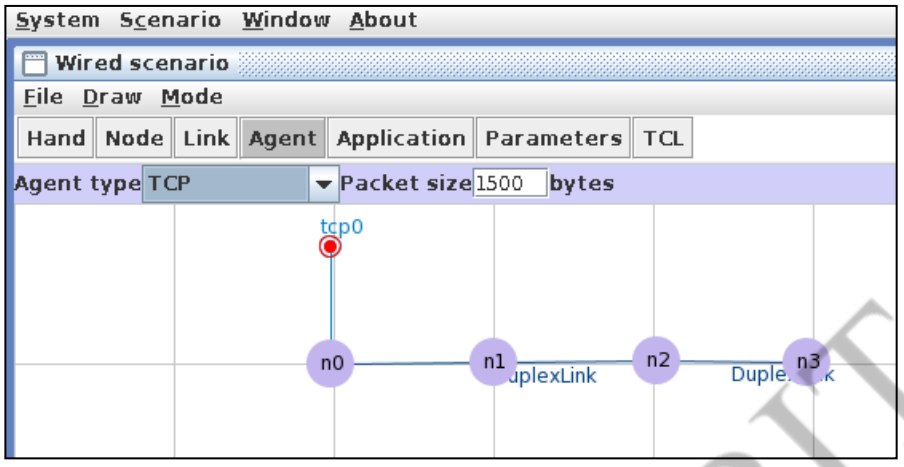
6) Click on link; define Link type, capacity, propagation delay and queue size and connect N0-N1, N1-N2, N2-N3.



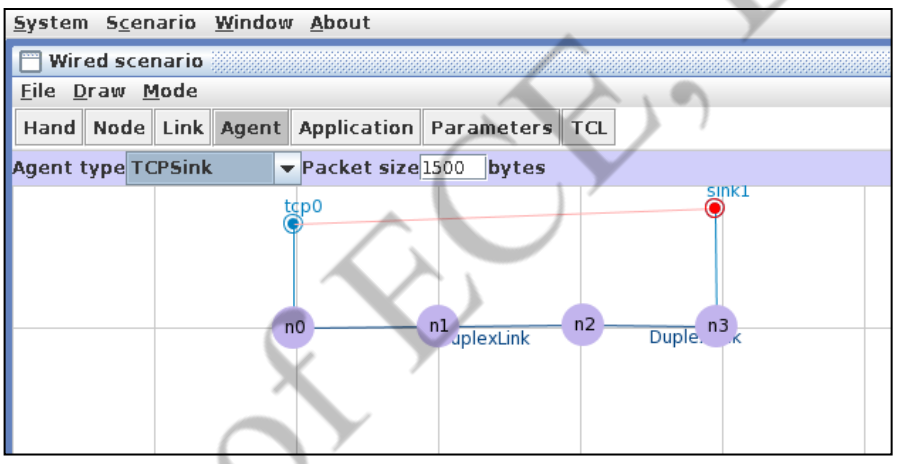
7) Click on agent; define type of agent and packet size.



Select TCP agent and connect to node N0, which is a source node

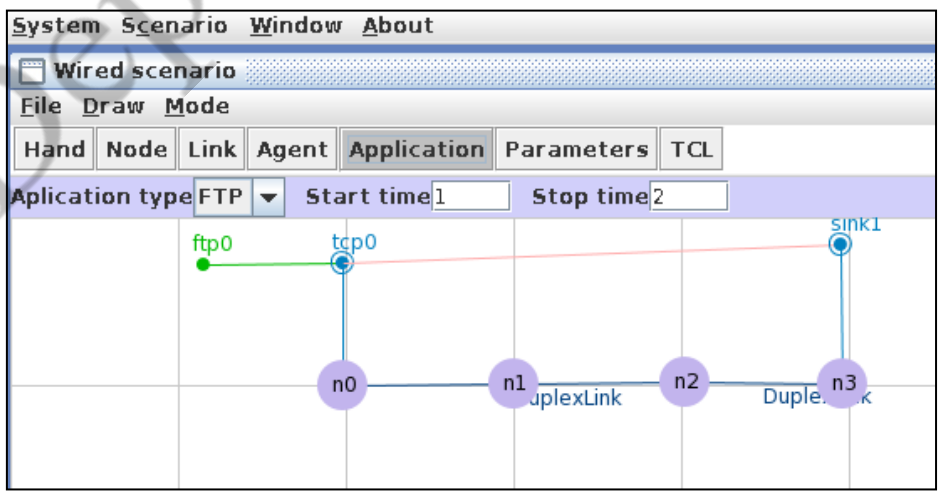


Select TCP sink and connect to node N3, which is a destination node. Connect TCP agent and TCP sink

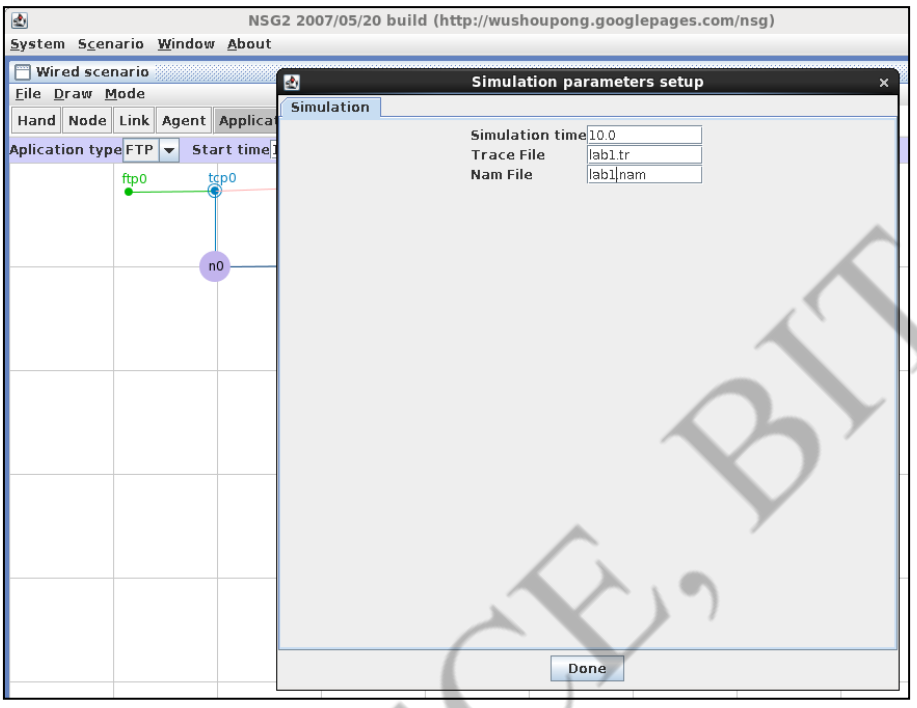


8) Click on Application; Define application type: FTP/CBR
Define start and stop time

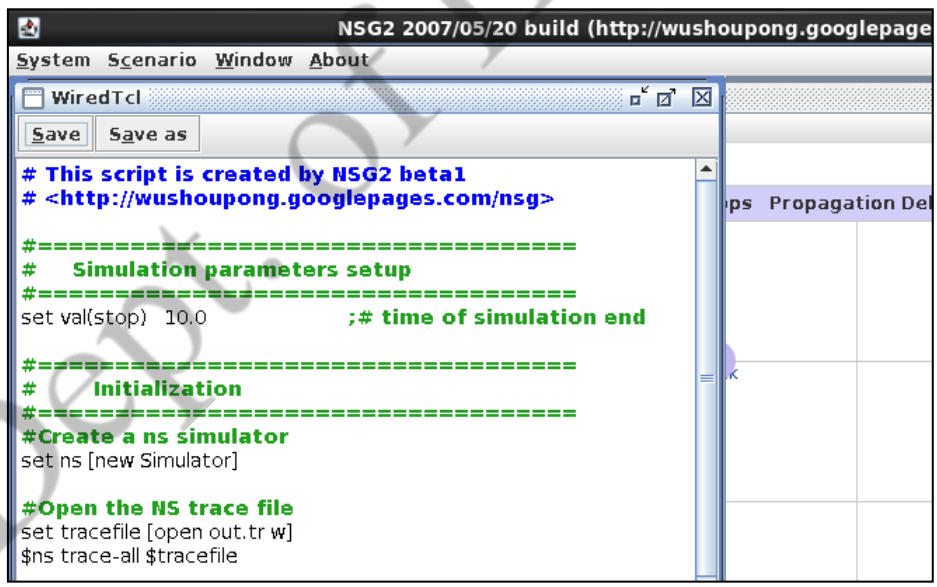
Attach ftp to TCP agent



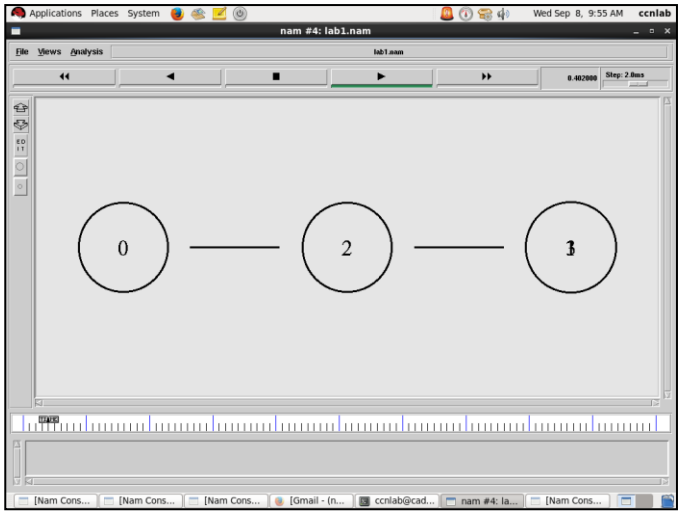
9) Click on parameters; Define the simulation time
Rename trace file with .tr extension
Rename Nam file with .nam extension



10) Click on TCL to generate TCL script, save it in appropriate location with .tcl as extension



11) Close NSG2 window, go back to terminal to create AWK file type the following command in terminal window: **gedit lab1.awk**
12) gedit window opens, type the AWK script and save file
13) To run Nam window type the following command in terminal window: **ns lab1.tcl**
14) Nam window opens, run the simulation



15) Close the Nam window after run time ends, to run AWK script type the following command in terminal window: **awk –f lab1.awk lab1.tr**
16) Observe the packets dropped and received in terminal window.

Experiment -1

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

Pre-requisites:

Nodes: A node is a connection point inside a network that can receive, send, create, or store data. Each node requires you to provide some form of identification to receive access, like an IP address. A few examples of nodes include computers, printers, modems, bridges, and switches.

Point-To-Point Network: Point-To-Point connection refers to a communications connection between two communication endpoints or nodes. An example is a telephone call, in which one telephone is connected with one other, and what is said by one caller can only be heard by the other.

Duplex Links: A duplex communication system is a point-to-point system composed of two or more connected parties or devices that can communicate with one another in both directions. In a full-duplex system, both parties can communicate with each other simultaneously.

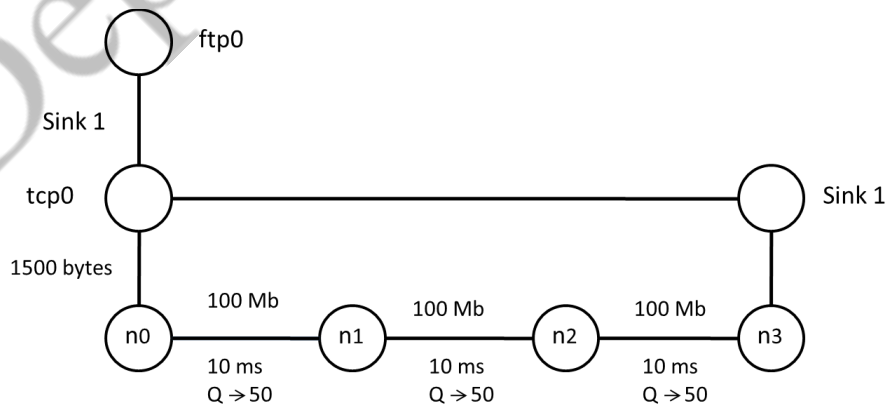
Queue: A queue, in computer networking, is a collection of data packets collectively waiting to be transmitted by a network device using a per-defined structure methodology.

Bandwidth: The maximum amount of data transmitted over an internet connection in a given amount of time.

Requirements for Network Setup:

- Four nodes
- Point to point network
- Duplex links
- TCP agent and FTP application

Network Diagram:



TCL Script:

```
#=================================
#     Simulation parameters setup
#=================================
setval(stop)   10.0                ;# time of simulation end

#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
settracefile [open lab1.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab1.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#=================================
#       Links Definition
#=================================
#Createlinks between nodes
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 50
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail   ## Vary the bandwidth for different cases ##
$ns queue-limit $n1 $n2 50                      ## Vary the queue size for different cases ##

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n2 orient right

#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500


#=================================
#        Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 8.0 "$ftp0 stop"


#=================================
#        Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab1.nam &
exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

AWK Script:

```
BEGIN{
drpd=0;
rcd=0;
}
{
if($1=="d"&&$4=="3")
drpd++;
if($1=="r"&&$4=="3")
rcd++;
}
END{
printf("\ndroppedreceievdpkts are %d\n%d\n\n",drpd,rcd);
}
```

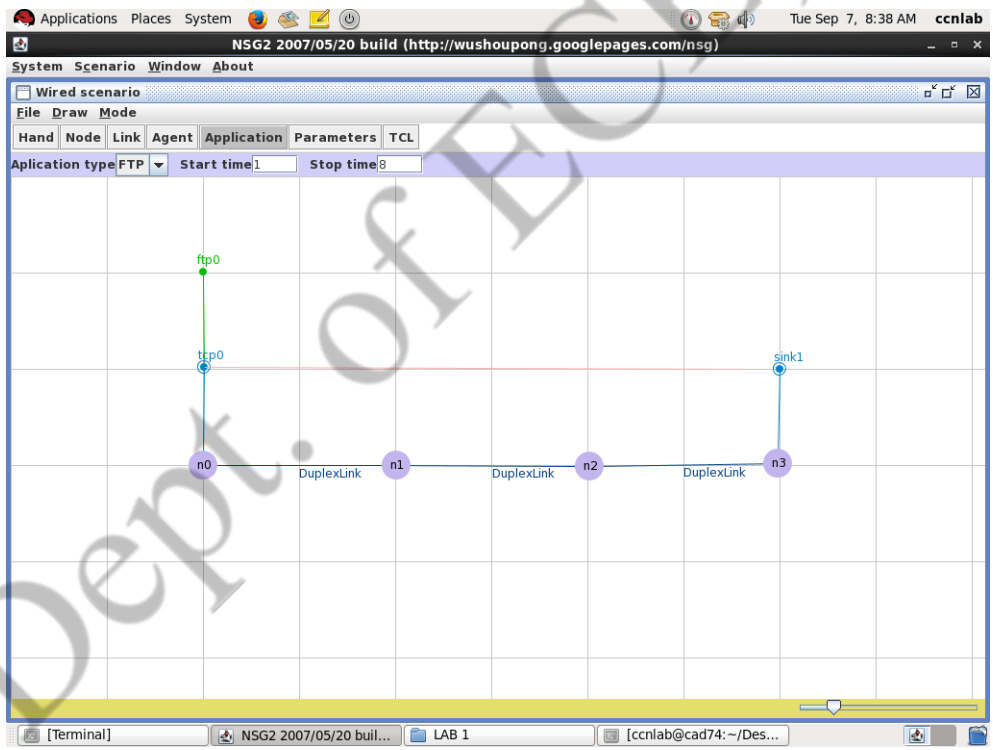
Execution Steps:

1. Create TCL code, save as eg:lab1.tcl
2. Create AWK code using the command eg: gedit lab1.awk
3. Run the network simulator in the terminal: ns lab1.tcl.
4. Observe the output in nam window and trace file.
5. Check the analysis of trace file: awk –f lab1.awk lab1.tr.
6. Vary the parameters of bandwidth, queue size in lab1.tcl.
(Open the file using command, gedit lab1.tcl)
7. Run ns again, check the output, tabulate the readings.

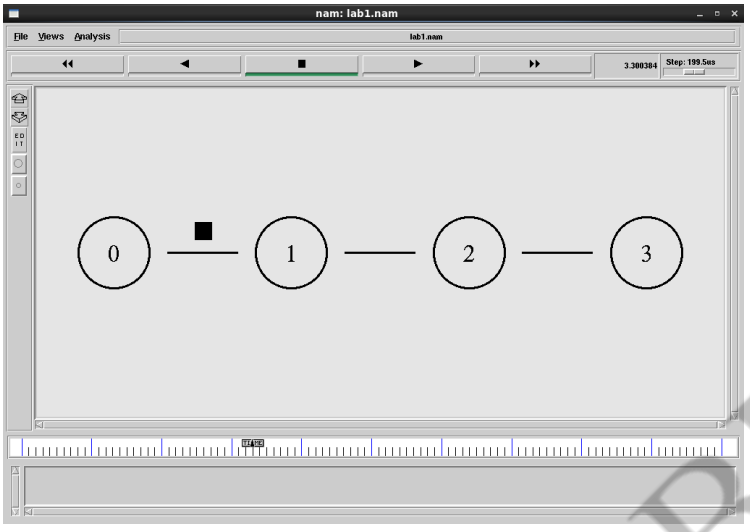
Data Inputs:

N0-N1	N1-N2	N2-N3
100Mb 10ms Q-50	100Mb 10ms Q-50	100Mb 10ms Q-50
100Mb 10ms Q-50	100Mb 10ms Q-50	50Mb 10ms Q-5
100Mb 10ms Q-50	100Mb 10ms Q-50	0.5Mb 10ms Q-5

Network Window:



NAM Window:



Trace File:

```
+ 1 0 1 tcp 40 ------- 0 0.0 3.0 0 0
- 1 0 1 tcp 40 ------- 0 0.0 3.0 0 0
r 1.010003 0 1 tcp 40 ------- 0 0.0 3.0 0 0
+ 1.010003 1 2 tcp 40 ------- 0 0.0 3.0 0 0
- 1.010003 1 2 tcp 40 ------- 0 0.0 3.0 0 0
r 1.020006 1 2 tcp 40 ------- 0 0.0 3.0 0 0
+ 1.020006 2 3 tcp 40 ------- 0 0.0 3.0 0 0
- 1.020006 2 3 tcp 40 ------- 0 0.0 3.0 0 0
r 1.03001 2 3 tcp 40 ------- 0 0.0 3.0 0 0
+ 1.03001 3 2 ack 40 ------- 0 3.0 0.0 0 1
- 1.03001 3 2 ack 40 ------- 0 3.0 0.0 0 1
r 1.040013 3 2 ack 40 ------- 0 3.0 0.0 0 1
+ 1.040013 2 1 ack 40 ------- 0 3.0 0.0 0 1
- 1.040013 2 1 ack 40 ------- 0 3.0 0.0 0 1
r 1.050016 2 1 ack 40 ------- 0 3.0 0.0 0 1
+ 1.050016 1 0 ack 40 ------- 0 3.0 0.0 0 1
- 1.050016 1 0 ack 40 ------- 0 3.0 0.0 0 1
r 1.060019 1 0 ack 40 ------- 0 3.0 0.0 0 1
+ 1.060019 0 1 tcp 1540 ------- 0 0.0 3.0 1 2
- 1.060019 0 1 tcp 1540 ------- 0 0.0 3.0 1 2
+ 1.060019 0 1 tcp 1540 ------- 0 0.0 3.0 2 3
- 1.060142 0 1 tcp 1540 ------- 0 0.0 3.0 2 3
r 1.070142 0 1 tcp 1540 ------- 0 0.0 3.0 1 2
+ 1.070142 1 2 tcp 1540 ------- 0 0.0 3.0 1 2
- 1.070142 1 2 tcp 1540 ------- 0 0.0 3.0 1 2
r 1.070266 0 1 tcp 1540 ------- 0 0.0 3.0 2 3
```

Output:

N0-N1	N1-N2	N2-N3	Packets Received	Packets dropped
100Mb 10ms Q-50	100Mb 10ms Q-50	100Mb 10ms Q-50		
100Mb 10ms Q-50	100Mb 10ms Q-50	50Mb 10ms Q-5		
100Mb 10ms Q-50	100Mb 10ms Q-50	0.5Mb 10ms Q-5		

Conclusion/Inference: (To be written by students)

Experiment -2

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

Pre-requisites:

TCP: Transmission Control Protocol is a standard that defines how to establish and maintain a network conversation through which application programs can exchange data. TCP works with the Internet Protocol (IP), which defines how computers send packets of data to each other. TCP is a protocol present in transport layer.

UDP: User Datagram Protocol a communications protocol that facilitates the exchange of messages between computing devices in a network. It's an alternative to the transmission control protocol (TCP). In a network that uses the Internet Protocol (IP).

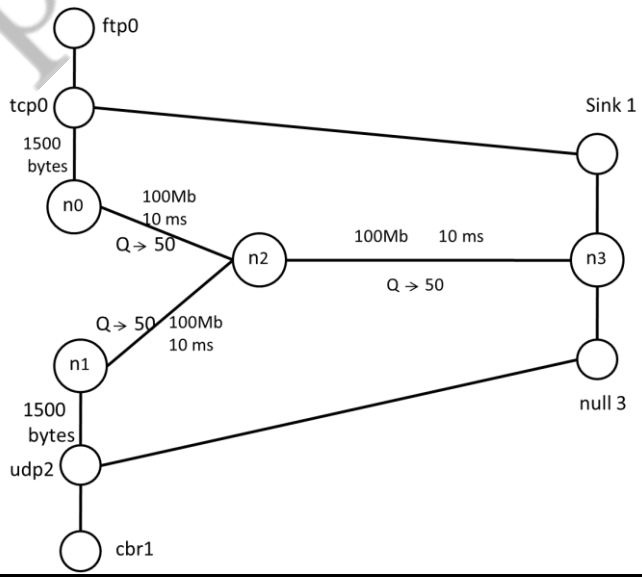
FTP: The File Transfer Protocol is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network. FTP is built on a client–server model architecture using separate control and data connections between the client and the server.

CBR: Constant bitrate is a term used in telecommunications, relating to the quality of service. Compare with variable bitrate. When referring to codecs, constant bit rate encoding means that the rate at which a codec's output data should be consumed is constant.

Requirements for Network Setup:

- Four nodes
- Point to point network
- Duplex links
- TCP agent and FTP application
- UDP agent and CBR application

Network Diagram:



TCL Script:

```
#=================================
#     Simulation parameters setup
#=================================
setval(stop)   10.0                ;# time of simulation end

#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
settracefile [open lab2.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab2.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#=================================
#       Links Definition
#=================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 100.0Mb 10ms DropTail     ## Vary the bandwidth for different cases ##
$ns queue-limit $n0 $n2 50                        ## Vary the queue size for different cases ##
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail     ## Vary the bandwidth for different cases ##
$ns queue-limit $n1 $n2 50                        ## Vary the queue size for different cases ##
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail     ## Vary the bandwidth for different cases ##
$ns queue-limit $n2 $n3 50                        ## Vary the queue size for different cases ##

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink1
```

```
$tcp0 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n1 $udp2
set null3 [new Agent/Null]
$ns attach-agent $n3 $null3
$ns connect $udp2 $null3
$udp2 set packetSize_ 1500


#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 8.0 "$ftp0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp2
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb                    ## Vary the CBR rate for different cases ##
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 8.0 "$cbr1 stop"


#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab2.nam &
exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

AWK Script:

```
BEGIN{
tcppkt=0;
udppkt=0;
}
{
if($1=="r"&&$4=="3"&&$5=="tcp"&&"1540")
tcppkt++;
if($1=="r"&&$4=="3"&&$5=="cbr"&&"1000")
udppkt++;
}
END{
printf("\n\nn0 tcppkts&udppkts:%d\n%d\n\n",tcppkt,udppkt);
}
```

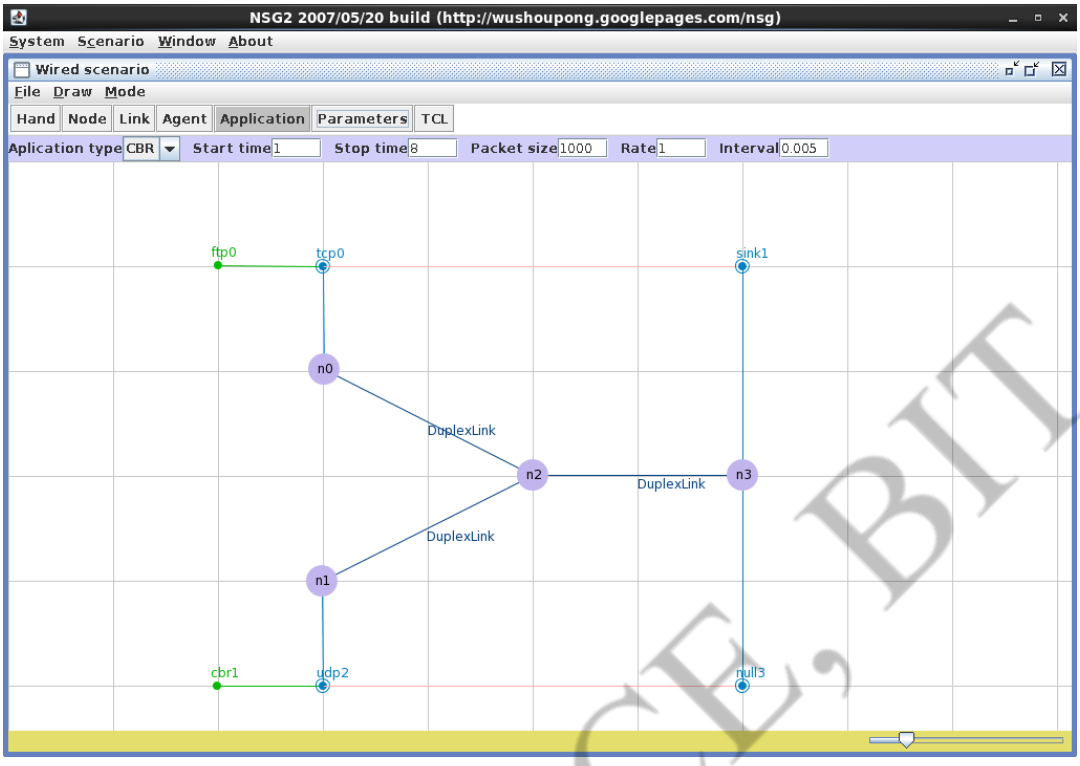
Execution Steps:

1. Create TCL code, save as eg: lab2.tcl
2. Create AWK code using the command eg: gedit lab2.awk
3. Run the network simulator in the terminal: ns lab2.tcl.
4. Observe the output in nam window and tracefile.
5. Check the analysis of tracefile: awk –f lab2.awk lab2.tr.
6. Vary the parameters of bandwidth, queue size in lab2.tcl.
(Open the file using command, gedit lab2.tcl)
7. Run ns again, check the output, tabulate the readings.

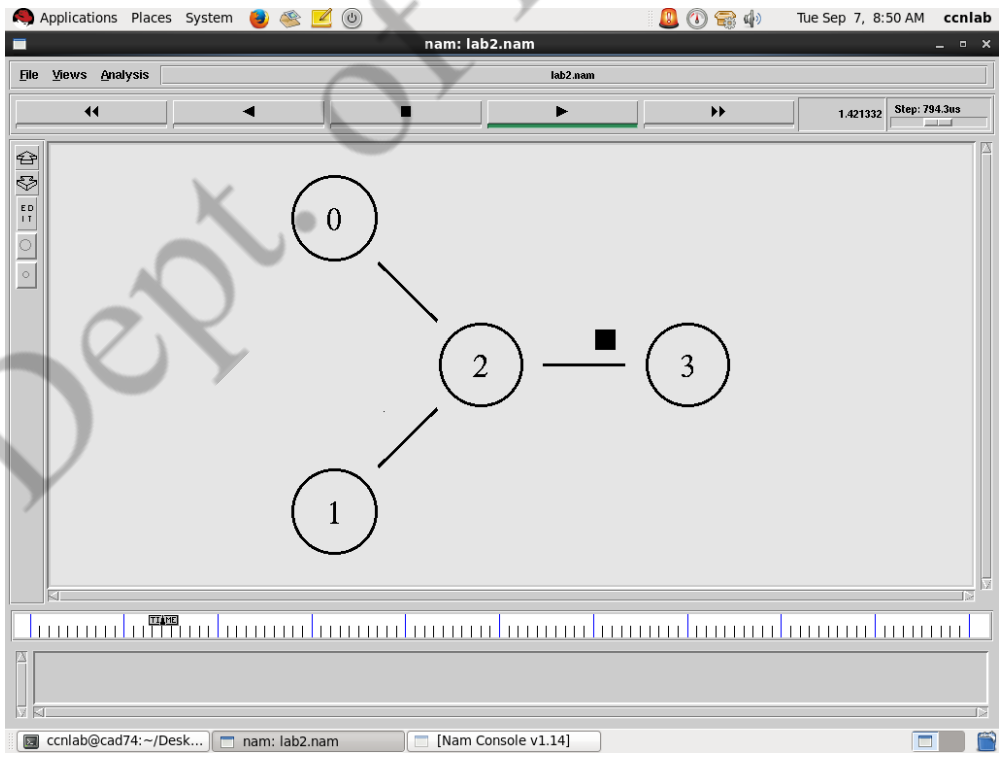
Data Inputs:

N0-N2	N1-N2	N2-N3	CBR Rate
100Mb 10ms Q-50	100Mb 10ms Q-50	100Mb 10ms Q-50	1.0
1Mb 10ms Q-50	1Mb 10ms Q-50	1Mb 10ms Q-50	1.0
1Mb 10ms Q-50	1Mb 10ms Q-50	1Mb 10ms Q-2	1.0
1Mb 10ms Q-5	1 Mb 10ms Q-50	1Mb 10ms Q-50	1.5

Network Window:



NAM Window:



Trace File:

```
+ 1 0 2 tcp 40 ------- 0 0.0 3.0 0 0
- 1 0 2 tcp 40 ------- 0 0.0 3.0 0 0
+ 1 1 2 cbr 1000 ------- 0 1.0 3.1 0 1
- 1 1 2 cbr 1000 ------- 0 1.0 3.1 0 1
+ 1.008 1 2 cbr 1000 ------- 0 1.0 3.1 1 2
- 1.008 1 2 cbr 1000 ------- 0 1.0 3.1 1 2
r 1.010003 0 2 tcp 40 ------- 0 0.0 3.0 0 0
+ 1.010003 2 3 tcp 40 ------- 0 0.0 3.0 0 0
- 1.010003 2 3 tcp 40 ------- 0 0.0 3.0 0 0
r 1.01008 1 2 cbr 1000 ------- 0 1.0 3.1 0 1
+ 1.01008 2 3 cbr 1000 ------- 0 1.0 3.1 0 1
- 1.01008 2 3 cbr 1000 ------- 0 1.0 3.1 0 1
+ 1.016 1 2 cbr 1000 ------- 0 1.0 3.1 2 3
- 1.016 1 2 cbr 1000 ------- 0 1.0 3.1 2 3
r 1.01808 1 2 cbr 1000 ------- 0 1.0 3.1 1 2
+ 1.01808 2 3 cbr 1000 ------- 0 1.0 3.1 1 2
- 1.01808 2 3 cbr 1000 ------- 0 1.0 3.1 1 2
r 1.020006 2 3 tcp 40 ------- 0 0.0 3.0 0 0
+ 1.020006 3 2 ack 40 ------- 0 3.0 0.0 0 4
- 1.020006 3 2 ack 40 ------- 0 3.0 0.0 0 4
r 1.02016 2 3 cbr 1000 ------- 0 1.0 3.1 0 1
+ 1.024 1 2 cbr 1000 ------- 0 1.0 3.1 3 5
- 1.024 1 2 cbr 1000 ------- 0 1.0 3.1 3 5
```

Output:

N0-N2	N1-N2	N2-N3	CBR Rate	TCP Packets	UDP Packets
100Mb 10ms Q-50	100Mb 10ms Q-50	100Mb 10ms Q-50	1.0		
1Mb 10ms Q-50	1Mb 10ms Q-50	1Mb 10ms Q-50	1.0		
1Mb 10ms Q-50	1Mb 10ms Q-50	1Mb 10ms Q-2	1.0		
1Mb 10ms Q-5	1 Mb 10ms Q-50	1Mb 10ms Q-50	1.5		

Conclusion/Inference: (To be written by students)

Experiment -3

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

Pre-requisites:

LAN: A local area network is a computer network that interconnects computers within a limited area such as a residence, school, laboratory, university campus or office building.

Throughput: Network throughput refers to how much data can be transferred from source to destination within a given timeframe. Throughput measures how many packets arrive at their destinations successfully. For the most part, throughput capacity is measured in bits per second, but it can also be measured in data per second.

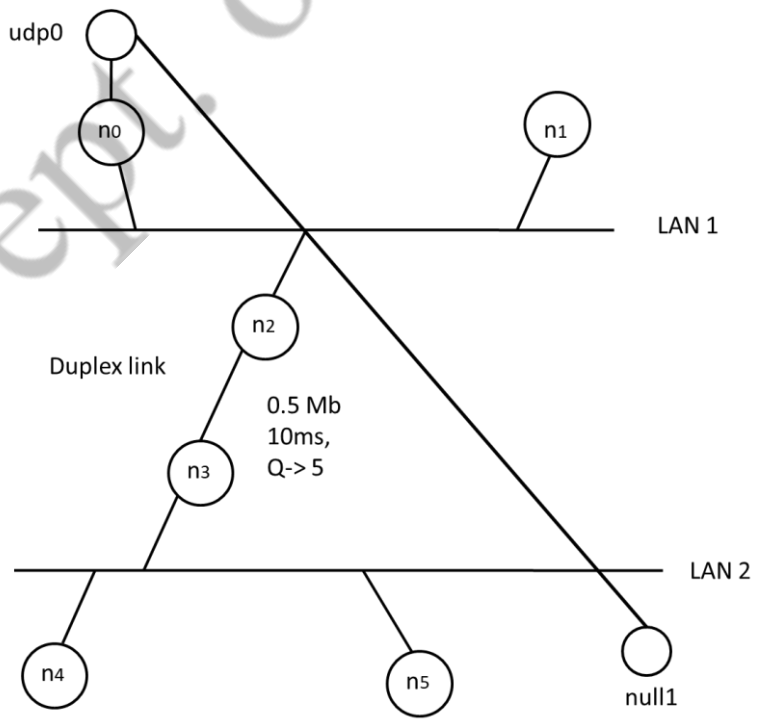
Error rate: The ratio of the number of erroneous units of data to the total number of units of data transmitted.

Data rate: The speed at which data is transferred within the computer or between a peripheral device and the computer, measured in bytes per second.

Requirements for Network Setup:

- 6 to 10 nodes
- Point to point network
- Duplex links
- UDP agent and CBR application
- LAN setup (to be done manually)

Network Diagram:



TCL Script:

```
#=================================
#     Simulation parameters setup
#=================================
setval(stop)   10.0                ;# time of simulation end

#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
settracefile [open lab3.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab3.nam w]
$ns namtrace-all $namfile

#=================================
#       Nodes Definition
#=================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#=================================
#       Links Definition
#=================================
#Createlinks between nodes
$ns duplex-link $n2 $n3 0.5Mb 10ms DropTail
$ns queue-limit $n2 $n3 5

$ns make-lan "$n0 $n1 $n2" 10Mb 10ms LL Queue/DropTail Mac/802_3 ## to be added manually ##
$ns make-lan "$n3 $n4 $n5" 10Mb 10ms LL Queue/DropTail Mac/802_3 ## to be added manually ##

#Give node position (for NAM)
$ns duplex-link-op $n2 $n3 orient right-down

set err [new ErrorModel]                              ## to be added manually ##
$ns lossmodel $err $n2 $n3                            ## to be added manually ##
$err set rate_ 0.2          ## to be added manually and vary the error rate for different cases ##

#=================================
#       Agents Definition
#=================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
```

```
set null1 [new Agent/Null]
$ns attach-agent $n5 $null1
$ns connect $udp0 $null1
$udp0 set packetSize_ 1500


#=================================
#       Applications Definition
#=================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb                          ## Vary the data rate for different cases ##
$cbr0 set random_ null

$udp0 set class_ 1                             ## to be added manually ##
$ns at 0.04 "$cbr0 start"
$ns at 8.0 "$cbr0 stop"

#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab3.nam &
exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

AWK Script:

```
BEGIN{
cnt=0;
time=0;
}
{
if($1=="r"&&$3=="2"&&$4=="3")
{
cnt+=$6;
time=$2;
}
}
END{
printf("\n total no. of bytes=%d\ttime=%1f",cnt,time);
printf("\n thrughput:%1f Mbps\n\n",(cnt/time)*(8/1000000));
}
```

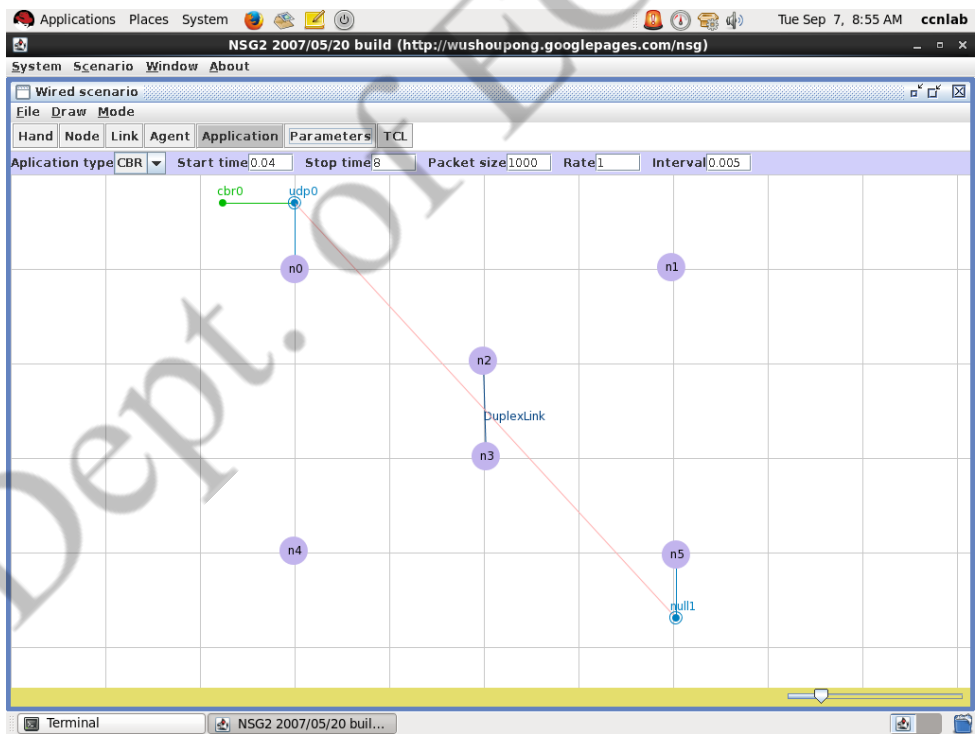
Execution Steps:

1. Create TCL code, save as eg: lab3.tcl
2. Create AWK code using the command eg: gedit lab3.awk
3. Run the network simulator in the terminal: ns lab3.tcl.
4. Observe the output in nam window and tracefile.
5. Check the analysis of tracefile: awk –f lab3.awk lab3.tr.
6. Vary the data rate and error rate in lab3.tcl.
(Open the file using command, gedit lab3.tcl)
7. Run ns again, check the output, tabulate the readings.

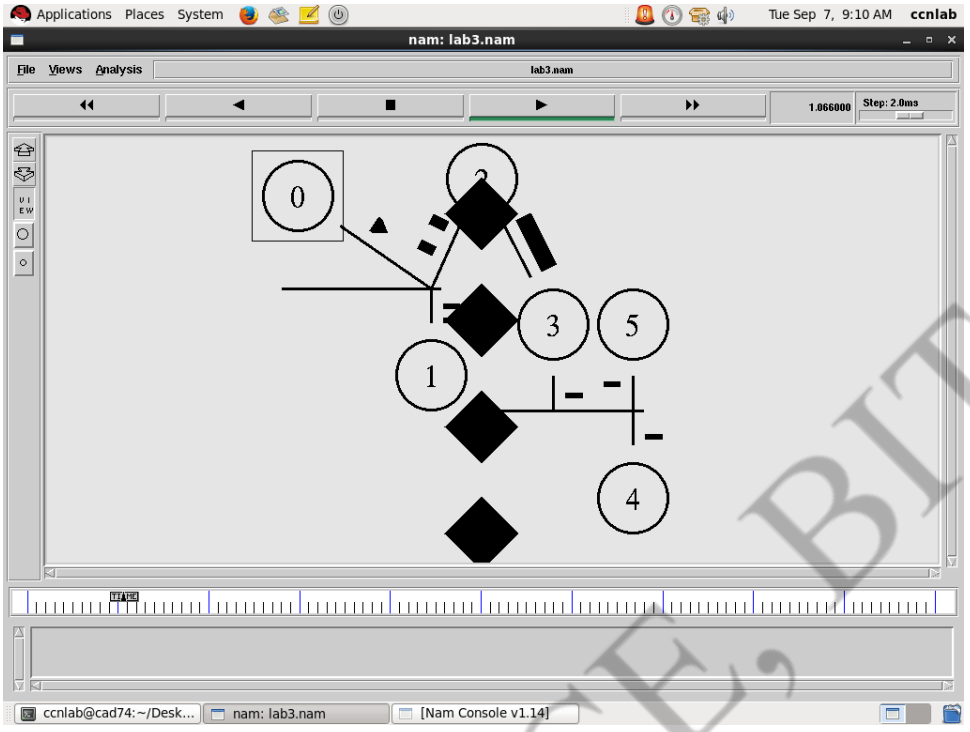
Data Inputs:

LAN N0, N1, N2, N3, N4, N5	Error Model N2-N3	Data Rate in Mb/s	Error Rate in Mb/s
10 Mb, 10ms	0.5Mb 10ms Q-5	1.0	0.2
10 Mb, 10ms	0.5Mb 10ms Q-5	0.5	0.2
10 Mb, 10ms	0.5Mb 10ms Q-5	0.5	0.5

Network Window:



NAM Window:



Trace File:

```
+ 0.05 0 6 cbr 1000 ------- 1 0.0 5.0 0 0
- 0.05 0 6 cbr 1000 ------- 1 0.0 5.0 0 0
h 0.056 0 6 cbr 1000 ------- 1 0.0 5.0 2 2
+ 0.058 0 6 cbr 1000 ------- 1 0.0 5.0 1 1
- 0.058 0 6 cbr 1000 ------- 1 0.0 5.0 1 1
r 0.060815 6 2 cbr 1000 ------- 1 0.0 5.0 0 0
d 0.060815 2 3 cbr 1000 ------- 1 0.0 5.0 0 0
h 0.064 0 6 cbr 1000 ------- 1 0.0 5.0 3 3
+ 0.066 0 6 cbr 1000 ------- 1 0.0 5.0 2 2
- 0.066 0 6 cbr 1000 ------- 1 0.0 5.0 2 2
r 0.068815 6 2 cbr 1000 ------- 1 0.0 5.0 1 1
+ 0.068815 2 3 cbr 1000 ------- 1 0.0 5.0 1 1
- 0.068815 2 3 cbr 1000 ------- 1 0.0 5.0 1 1
h 0.072 0 6 cbr 1000 ------- 1 0.0 5.0 4 4
+ 0.074 0 6 cbr 1000 ------- 1 0.0 5.0 3 3
- 0.074 0 6 cbr 1000 ------- 1 0.0 5.0 3 3
r 0.076815 6 2 cbr 1000 ------- 1 0.0 5.0 2 2
```

Output:

LAN N0, N1, N2, N3, N4, N5	Error Model N2-N3	Data Rate in Mb/s	Error Rate in Mb/s	Throughput in Mbps
10 Mb, 10ms	0.5Mb 10ms Q-5	1.0	0.2	
10 Mb, 10ms	0.5Mb 10ms Q-5	0.5	0.2	
10 Mb, 10ms	0.5Mb 10ms Q-5	0.5	0.5	

Conclusion/Inference: (To be written by students)

Experiment -4

Implement Ethernet LAN using n nodes and obtain congestion window for different sources/destination.

Pre-requisites:

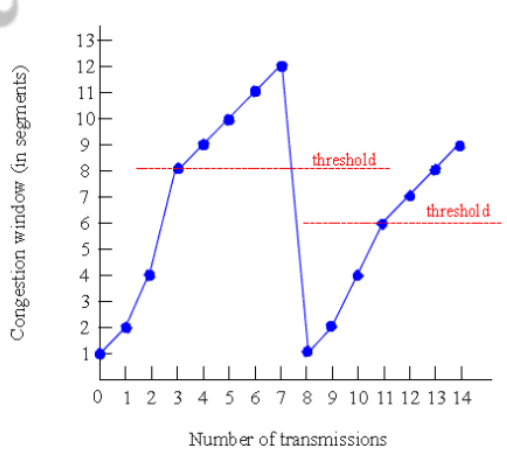
LAN: A local area network is a computer network that interconnects computers within a limited area such as a residence, school, laboratory, university campus or office building.

Congestion Window (cwnd): Congestion Window is a TCP state variable that limits the amount of data the TCP can send into the network before receiving an ACK. Together, the two variables are used to regulate data flow in TCP connections, minimize congestion, and improve network performance.

TCP connection consists of a receive buffer, a send buffer, and several variables (*LastByteRead, RcvWin*, etc.) The TCP congestion control mechanism has each side of the connection keep track of two additional variables: the congestion window and the threshold. The congestion window, denoted *CongWin*, imposes an additional constraint on how much traffic a host can send into a connection. Specifically, the amount of unacknowledged data that a host can have within a TCP connection may not exceed the minimum of *CongWin* and *RcvWin*, i.e.,*LastByteSent - LastByteAcked<= min{CongWin, RcvWin}.*

- When the congestion window is below the threshold, the congestion window grows exponentially.
- When the congestion window is above the threshold, the congestion window grows linearly.
- Whenever there is a timeout, the threshold is set to one half of the current congestion window and the congestion window is then set to one.

If we ignore the slow start phase, we see that TCP essentially increases its window size by 1 each RTT (and thus increases its transmission rate by an additive factor) when its network path is not congested, and decreases its window size by a factor of two each RTT when the path is congested. For this reason, TCP is often referred to as an additive-increase, multiplicative-decrease (AIMD) algorithm.



TCP Reno: A fast retransmit is sent, half of the current CWND is saved as *ssthresh* and as new CWND, thus skipping slow start and going directly to the congestion avoidance algorithm. The overall algorithm here is called fast recovery.

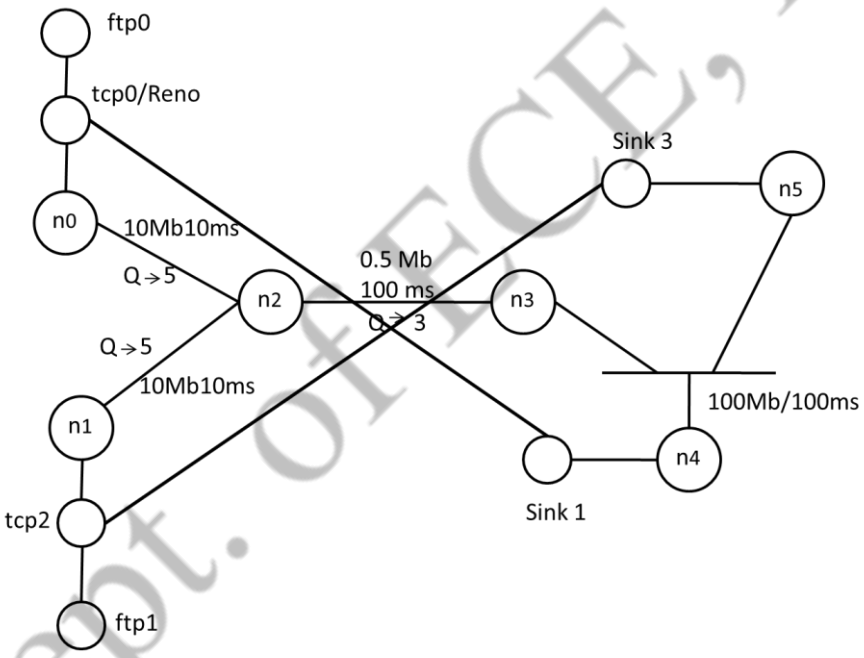
Once ssthresh is reached, TCP changes from slow-start algorithm to the linear growth (congestion avoidance) algorithm. At this point, the window is increased by 1 segment for each round-trip delay time (RTT).

Slow start assumes that unacknowledged segments are due to network congestion. While this is an acceptable assumption for many networks, segments may be lost for other reasons, such as poor data link layer transmission quality. Thus, slow start can perform poorly in situations with poor reception, such as wireless networks.

Requirements for Network Setup:

- 6 nodes
- Point to point network
- Duplex links
- TCP agent and FTP application
- LAN setup (to be done manually)

Network Diagram:



TCL Script:

```
#==================================
#     Simulation parameters setup
#==================================
setval(stop)   10.0                ;# time of simulation end

#==================================
#        Initialization
#==================================
#Create a ns simulator
set ns [new Simulator]
```

```
#Open the NS trace file
settracefile [open lab4.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab4.nam w]
$ns namtrace-all $namfile

#===================================
#       Nodes Definition
#===================================
#Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#===================================
#       Links Definition
#===================================
#Createlinks between nodes
$ns duplex-link $n0 $n2 10.0Mb 10ms DropTail
$ns queue-limit $n0 $n2 5
$ns duplex-link $n1 $n2 10.0Mb 10ms DropTail
$ns queue-limit $n1 $n2 5
$ns duplex-link $n2 $n3 0.5Mb 100ms DropTail
$ns queue-limit $n2 $n3 3

$ns make-lan "$n3 $n4 $n5" 10Mb 10ms LL Queue/DropTail Mac/802_3  ## to be added manually##

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#===================================
#       Agents Definition
#===================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n4 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 500

#Setup a TCP/Reno connection
set tcp2 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp2
set sink3 [new Agent/TCPSink]
$ns attach-agent $n5 $sink3
$ns connect $tcp2 $sink3
$tcp2 set packetSize_ 600
```

```
#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ftp0 set type_ FTP                     ## to be added manually##
set file1 [open file1.tr w]             ## to be added manually##
$tcp0 attach $file1                     ## to be added manually##
$tcp0 trace cwnd_                       ## to be added manually##

$ns at 0.1 "$ftp0 start"
$ns at 1.5 "$ftp0 stop"
$ns at 2.0 "$ftp0 start"                ## to be added manually##
$ns at 3.0 "$ftp0 stop"                 ## to be added manually##

#Setup a FTP Application over TCP/Reno connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp2

$ftp1 set type_ FTP                     ## to be added manually##
set file2 [open file2.tr w]             ## to be added manually##
$tcp2 attach $file2                     ## to be added manually##
$tcp2 trace cwnd_                       ## to be added manually##

$ns at 0.2 "$ftp1 start"
$ns at 2.0 "$ftp1 stop"
$ns at 2.5 "$ftp1 start"                ## to be added manually##
$ns at 4.0 "$ftp1 stop"                 ## to be added manually##


#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab4.nam &
exit 0
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

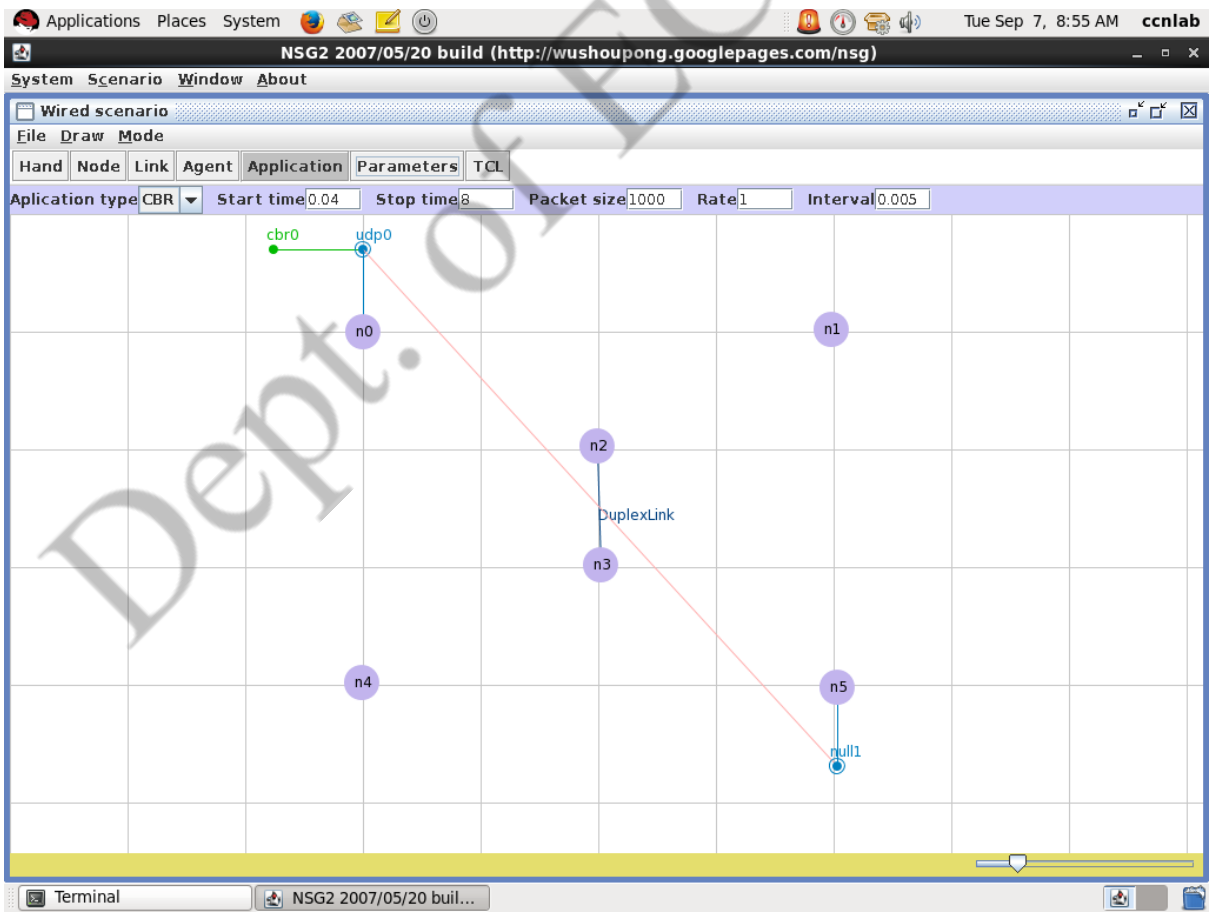
AWK Script:

```
BEGIN{
}
{
if($6=="cwnd_")
printf("\n %f \t %f", $1,$7);
}
END{
}
```

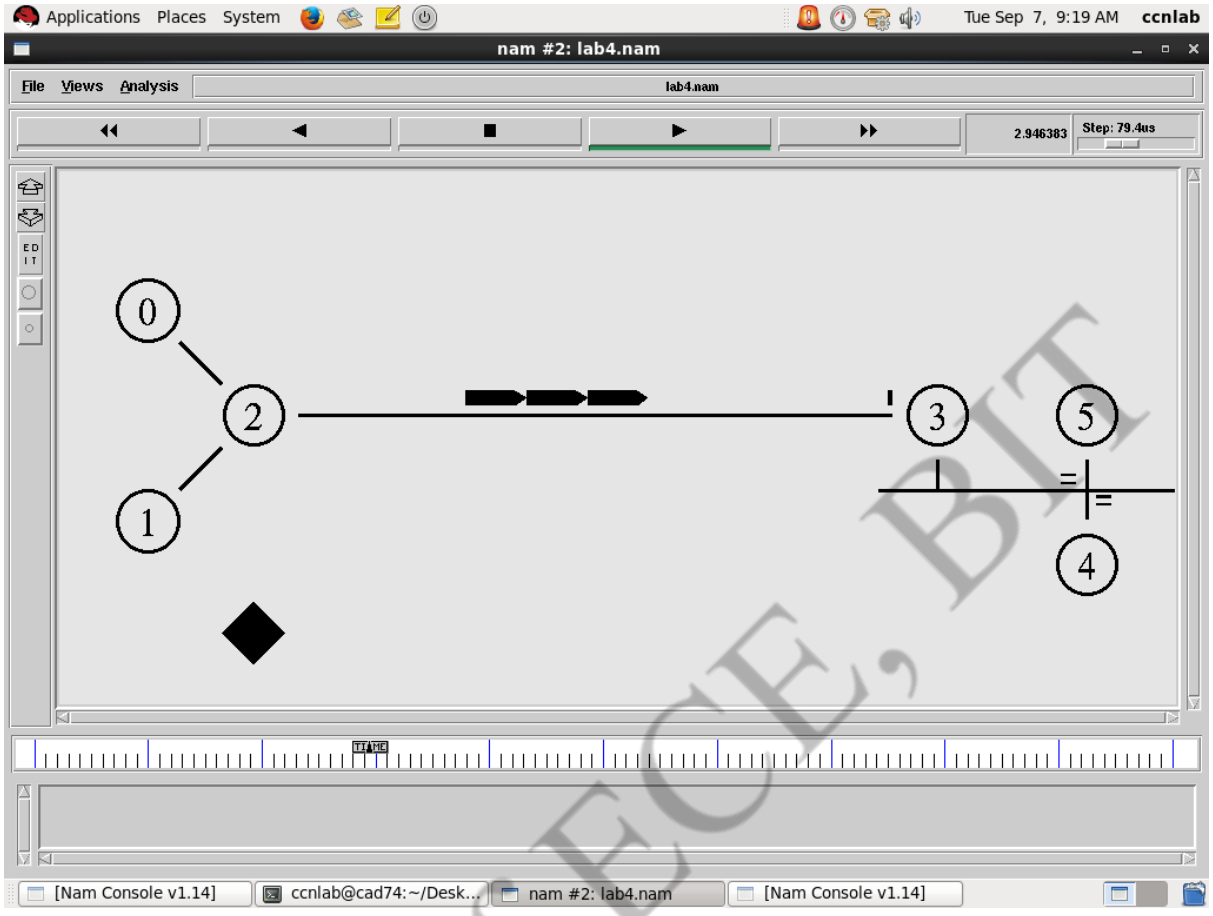
Execution Steps:

1. Create TCL code, save as eg: lab4.tcl
2. Create AWK code in gedit text editor, save as eg: lab4.awk
3. Run the network simulation in terminal: ns lab4.tcl
4. Observe the output in nam window and trace files
5. Check the analysis of trace file: awk –f lab4.awk file1.tr>a1
   awk –f lab4.awk file2.tr>a2
   xgraph a1 a2
6. Observe the graphical output of TCP congestion window (simulation time vs congestion)

Network Window:



NAM Window:



Trace File:

lab4.tr trace file:

```
+ 0.1 0 2 tcp 40 ------- 0 0.0 4.0 0 0
- 0.1 0 2 tcp 40 ------- 0 0.0 4.0 0 0
r 0.110032 0 2 tcp 40 ------- 0 0.0 4.0 0 0
+ 0.110032 2 3 tcp 40 ------- 0 0.0 4.0 0 0
- 0.110032 2 3 tcp 40 ------- 0 0.0 4.0 0 0
+ 0.2 1 2 tcp 40 ------- 0 1.0 5.0 0 1
- 0.2 1 2 tcp 40 ------- 0 1.0 5.0 0 1
r 0.210032 1 2 tcp 40 ------- 0 1.0 5.0 0 1
+ 0.210032 2 3 tcp 40 ------- 0 1.0 5.0 0 1
- 0.210032 2 3 tcp 40 ------- 0 1.0 5.0 0 1
```

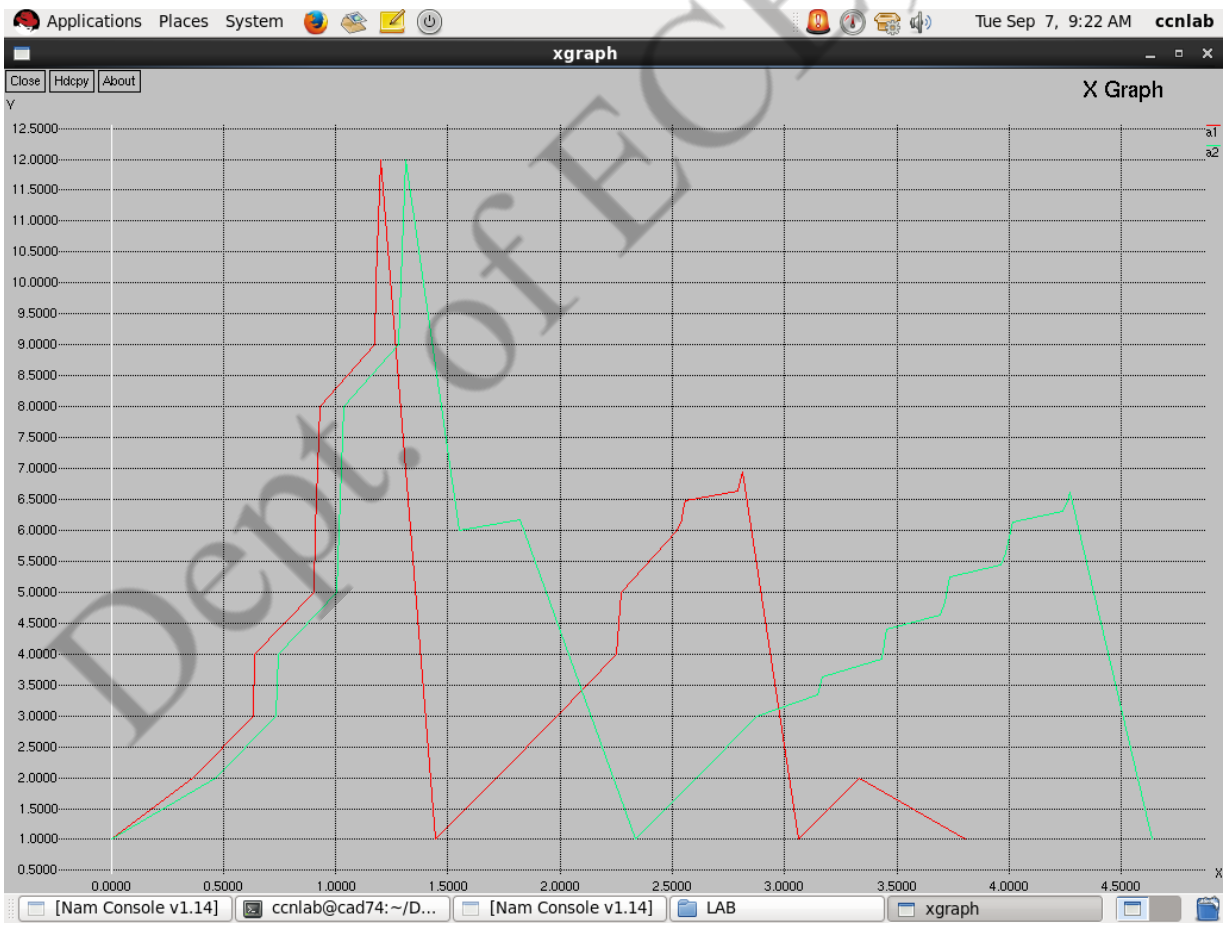
a1 file:

```
0.000000        1.000000
0.361450        2.000000
0.631700        3.000000
0.640340        4.000000
0.901950        5.000000
0.910590        6.000000
0.919230        7.000000
0.927870        8.000000
1.172190        9.000000
1.180830        10.000000
1.189470        11.000000
1.198110        12.000000
```

a2 file:

```
0.000000        1.000000
0.461450        2.000000
0.733460        3.000000
0.743700        4.000000
1.005960        5.000000
1.016200        6.000000
1.025950        7.000000
1.036190        8.000000
1.278450        9.000000
1.288690        10.000000
1.298930        11.000000
1.309170        12.000000
```

Graph:



Conclusion/Inference: (To be written by students)

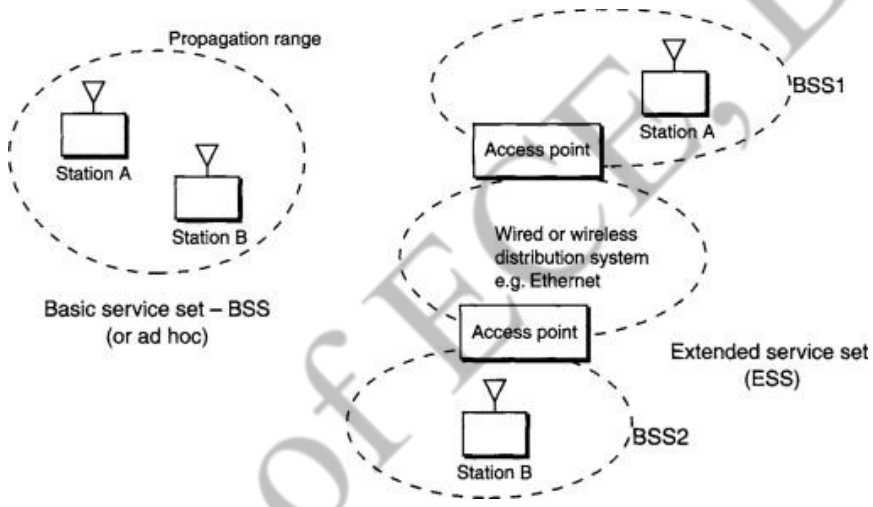
Experiment -5

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

Pre-requisites:

BSS: Basic Service Set. The Basic Service Set is a term used to describe the collection of Stations which may communicate together within an 802.11 network. The BSS may or may not include AP (Access Point) which provide a connection onto a fixed distribution system such as an Ethernet network.

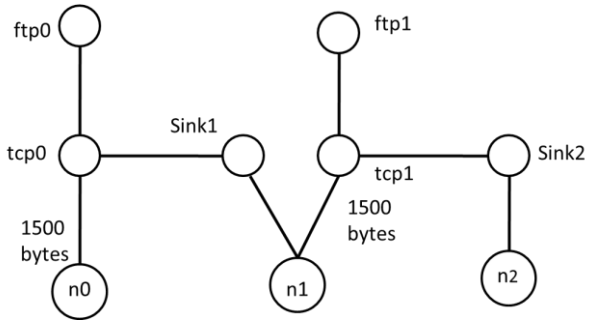
ESS: An extended service set (ESS) is a wireless network, created by multiple access points, which appears to users as a single, seamless network, such as a network covering a home or office that is too large for reliable coverage by a single access point.



Requirements for Network Setup:

- 3 nodes
- Wireless network
- TCP agent and FTP application

Network Diagram:



TCL Script:

```
#=================================
#     Simulation parameters setup
#=================================
setval(chan)   Channel/WirelessChannel    ;# channel type
setval(prop)   Propagation/TwoRayGround   ;# radio-propagation model
setval(netif)  Phy/WirelessPhy            ;# network interface type
setval(mac)    Mac/802_11                 ;# MAC type
setval(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
setval(ll)     LL                         ;# link layer type
setval(ant)    Antenna/OmniAntenna        ;# antenna model
setval(ifqlen) 50                         ;# max packet in ifq
setval(nn)     3                          ;# number of mobilenodes
setval(rp)     DSDV                       ;# routing protocol
setval(x)      1000                       ;# X dimension of topography
setval(y)      1000                       ;# Y dimension of topography
setval(stop)   50.0                       ;# time of simulation end

#=================================
#       Initialization
#=================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
settopo       [new Topography]
$topoload_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
settracefile [open lab5.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab5.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
setchan [new $val(chan)];#Create wireless channel

#=================================
#    Mobile node parameter setup
#=================================
$ns node-config -adhocRouting  $val(rp) \
            -llType        $val(ll) \
            -macType       $val(mac) \
            -ifqType       $val(ifq) \
            -ifqLen        $val(ifqlen) \
            -antType       $val(ant) \
            -propType      $val(prop) \
            -phyType       $val(netif) \
            -channel       $chan \
            -topoInstance  $topo \
            -agentTrace    ON \
            -routerTrace   ON \
```

```
        -macTrace      ON \
        -movementTrace ON

#=================================
#       Nodes Definition
#=================================
#Create 3 nodes
set n0 [$ns node]
$n0 set X_ 79                   ## redefine the coordinates to 50 ##
$n0 set Y_ 129                  ## redefine the coordinates to 50 ##
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 170                  ## redefine the coordinates to 100 ##
$n1 set Y_ 220                  ## redefine the coordinates to 100 ##
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 200                  ## redefine the coordinates to 300 ##
$n2 set Y_ 250                  ## redefine the coordinates to 300 ##
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20

#=================================
#       Agents Definition
#=================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink2 [new Agent/TCPSink]
$ns attach-agent $n1 $sink2
$ns connect $tcp0 $sink2
$tcp0 set packetSize_ 1500

#Setup a TCP connection
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set sink3 [new Agent/TCPSink]
$ns attach-agent $n2 $sink3
$ns connect $tcp1 $sink3
$tcp1 set packetSize_ 1500


#=================================
#       Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 50.0 "$ftp0 stop"

#Setup a FTP Application over TCP connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

```
$ns at 1.0 "$ftp1 start"
$ns at 50.0 "$ftp1 stop"

$ns at 0.1 "$n0 setdest 50 50 15"                ## to be added manually ##
$ns at 0.1 "$n1 setdest 100 100 25"              ## to be added manually ##
$ns at 0.1 "$n2 setdest 300 300 25"              ## to be added manually ##
$ns at 15 "$n0 setdest 550 550 15"               ## to be added manually ##
$ns at 30 "$n1 setdest 70 70 15"                 ## to be added manually ##
$ns at 20 "$n2 setdest 200 100 15"               ## to be added manually ##

#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab5.nam &
exit 0
}
for {set i 0} {$i< $val(nn) } { incri } {
   $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

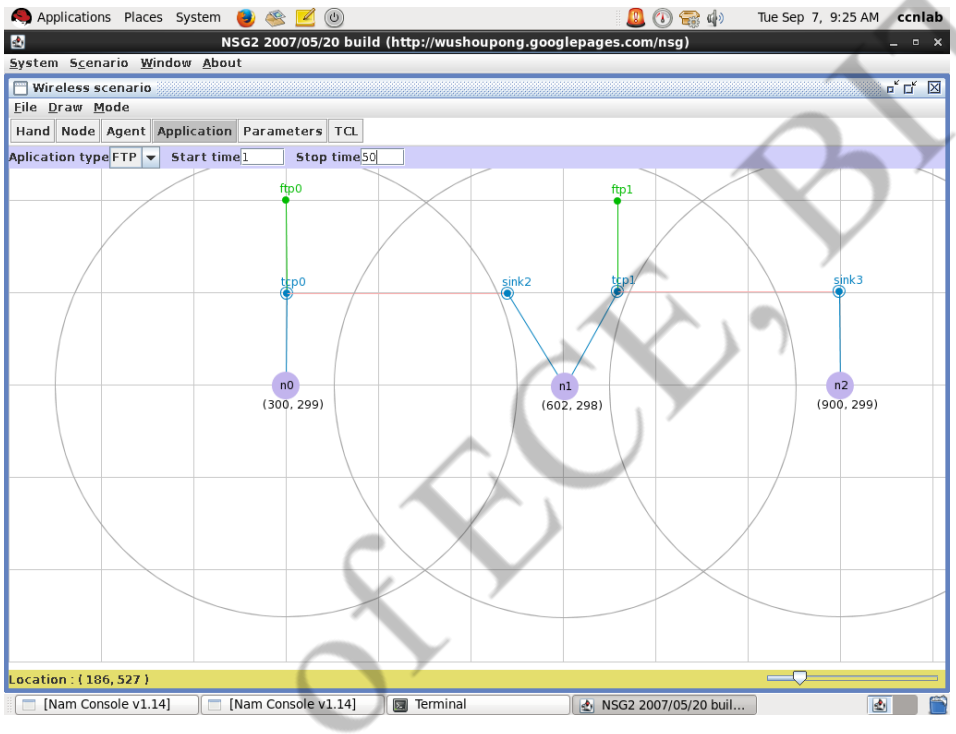
AWK Script:

```
BEGIN{
tcpcnt1=0;
tcpcnt2=0;
}
{
if($1=="r"&&$3=="_1_"&&$4=="AGT")
{
        tcpcnt1++;
}
if($1=="r"&&$3=="_2_"&&$4=="AGT")
{
        tcpcnt2++;
}
}
END{
printf("\n total no tcppktsrecd at node 1 and node 2 : %d\n %d\n", tcpcnt1, tcpcnt2);
}
```

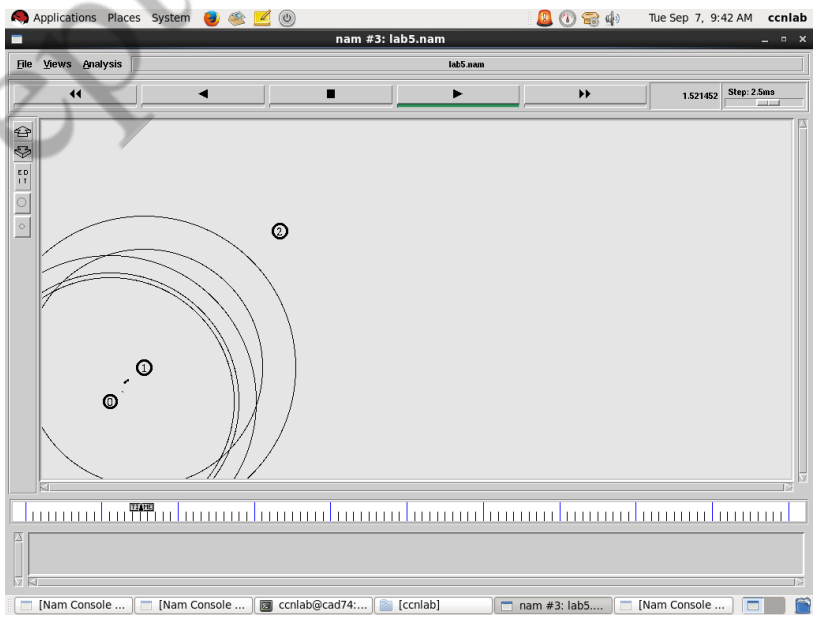
Execution Steps:

1. Create TCL code, save as eg: lab5.tcl
2. Create AWK code using the command eg: gedit lab5.awk
3. Run the network simulator in the terminal: ns lab5.tcl.
4. Observe the output in nam window and trace file.
5. Check the analysis of trace file: awk –f lab5.awk lab5.tr.
6. Run ns again, check the output, tabulate the readings.

Network Window:

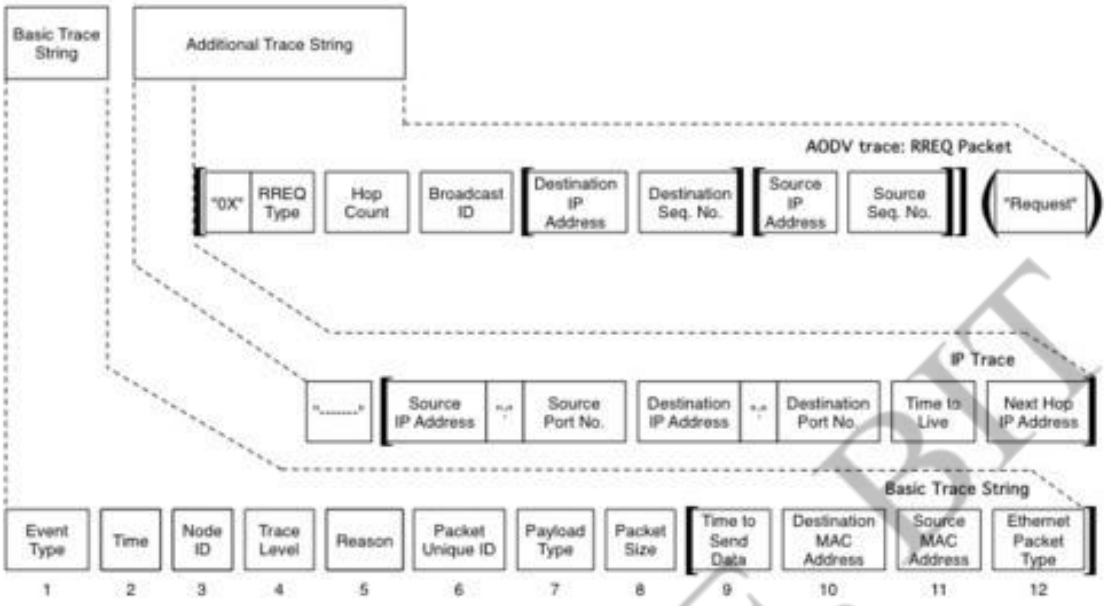


NAM Window:



Trace File:

Trace file format for wireless scenario:



Trace file output:

```
s 0.036400876 _0_ RTR  --- 0 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]
s 0.036675876 _0_ MAC  --- 0 message 90 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
r 0.037396112 _1_ MAC  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
r 0.037421112 _1_ RTR  --- 0 message 32 [0 ffffffff 0 800] ------- [0:255 -1:255 32 0]
M 0.10000 0 (50.00, 50.00, 0.00), (50.00, 50.00), 15.00
M 0.10000 1 (100.00, 100.00, 0.00), (100.00, 100.00), 25.00
M 0.10000 2 (300.00, 300.00, 0.00), (300.00, 300.00), 25.00
s 0.182633994 _1_ RTR  --- 1 message 32 [0 0 0 0] ------- [1:255 -1:255 32 0]
s 0.182948994 _1_ MAC  --- 1 message 90 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
r 0.183669230 _0_ MAC  --- 1 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
r 0.183694230 _0_ RTR  --- 1 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 0.882774710 _2_ RTR  --- 2 message 32 [0 0 0 0] ------- [2:255 -1:255 32 0]
s 0.883209710 _2_ MAC  --- 2 message 90 [0 ffffffff 2 800] ------- [2:255 -1:255 32 0]
s 1.000000000 _0_ AGT  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
r 1.000000000 _0_ RTR  --- 3 tcp 40 [0 0 0 0] ------- [0:0 1:0 32 0] [0 0] 0 0
s 1.000000000 _0_ RTR  --- 3 tcp 60 [0 0 0 0] ------- [0:0 1:0 32 1] [0 0] 0 0
s 1.000000000 _1_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
r 1.000000000 _1_ RTR  --- 4 tcp 40 [0 0 0 0] ------- [1:1 2:0 32 0] [0 0] 0 0
s 1.000215000 _0_ MAC  --- 0 ARP 86 [0 ffffffff 0 806] ------- [REQUEST 0/0 0/1]
r 1.000903236 _1_ MAC  --- 0 ARP 28 [0 ffffffff 0 806] ------- [REQUEST 0/0 0/1]
s 1.001298236 _1_ MAC  --- 0 RTS 44 [52e 0 1 0]
r 1.001650471 _0_ MAC  --- 0 RTS 44 [52e 0 1 0]
s 1.001660471 _0_ MAC  --- 0 CTS 38 [3f4 1 0 0]
r 1.001964707 _1_ MAC  --- 0 CTS 38 [3f4 1 0 0]
s 1.001974707 _1_ MAC  --- 0 ARP 86 [13a 0 1 806] ------- [REPLY 1/1 0/0]
r 1.002662943 _0_ MAC  --- 0 ARP 28 [13a 0 1 806] ------- [REPLY 1/1 0/0]
```

Output:

TCP packets received at node 1:
TCP packets received at node 2:

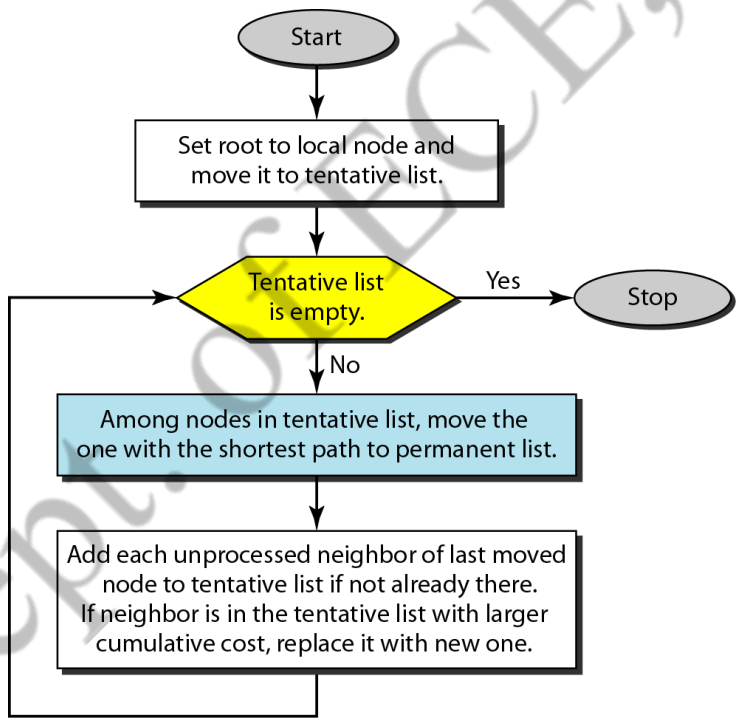
Conclusion/Inference: (To be written by students)

Experiment -6

Implementation of Link state routing algorithm

Pre-requisites:

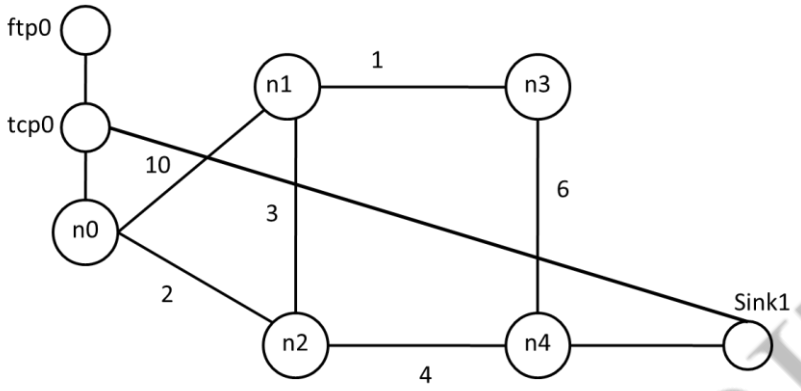
Link State Routing Algorithm: A routing algorithm that directly allows creating least-cost trees and forwarding tables is called link-state (LS) routing. This method uses the term link-state to define the characteristic of a link (an edge) that represents a network in the internet. To create a least-cost tree with this method, each node needs to have a complete map of the network, which means it needs to know the state of each link. The collection of states for all links is called the link-state database (LSDB).There is only one LSDB for the whole internet. Now the question is how each node can create this LSDB that contains information about the whole internet? This can be done by a process called flooding. Each node can send some greeting messages to all its immediate neighbors (those nodes to which it is connected directly) to collect two pieces of information for each neighboring node. The identity of the node and the cost of the link. To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous Dijkstra Algorithm



Requirements for Network Setup:

- 5 nodes
- Point to point network
- TCP agent and FTP application

Network Diagram:



TCL Script:

```
#==================================
#     Simulation parameters setup
#==================================
setval(stop)   10.0                ;# time of simulation end

#==================================
#        Initialization
#==================================
#Create a ns simulator
set ns [new Simulator]

#Open the NS trace file
settracefile [open lab6.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
setnamfile [open lab6.nam w]
$ns namtrace-all $namfile

#==================================
#        Nodes Definition
#==================================
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#==================================
#        Links Definition
#==================================
#Createlinks between nodes
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n1 50
$ns duplex-link $n1 $n2 100.0Mb 10ms DropTail
```

```
$ns queue-limit $n1 $n2 50
$ns duplex-link $n2 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n3 50
$ns duplex-link $n3 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n4 50
$ns duplex-link $n4 $n0 100.0Mb 10ms DropTail
$ns queue-limit $n4 $n0 50
$ns duplex-link $n4 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n4 $n1 50

#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient left-down
$ns duplex-link-op $n3 $n4 orient left

$ns duplex-link-op $n4 $n0 orient left-up
$ns duplex-link-op $n4 $n1 orient left-up

#===================================
#       Agents Definition
#===================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp0 $sink1
$tcp0 set packetSize_ 1500


#===================================
#       Applications Definition
#===================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns at 1.0 "$ftp0 start"
$ns at 4.0 "$ftp0 stop"

$ns rtproto LS                    ## to be added manually ##


#===================================
#       Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
global ns tracefilenamfile
   $ns flush-trace
close $tracefile
close $namfile
execnam lab6.nam &
exit 0
}
```

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

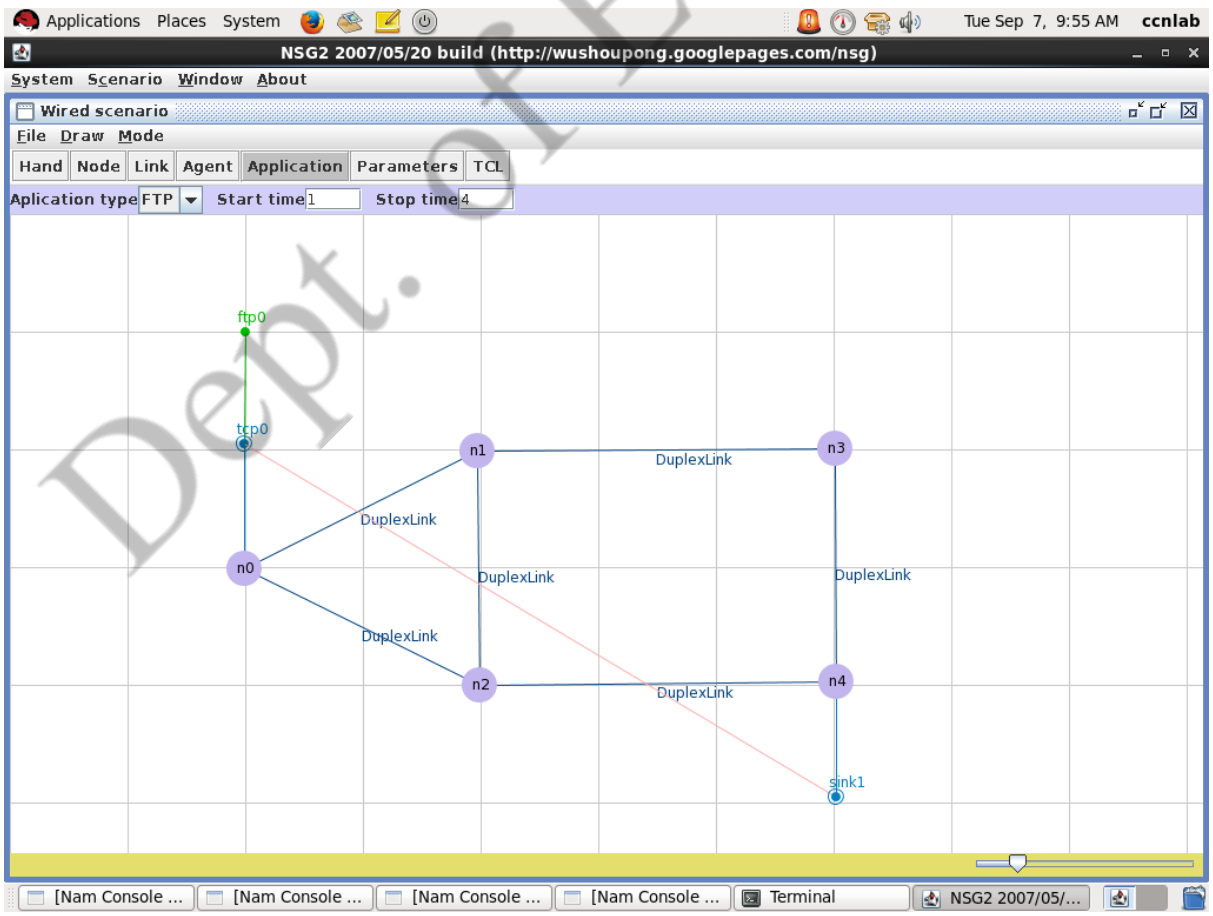
Execution Steps:

1. Create TCL code, save as eg: lab6.tcl
2. Run the network simulator in the terminal: ns lab5.tcl.
3. Observe the output in nam window and tracefile.
4. Tabulate the readings

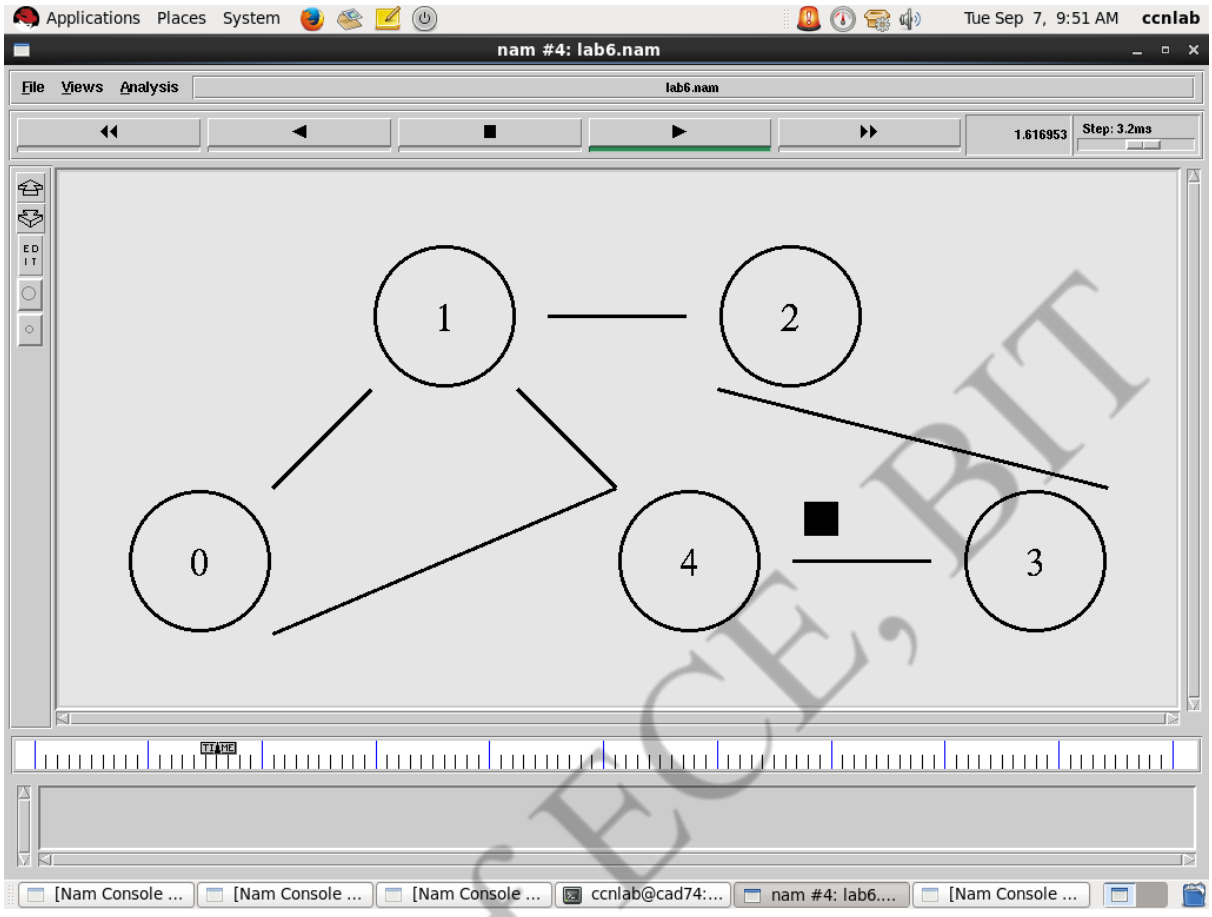
Working:

Origin/Path	A	B	C	D	E
	∞	∞	∞	∞	∞
{A}	-	10	2 ✓	∞	∞
{A,C}	-	5 ✓	2	∞	6
{A,C,B}	-	5	2	6 ✓	6
{A,C,B,D}	-	5	2	6	6 ✓
{A,C,B,D,E}	-	5	2	6	6

Network Window:



NAM Window:



Trace File:

```
+ 0.00017 0 1 rtProtoLS 100 ------- 0 0.2 1.1 -1 0
- 0.00017 0 1 rtProtoLS 100 ------- 0 0.2 1.1 -1 0
+ 0.00017 0 4 rtProtoLS 100 ------- 0 0.2 4.1 -1 1
- 0.00017 0 4 rtProtoLS 100 ------- 0 0.2 4.1 -1 1
+ 0.007102 2 1 rtProtoLS 100 ------- 0 2.1 1.1 -1 2
- 0.007102 2 1 rtProtoLS 100 ------- 0 2.1 1.1 -1 2
+ 0.007102 2 3 rtProtoLS 100 ------- 0 2.1 3.2 -1 3
- 0.007102 2 3 rtProtoLS 100 ------- 0 2.1 3.2 -1 3
r 0.010178 0 1 rtProtoLS 100 ------- 0 0.2 1.1 -1 0
+ 0.010178 1 0 rtProtoLS 20 ------- 0 1.1 0.2 -1 4
- 0.010178 1 0 rtProtoLS 20 ------- 0 1.1 0.2 -1 4
+ 0.010178 1 2 rtProtoLS 100 ------- 0 1.1 2.1 -1 5
- 0.010178 1 2 rtProtoLS 100 ------- 0 1.1 2.1 -1 5
+ 0.010178 1 4 rtProtoLS 100 ------- 0 1.1 4.1 -1 6
- 0.010178 1 4 rtProtoLS 100 ------- 0 1.1 4.1 -1 6
r 0.010178 0 4 rtProtoLS 100 ------- 0 0.2 4.1 -1 1
```

Output:

Source = n0, Destination = n4
No. of path from n0-n4 = 04
Shortest path from n0-n4 is n0 > n2 > n4 and cost = 06

Conclusion/Inference: (To be written by students)

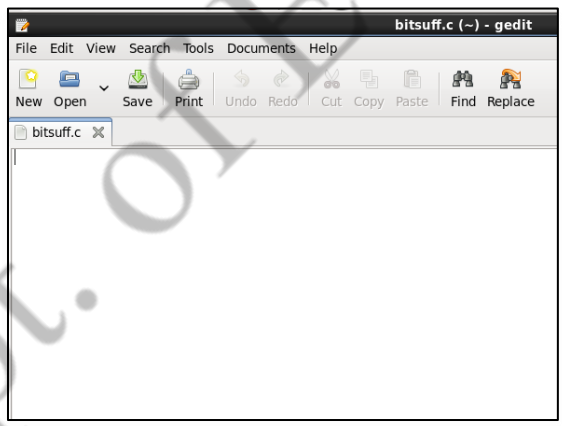
PART B

Execution Steps:

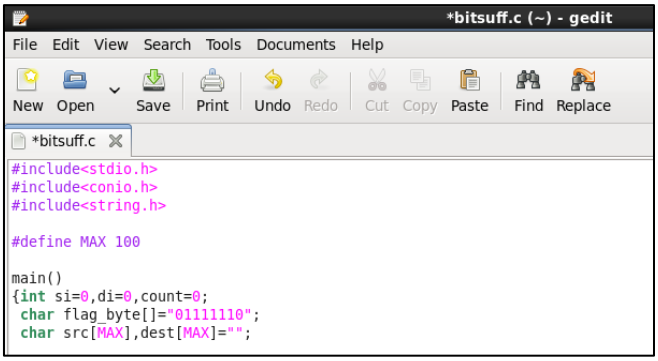
1) To type C Program:
Double click on terminal

and type: gedit filename.c (gedit is the text editor) in terminal window

2) Go to gedit text editor window,



Type the C program and save.

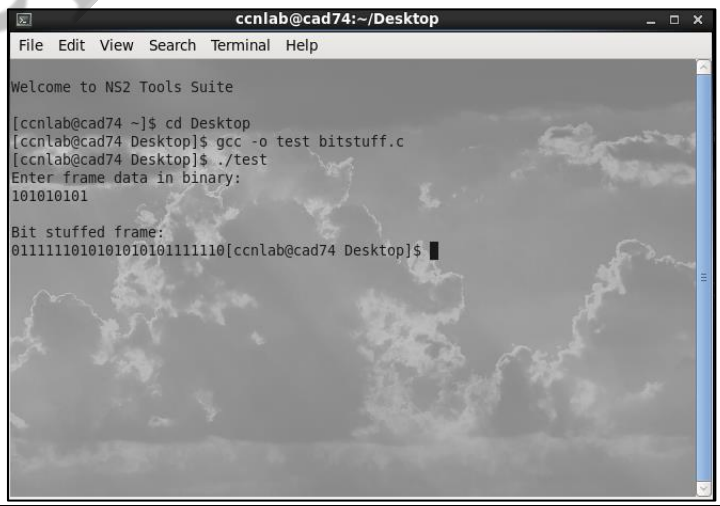


3) To Compile C Program: Go to terminal and type: gcc –o test bitstuff.c (test is a exe file)

Correct errors and save

4) To run Program: Go to terminal and type: ./test

Give the appropriate inputs and observe the outputs

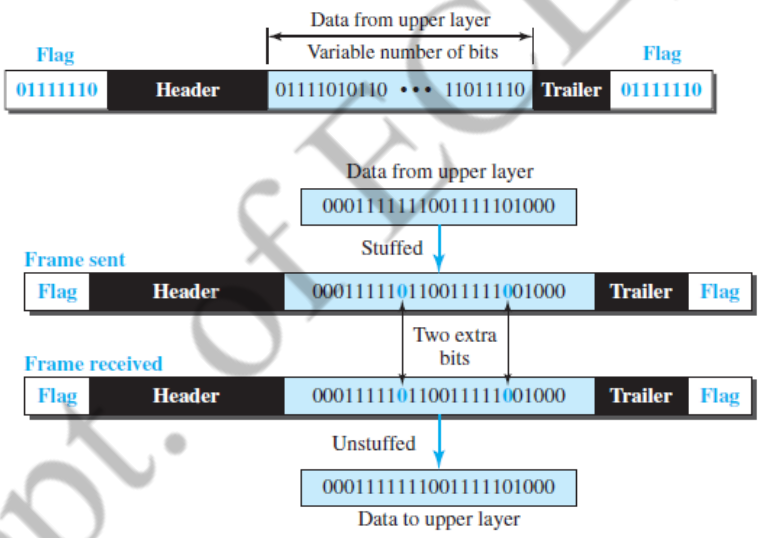


EXPERIMENT – 1

Write a C program to perform the following:
a) Bit stuffing
b) Character stuffing

Theory:

Bit stuffing: In bit-oriented framing, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other. Most protocols use a special 8-bit pattern flag, 01111110, as the delimiter to define the beginning and the end of the frame. This flag can create the same type of problem we saw in the character-oriented protocols. That is, if the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit (instead of 1 byte) to prevent the pattern from looking like a flag. The strategy is called bit stuffing. In bit stuffing, if a 0 and five consecutive 1 bits are encountered, an extra 0 is added. This extra stuffed bit is eventually removed from the data by the receiver. Note that the extra bit is added after one 0 followed by five 1s regardless of the value of the next bit. This guarantees that the flag field sequence does not inadvertently appear in the frame.



Algorithm:

Step1: Input data sequence
Step2: Add start of frame to data sequence
Step3: For every bit in Input, append bit to output, is bit a "1", if yes increment count. If count is 5, append 0 to output sequence and reset count. If not, set count to zero
Step4: Add stop of frame to the output sequence.

C Code:

```
#include<stdio.h>
#include<string.h>
#define MAX 100

main()
{int si=0,di=0,count=0;
char flag_byte[]="01111110";
char src[MAX],dest[MAX]="";
printf("Enter frame data in binary:\n");
scanf("%s",src);
strcat(dest,flag_byte);
di=strlen(flag_byte);
while(src[si]!='\0')
{  if(src[si]=='1')
count++;
else
count=0;
dest[di++]=src[si++];
if(count==5)
{ dest[di++]='0';
count=0;
          }
 }
dest[di++]='\0';
strcat(dest,flag_byte);
printf("\nBit stuffed frame:\n%s",dest);
}
```

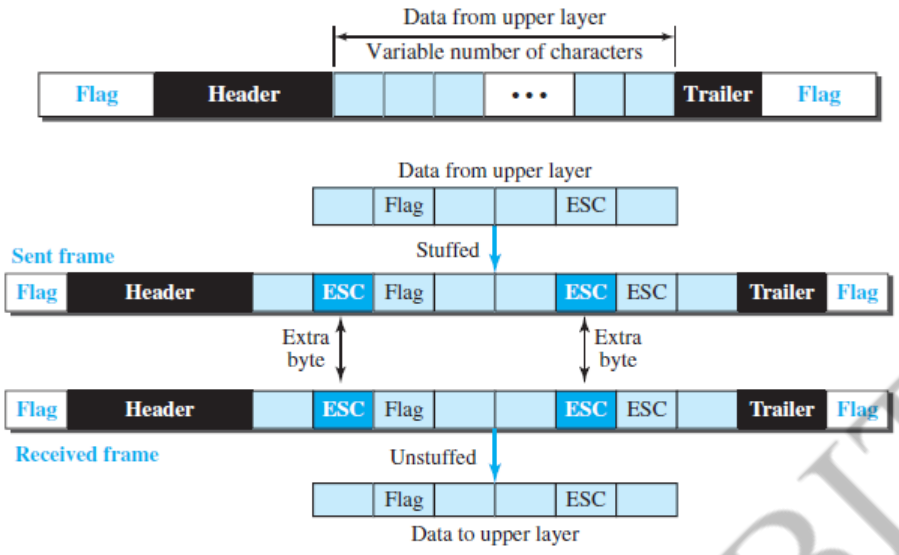
Output:

```
Enter frame data in binary:
10111011111111110
Bit stuffed frame:
01111110111011111011110001111110
```

b) Character stuffing

Theory:

Character stuffing: Character-oriented framing was popular when only text was exchanged by thedata-link layers. The flag could be selected to be any character not used for text communication. Now, however, we send other types of information such as graphs, audio, and video; any character used for the flag could also be part of the information. If this happens, the receiver, when it encounters this pattern in the middle of the data, thinks it has reached the end of the frame. To fix this problem, a byte-stuffing strategy was added to character-oriented framing. In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the *escape character (ESC)* and has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not as a delimiting. In the program Frame starts with DLE STX and ends with sequence DLE ETX(where DLE is data link escape, TSX is start of text and ETX is end of text). DLE is used as an escape character.



Algorithm:

Step1: Input data sequence
Step2: Add start of text to output sequence
Step3: Foe every character in input, append character to output sequence, is character 'DLE', add DLE to output sequence.
Step4: Add stop of frames to the output sequence.

C Code:

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>

#define MAX 100
int main()
{   int si=0,di=0;
char begin[]="DLE STX",end[]="DLE ETX";
        char src[MAX],dest[MAX]="";

        printf("Enter frame data in ASCII:\n");
        scanf("%s",src);
        strcat(dest,begin);
        di=strlen(begin);
        while(src[si]!='\0')
        {
if(((src[si]=='D')||(src[si]=='d'))&&((src[si+1]=='L')||(src[si+1]=='l'))&&((src[si+2]=='E')||(src[si+2]=
='e')))
          { dest[di+0]='D';dest[di+1]='L';dest[di+2]='E';
        dest[di+3]='D';dest[di+4]='L';dest[di+5]='E';
        di+=6;
        si+=3;
          }
        else
          {dest[di++]=src[si++];
    }
  }
```

```
dest[di]='\0';
strcat(dest,end);
printf("\n Character stuffed frame:\n%s",dest);
}
```

Output:

1) Enter frame data in ASCII:
BIT
Character stuffed frame:
DLESTX BIT DLE ETX

2) Enter frame data in ASCII:
BITDLE
Character stuffed frame:
DLESTX BITDLEDLE DLE ETX

Conclusion/Inference: (to be written by students)

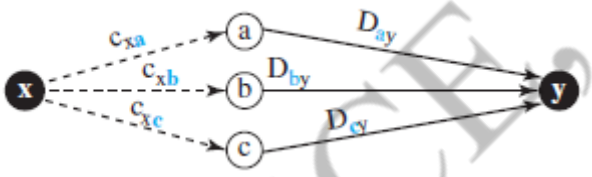
EXPERIMENT – 2

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

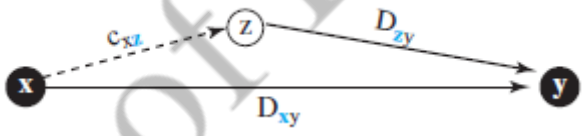
Theory:

Distance Vector Routing: In distance-vector routing, the first thing each node creates is its own least-cost tree with the rudimentary information it has about its immediate neighbors. The incomplete trees are exchanged between immediate neighbors to make the trees more and more complete and to represent the whole internet.

Bellman-Ford Equation: The heart of distance-vector routing is the famous Bellman-Ford equation. This equation is used to find the least cost (shortest distance) between a source node, x, and a destination node, y, through some intermediary nodes (a, b, c, . . .)
Dxy = min{(cxa + Day), (cxb + Dby), (cxc + Dcy), …}
$D_{xy}$is the shortest distance
$C_{xa}$is the cost between nodes xand a



In distance-vector routing, normally we want to update an existing least cost with a least cost through an intermediary node, such as *z*.
D*xy* = min{D*xy*, (c*xz* + D*zy*)}



Algorithm:

Step 1: Enter cost matrix
Step 2: Find minimum distance from one node to another and update.
Step 3: Record route from every node to all node.

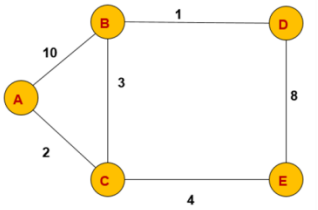
C Code:

```
#include<stdio.h>
#include<string.h>
#define MAX 100

main()
{int si=0,di=0,count=0;
char flag_byte[]="01111110";
char src[MAX],dest[MAX]="";
printf("Enter frame data in binary:\n");
scanf("%s",src);
strcat(dest,flag_byte);
di=strlen(flag_byte);
while(src[si]!='\0')
```

```
{  if(src[si]=='1')
count++;
else
count=0;
dest[di++]=src[si++];
if(count==5)
{ dest[di++]='0';
count=0;
          }
 }
dest[di++]='\0';
strcat(dest,flag_byte);
printf("\nBit stuffed frame:\n%s",dest);
}
```

Working:

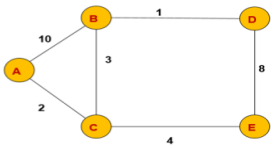


Step 1: Initial Routing Table

	A		B		C		D		E	
D	**C**	**H**	**C**	**H**	**C**	**H**	**C**	**H**	**C**	**H**
A	**0**	A	10	B	2	C	∞	-	∞	-
B	10	A	0	B	3	C	1	D	∞	-
C	2	A	3	B	0	C	∞	-	4	E
D	∞	-	1	B	∞	-	0	D	8	E
E	∞	-	∞	-	4	C	8	D	0	E

22.10

Step 2: Updating Routing table



	A		B		C		D		E	
D	**C**	**H**	**C**	**H**	**C**	**H**	**C**	**H**	**C**	**H**
A	**0**	A	5	C	2	C	6	B	6	C
B	5	C	0	B	3	C	1	D	7	C
C	2	A	3	B	0	C	4	B	4	E
D	6	B	1	B	4	B	0	D	8	E
E	6	C	7	C	4	C	8	D	0	E

B+10 **C+2**	**A+10** **C+3** **D+1**	**A+2** **B+3** **E+4**	**B+1** **E+8**	**C+4** **D+8**

22.12

Output:

```
Enter the number of nodes: 5
Enter the record for route a
<a : a> : : 0
<a : b> : :10
<a : c> : : 2
<a : d> : : 99
<a : e> : : 99
Enter the record for route b
<b : a> : : 10
<b : b> : : 0
<b : c> : : 3
<b : d> : : 1
<b : e> : : 99
Enter the record for route c
<c : a> : : 2
<c : b> : : 3
<c : c> : : 0
<c : d> : : 99
<c : e> : : 4
Enter the record for route d
<d : a> : : 99
<d : b> : : 1
<d : c> : : 99
<d : d> : : 0
<d : e> : : 8
Enter the record for route e
<e : a> : : 99
<e : b> : : 99
<e : c> : : 4
<e : d> : : 8
<e : e> : : 0
Cost of route a :
a : 0 via a
b : 5 via c
c : 2 via a
d : 6 via b
e : 6 via c
Cost of route b :
a : 5 via c
b : 0 via b
c : 3 via b
d : 1 via b
e : 7 via c
Cost of route c :
a : 2 via c
b : 3 via c
c : 0 via c
d : 4 via b
e : 4 via c
Cost of route d :
a : 6 via b
b : 1 via d
c : 4 via b
```

d : 0 via d
e : 8 via a
Cost of route e :
a : 6 via c
b : 7 via c
c : 4 via e
d : 8 via e
e : 0 via e

Conclusion/Inference: (to be written by students)

EXPERIMENT – 3

Implement Dijkstra's algorithm to compute the shortest routing path

Theory:

To create a least-cost tree with this method, each node needs to have a complete map of the network, which means it needs to know the state of each link. The collection of states for all links is called the link-state database (LSDB). There is only one LSDB for the whole internet; each node needs to have a duplicate of it to be able to create the least-cost tree. Figure 20.8 shows an example of an LSDB for the graph in Figure 20.1. The LSDB can be represented as a two-dimensional array (matrix) in which the value of each cell defines the cost of the corresponding link. Now the question is how each node can create this LSDB that contains information about the whole internet. This can be done by a process called flooding. To create a least-cost tree for itself, using the shared LSDB, each node needs to run the famous Dijkstra Algorithm.

This iterative algorithm uses the following steps:
1. The node chooses itself as the root of the tree, creating a tree with a single node, and sets the total cost of each node based on the information in the LSDB.
2. The node selects one node, among all nodes not in the tree, which is closest to the root, and adds this to the tree. After this node is added to the tree, the cost of all other nodes not in the tree needs to be updated because the paths may have been changed.
3. The node repeats step 2 until all nodes are added to the tree.

Algorithm:

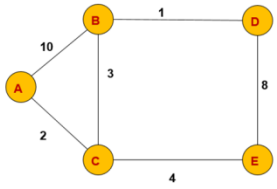
Step 1: Enter cost matrix
Step 2: Find minimum distance from the source to the visited node
Step 3: Data is updated from source similarly to all other nodes.
Step 4: Record the minimum distance and path from source to all other nodes.

C Code:

```
#include<stdio.h>
#define INFINITY 99
#define MAX 10
#define startnode 0
void dijkstra(int cost[MAX][MAX],int n);
int main()
{
int cost[MAX][MAX],i,j,n;
printf(" enter no of nodes: ");
scanf("%d",&n);
printf("\n enter the cost matrix:\n ");
for(i=0;i<n;i++)
    {for(j=0;j<n;j++)
    {
scanf("%d",&cost[i][j]);
    }
}
```

```
dijkstra(cost,n);
return 0;
 }
void dijkstra(int cost[MAX][MAX],int n)
     {
int dist[MAX],pred[MAX];
int visited[MAX],count,mindist,nextnode,i,j;
for(i=0;i<n;i++)
      {
dist[i]=cost[startnode][i];
pred[i]=startnode;
visited[i]=0;
}
dist[startnode]=0;
visited[startnode]=1;
count=1;
while(count<n-1)
{ mindist=INFINITY;
for(i=0;i<n;i++)
if(dist[i]<mindist&&!visited[i])
{
mindist=dist[i];
nextnode=i;
}
visited[nextnode]=1;
for(i=0;i<n;i++)
if(!visited[i])
if(mindist+cost[nextnode][i]<dist[i])
{
dist[i]=mindist+cost[nextnode][i];
pred[i]=nextnode;
}
count++;
}
for(i=0;i<n;i++)
if(i!=startnode)
{
printf("\ndistance of node %d from the startnode=%d",i,dist[i]);
printf("\npath=%d",i);
j=i;
do
{
j=pred[j];
printf("<-%d",j);
}
while(j!=startnode);
}
}
```

Working:



Origin/Path	A	B	C	D	E
	∞	∞	∞	∞	∞
{A}	-	10	2 ✓	∞	∞
{A,C}	-	5 ✓	2	∞	6
{A,C,B}	-	5	2	6 ✓	6
{A,C,B,D}	-	5	2	6	6 ✓
{A,C,B,D,E}	-	5	2	6	6

22.3

Output:

Enter the no. of node: 5
Enter the cost matrix:
0
10
5
99
99
10
0
3
1
99
5
3
0
99
2
99
1
99
0
6
99
99
2
6
0

Distance of node 1 from the startnode = 8
Path = 1 < 2 < 0
Distance of node 2 from the startnode = 5
Path = 2 < 0
Distance of node 3 from the startnode = 9
Path = 3 < 1 <2 < 0
Distance of node 4 from the startnode = 7
Path = 4 < 2 < 0

Conclusion/Inference: (to be written by students)

EXPERIMENT – 4

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

a) **Without error**
b) **With error**

Theory: A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short *check value* attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRCs can be used for error correction .

CRCs are so called because the *check* (data verification) value is a *redundancy* (it expands the message without adding information) and the algorithm is based on *cyclic* codes. CRCs are popular because they are simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function.

- At the sender side, the data unit to be transmitted is divided by a predetermined divisor (binary number) in order to obtain the remainder. This remainder is called CRC.
- The CRC has one bit less than the divisor. It means that if CRC is of n bits, divisor is of n+1 bit.
- The sender appends this CRC to the end of data unit such that the resulting data unit becomes exactly divisible by predetermined divisor i.e. remainder becomes zero.
- At the destination, the incoming data unit i.e data +CRC is divided by the same number (predetermined binary divisor).
- If the remainder after division is zero then there is no error in the data unit and receiver accepts it.
- If the remainder after division is not zero, it indicates that the data unit has been damaged in transit and therefore it is rejected.

Algorithm:

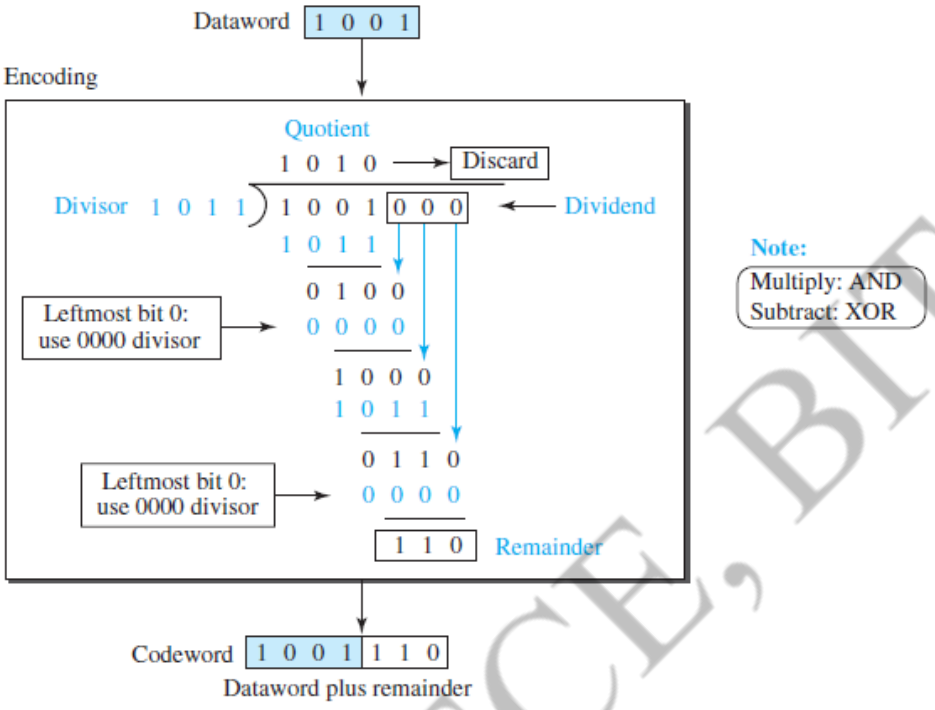
Step 1: To generate the CRC-CCITT code for the given bit pattern, append 16 zeroes to the end of the bit pattern.
Step 2: Divide this extended bit pattern by the 17 bit number corresponding to the generator polynomial.
Step 3: Replace the last 16 bits that you appended to the original bit pattern by the 16 bit remainder obtained in step 2.
Step 4: At the receiver, the code received is error free or in error by dividing the received data by the same generator polynomial.
Step 5: Implementation of stop and wait protocol and sliding window protocol.

C Code:

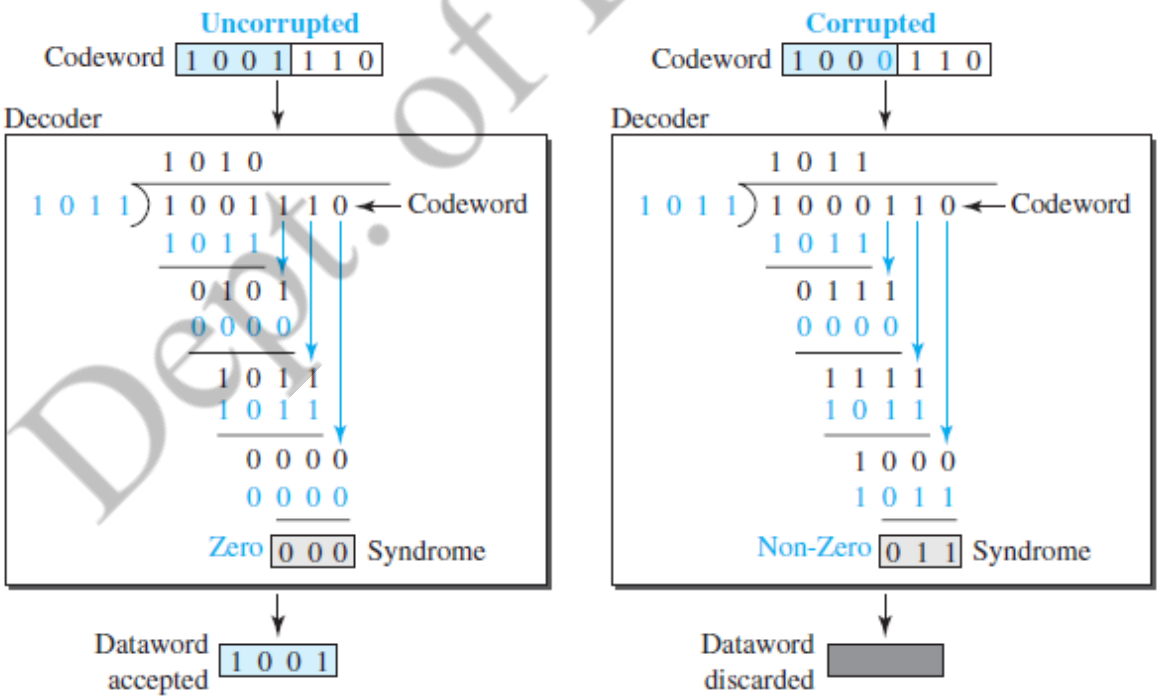
```
#include<stdio.h>
#include<string.h>
unsigned int xor2_div(char*i,char*o,int mode)
{
unsigned int j,k;
char g[81]={"10001000000100001"};
strcpy(o,i);
if(mode)
strcat(o,"0000000000000000");
for(j=0;j<strlen(i);j++)
if(*(o+j)=='1')
for(k=0;k<strlen(g);k++)
{if(((*(o+j+k)=='0')&&(g[k]=='0'))||((*(o+j+k)=='1')&&(g[k]=='1')))*(o+j+k)='0';
else
   *(o+j+k)='1';
}
for(j=0;j<strlen(o);j++)
if(o[j]=='1')
return(1);
return(0);
}
main()
{
char i[81]="\0",o[81]="\0",r[81]="\0";
printf("enter message code in binary:");
scanf("%s",i);
xor2_div(i,o,1);
printf("CCIT-CRC code is : \n%s%s\n",i,o+strlen(i));
printf("enter received code in binary:\n");
scanf("%s",r);
if(!xor2_div(r,o,0))
printf("received code is error free : \n%s",i);
else
printf("received code is erroneous");
}
```

Working:

Division in CRC encoder:



Division in the CRC decoder for two cases:



Output:

1) Without error:

Enter message code in binary: 10011
CCIT-CRC code is:
10011001000010010010

Enter received code in binary:
10011001000010010010

Received code is error free

2) With error

Enter message code in binary: 10011
CCIT-CRC code is:
10011001000010010010

Enter received code in binary:
10011001000010010011

Received code is erroneous

Conclusion/Inference: (to be written by students)

EXPERIMENT – 5

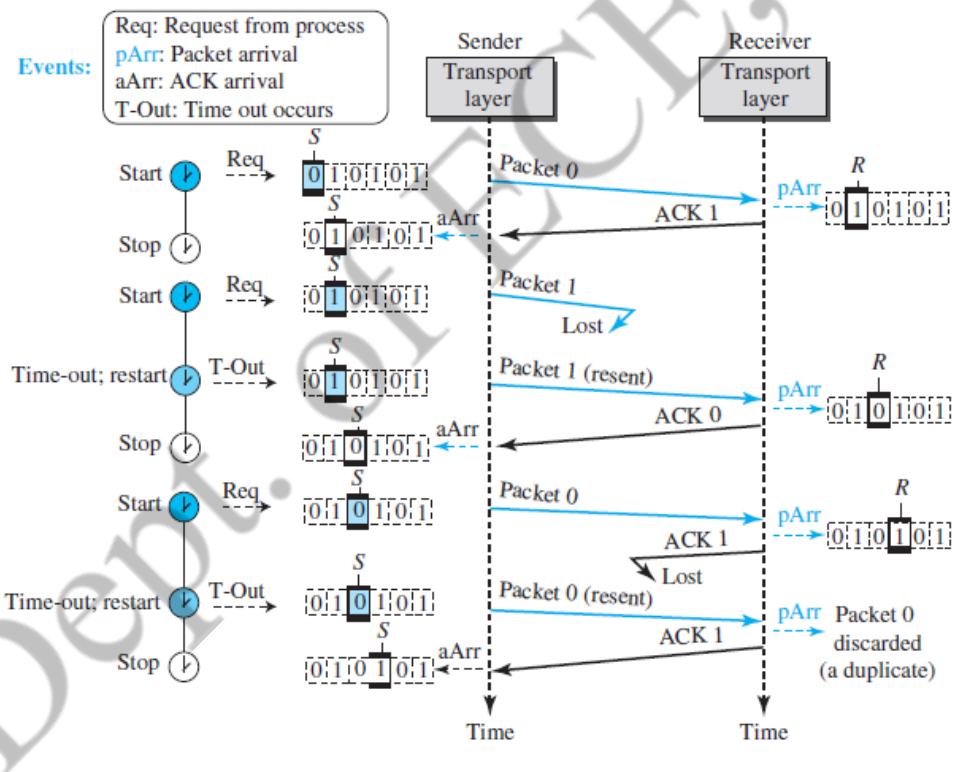
Implementation of stop and wait protocol and sliding window Protocol.

Theory: Stop and Wait Protocol

It is the simplest flow control method. In this, the sender will transmit one frame at a time to the receiver. The sender will stop and wait for the acknowledgement from the receiver.

This time (i.e. the time joining message transmitting and acknowledgement receiving) is the sender's waiting time, and the sender is idle during this time. When the sender gets the acknowledgement (ACK), it will send the next data packet to the receiver and wait for the disclosure again, and this process will continue as long as the sender has the data to send. While sending the data from the sender to the receiver, the data flow needs to be controlled. If the sender is transmitting the data at a rate higher than the receiver can receive and process it, the data will get lost. The Flow-control methods will help in ensuring that the data doesn't get lost. The flow control method will check that the senders send the data only at a rate that the receiver can receive and process.

The working of Stop and Wait Protocol is shown in the figure below −



The main advantage of stop & wait protocols is their accuracy. The next frame is transmitted only when the first frame is acknowledged. So there is no chance of the frame being lost. The drawback of this approach is that it is inefficient. It makes the transmission process slow. An individual frame travels from source to destination in this method, and a single acknowledgement travels from destination to source. As a result, each frame sent and received uses the entire time needed to traverse the link. Moreover, if two devices are a distance apart, a lot of time is wasted waiting for ACKs leading to an increase in total transmission time.

Features:

The features of Stop and Wait Protocol are as follows −

- It is used in Connection-oriented communication.
- It offers error and flows control.
- It can be used in data Link and transport Layers.
- Stop and Wait ARQ executes Sliding Window Protocol with Window Size 1.

Algorithm:

Step1: Enter the number of frames
Step2: Enter the first data
Step3: Enter the subsequent data, only when acknowledge.
Step4: If error, resend data
Step5: Repeat step 3 till end of frames

C Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
main()
{
int i=0,j=1,numframes,x;
char a[100]="";
printf("enter number of frames:");
scanf("%d",&numframes);
while(numframes>0)
{  printf("enter data:");
scanf("%c%*c",&a[i]);
printf("\nsending frame %d",i+1);
sleep(3);
x=rand()%10;
if((x%10==6)||(x%10==4))
{
sleep(5);
printf("\n error, resend data %d : ",i+1);
scanf("%c%*c",&a[i]);
printf("\nresending frame %d", i+1);
}
sleep(5);
printf(" \n ack %d\n",j+1);
numframes=numframes-1;
i++;
j++;
}
printf("\n end of stop and wait protocol\n");
}
```

Output:

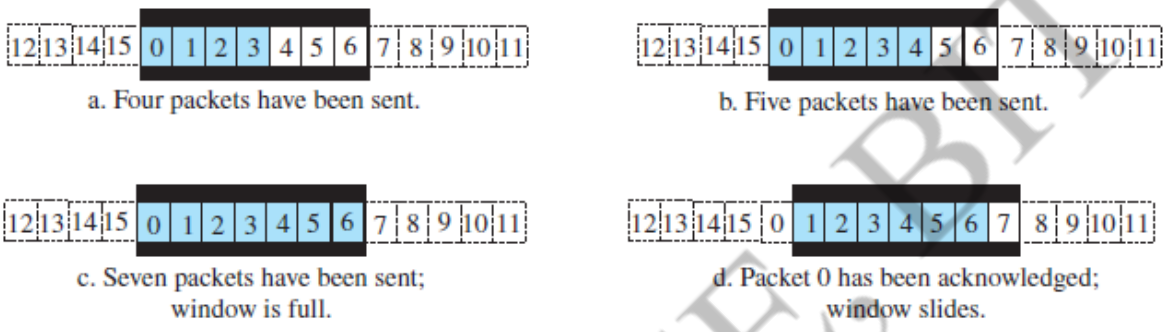
```
Enter number of frames: 10
Enter data: h
Sending frame 1
ack 2
Enter data: e
Sending frame 2
ack 3
Enter data: 1
Sending frame 3
Error, resend data 3: 1
Resending frame 3
ack 4
Enter data: 1
Sending frame 4
ack 5
Enter data: 0
Sending frame 5
ack 6
Enter data: w
Sending frame 6
Error, resend data 6: w
Resending frame 6
ack 7
Enter data: o
Sending frame 7
ack 8
Enter data: r
Sending frame 8
ack 9
Enter data: 1
Sending frame 9
ack 10
Enter data: d
Sending frame 10
Error, resend data 10: d
Resending frame 10
ack 11
End of stop and wait protocol
```

Conclusion/Inference: (to be written by students)

Theory: Sliding window Protocol

Sliding window protocol is a flow control protocol. It allows the sender to send multiple frames before needing the acknowledgements. Sender slides its window on receiving the acknowledgements for the sent frames. This allows the sender to send more frames. It is called so because it involves sliding of sender's window.

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 as well as in the Transmission Control Protocol (TCP). They are also used to improve efficiency when the channel may include high latency.

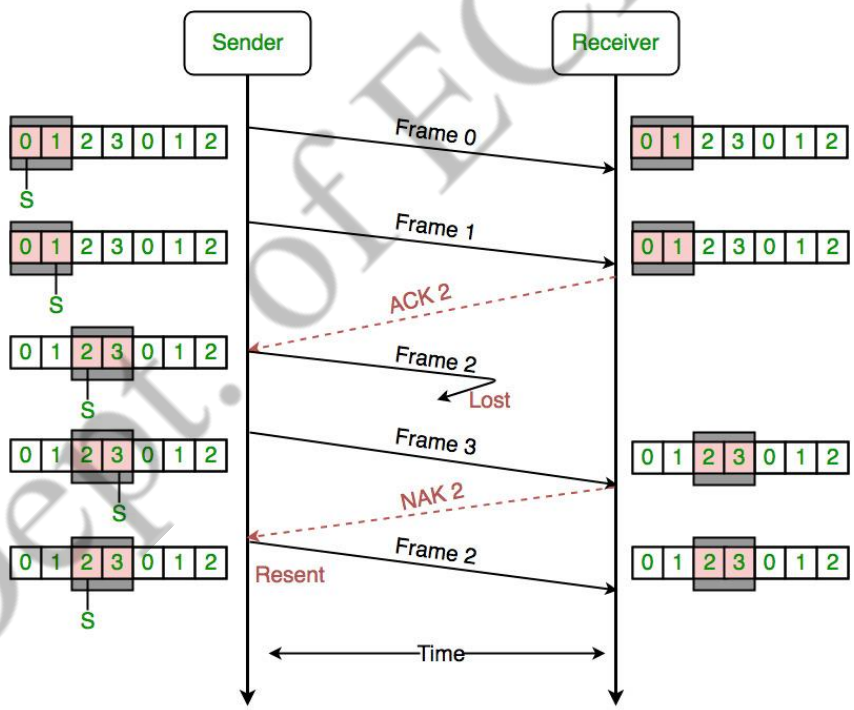


a. Four packets have been sent.

b. Five packets have been sent.

c. Seven packets have been sent; window is full.

d. Packet 0 has been acknowledged; window slides.



.Algorithm:

Step1: Enter the number of frames
Step2: Enter the first data
Step3: Enter the subsequent data's, while receiving acknowledgement.
Step4: If error, resend data
Step5: Repeat step 3 till end of frames.

C Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
main()
{
int i,j,numframes,x,y,count=0;
char a[100];
printf("enter number of frames:\n");
scanf("%d",&numframes);
for(i=0;i<numframes;i++)
{
printf("enter data %d:\n",i+1);
scanf("%c%*c",&a[i]);
x=rand()%10;
y=rand()%10;
if(i==x)
{
sleep(5);
printf("nak %d \n\n",i);
printf(" resend data %d : \n",i);
scanf("%c%*c",&a[i]);
}
if((y!=x)&&(y<x)&&(count<=1))
{count++;
sleep(3);
printf(" ack %d\n",i+1);
}
}
sleep(3);
printf(" ack %d\n",i+1);
printf("\n end of sliding window protocol\n");
}
```

Output:

```
Enter the number of frames :
10
Enter data 1:
h
Enter data 2:
e
ack 2
Enter data 3:
l
ack 3
Enter data 4:
l
Enter data 5:
o
Enter data 6:
u
```

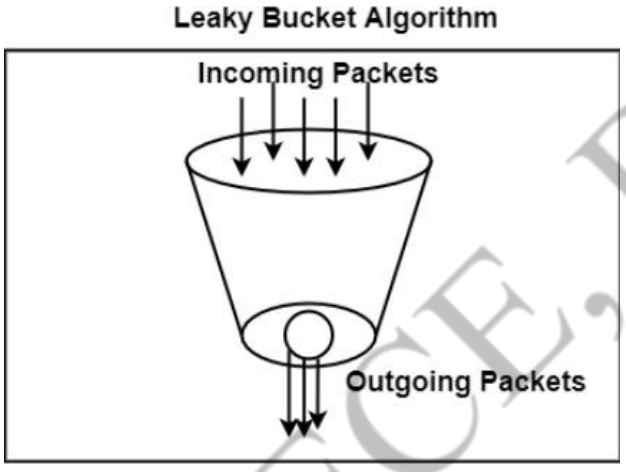
nak 5
Resend data 5 :
o
Enter data 7:
o
Enter dara 8:
r
Enter data 9:
l
Enter data 10:
d
ack 11
End of sliding window protocol

Conclusion/Inference: (to be written by students)

EXPERIMENT – 6

Write a c program for congestion control Using Leaky Bucket Algorithm

Theory: The leaky bucket algorithm is used in the context of network traffic shaping or rate-limiting. The algorithm allows controlling the rate at which a record is injected into a network and managing burstiness in the data rate. A leaky bucket execution and a token bucket execution are predominantly used for traffic shaping algorithms. This algorithm is used to control the rate at which traffic is sent to the network and shape the burst traffic to a steady traffic stream. The figure shows the leaky bucket algorithm.



In this algorithm, a bucket with a volume of, say, b bytes and a hole in the Notes bottom is considered. If the bucket is null, it means b bytes are available as storage. A packet with a size smaller than b bytes arrives at the bucket and will forward it. If the packet's size increases by more than b bytes, it will either be discarded or queued. It is also considered that the bucket leaks through the hole in its bottom at a constant rate of r bytes per second.

The outflow is considered constant when there is any packet in the bucket and zero when it is empty. This defines that if data flows into the bucket faster than data flows out through the hole, the bucket overflows. The leaky bucket algorithm is also used in leaky bucket counters, e.g. to detect when the average or peak rate of random or stochastic events or stochastic processes, such as faults or failures, exceed defined limits.

Algorithm:

Step1: Initialize bucket size, outgoing rate and counter initial value.
Step2: Enter incoming packet size, if packet size is greater than bucket size, drop packets.
Step3: Packets are output out of bucket as per outgoing rate.
Step4: Repeat process by initializing counter, incoming packets are added to remaining packets in bucket in next cycle.

C Code:

```
#include<stdio.h>
#include<unistd.h>
int main()
{
int incoming,outgoing,bucksize,initialval,count,ans,store=0;
printf("enter bucket size, outgoing rate and counter initial size:");
scanf("%d %d %d", &bucksize, &outgoing, &initialval);
count=initialval;
while(count!=0)
{
printf("enter the incoming packet size:");
scanf("%d", &incoming);
printf("incoming packet size %d\n",incoming);
if(incoming<=(bucksize-store))
{
store+=incoming;
printf("bucket buffer size %d out of %d\n",store,bucksize);
}
else
 {
printf(" dropped %d no of packets\n", incoming-(bucksize-store));
store=bucksize;
printf(" bucket buffer size %d out of %d\n",store,bucksize);
}
store=store-outgoing;
sleep(5);
printf(" after outgoing %d pkts left out of %d in buffer\n",store,bucksize);
if(count<=incoming)
{    printf("do u want to continue? type 1 to continue");
scanf("%d",&ans);
if(ans!=1)
break;
count=initialval;
}
count=count-incoming;
printf("count=%d\n",count);
}
}
```

Output:

```
Enter bucket size, outgoing rate and counter initial size: 15 10 40
Enter the incoming packet size: 20
Incoming packet size 20
Dropped 5 no. of packets
Bucket buffer size 15 out of 15
After outgoing 5 pkts left out of 15 in buffer
Count = 20
```

Enter the incoming packet size: 15
Incoming packet size 15
Dropped 5 no. of packets
Bucket buffer size 15 out of 15
After outgoing 5 pkts left out of 15 in buffer
Count = 5

Enter the incoming packet size: 5
Incoming packet size 5
Dropped 5 no. of packets
Bucket buffer size 10 out of 15
After outgoing 0 pkts left out of 15 in buffer
Do you want to continue? Type 1 to continue 1
Count = 35
Enter the incoming packet size: 40
Incoming packet size 40
Dropped 25 no. of packets
Bucket buffer size 15 out of 15
After outgoing 5 pkts left out of 15 in buffer
Do you want to continue? Type 1 to continue……

Conclusion/Inference: (to be written by students)

Computer Networks Lab VIVA Questions:-

What is Network?
A network is a set of devices connected by physical media links. A network is recursively is a connection of two or more nodes by a physical link or two or more networks connected by one or more nodes.

What is a Link?
At the lowest level, a network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Link.

What is a node?
A network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Links and the computer it connects is called as Nodes.

What is a gateway or Router?
A node that is connected to two or more networks is commonly called as router or Gateway. It generally forwards message from one network to another.

What is point-point link?
If the physical links are limited to a pair of nodes it is said to be point-point link.

What is Multiple Access?
If the physical links are shared by more than two nodes, it is said to be Multiple Access.

What are the criteria necessary for an effective and efficient network?
a. Performance: It can be measured in many ways, including transmit time and response time. b. Reliability: It is measured by frequency of failure, the time it takes a link to recover from a failure, and the networks robustness.
c. Security: Security issues include protecting data from unauthorized access and virus.

Name the factors that affect the performance of the network? a. Number of Users
b. Type of transmission medium c. Hardware
d. Software

Name the factors that affect the reliability of the network? a. Frequency of failure
b. Recovery time of a network after a failure

What is Protocol?
A protocol is a set of rules that govern all aspects of information communication

What are the key elements of protocols?
The key elements of protocols are
a. Syntax: It refers to the structure or format of the data that is the order in which they are presented.
b. Semantics: It refers to the meaning of each section of bits.
c. Timing: Timing refers to two characteristics: When data should be sent and how fast they can be sent.

What are the key design issues of a computer Network?
a. Connectivity b. Cost-effective Resource Sharing c. Support for common Services d. Performance

Define Bandwidth and Latency?
Network performance is measured in Bandwidth (throughput) and Latency (Delay). Bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. Latency corresponds to how long it t5akes a message to travel from one end of a network to the other. It is strictly measured in terms of time.

Define Routing?
The process of determining systematically hoe to forward messages toward the destination nodes based on its address is called routing.

What is a peer-peer process?
The processes on each machine that communicate at a given layer are called peer-peer process.

When is a switch said to be congested?
It is possible that a switch receives packets faster than the shared link can accommodate and stores in its memory, for an extended period of time, then the switch will eventually run out of buffer space, and some packets will have to be dropped and in this state is said to congested state.

What is Round Trip Time?
The duration of time it takes to send a message from one end of a network to the other and back, is called RTT.

Define the terms Unicasting, Multicasting and Broadcasting?

If the message is sent from a source to a single destination node, it is called Unicasting. If the message is sent to some subset of other nodes, it is called Multicasting. If the message is sent to all the m nodes in the network it is called Broadcasting.

List the layers of OSI
a. Physical Layer b. Data Link Layer c. Network Layer d. Transport Layer e. Session Layer
f. Presentation Layer g. Application Layer

List the layers of TCP/IP model
a. Physical Layer b. Data Link Layer c. Network Layer d. Transport Layer e. Application Layer

Which layers are network support layers?
a. Physical Layer b. Data link Layer and c. Network Layers

Which layers are user support layers?
a. Session Layer b. Presentation Layer and c. Application Layer

Which layer links the network support layers and user support layers?
The Transport layer links the network support layers and user support layers.

What are the concerns of the Physical Layer?
Physical layer coordinates the functions required to transmit a bit stream over a physical medium.
a. Physical characteristics of interfaces and media
b. Representation of bits c. Data rate
d. Synchronization of bits e. Line configuration
f. Physical topology g. Transmission mode

What are the responsibilities of Data Link Layer?
The Data Link Layer transforms the physical layer, a raw transmission facility, to a reliable link and is responsible for node-node delivery.
Framing, Physical Addressing, Flow Control, Error Control and Access Control

What are the responsibilities of Network Layer?
The Network Layer is responsible for the source-to-destination delivery of packet possibly across multiple networks (links), Logical addressing and Routing

What are the responsibilities of Transport Layer?
The Transport Layer is responsible for source-to-destination delivery of the entire message.
a. Service-point Addressing b. Segmentation and reassembly c. Connection Control d. Flow Control e. Error Control

What are the responsibilities of Application Layer?
The Application Layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as e-mail, shared database management and other types of distributed information services, Network virtual Terminal, File transfer, access and Management (FTAM), Mail services, Directory Services

What are the two classes of hardware building blocks?
Nodes and Links.

What are the types of errors?
a. Single-Bit error
In a single-bit error, only one bit in the data unit has changed
b. Burst Error
A Burst error means that two or more bits in the data have changed.

What is Error Detection?
Data can be corrupted during transmission. For reliable communication errors must be deducted and corrected. Error Detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination.

What is Redundancy?
The concept of including extra information in the transmission solely for the purpose of comparison. This technique is called redundancy.

What is CRC?
CRC, is the most powerful of the redundancy checking techniques, is based on binary division.

What is Checksum?
Checksum is used by the higher layer protocols (TCP/IP) for error detection.

List the steps involved in creating the checksum.
a. Divide the data into sections
b. Add the sections together using 1s complement arithmetic
c. Take the complement of the final sum, this is the checksum.

Compare Error Detection and Error Correction:
The correction of errors is more difficult than the detection. In error detection, checks only any error has occurred. In error correction, the exact number of bits that are corrupted and location in the message are known. The number of the errors and the size of the message are important factors.

Define Retransmission?
Re transmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Re sending is repeated until a message arrives that the receiver believes is error-freed.

Define Encoder?
A device or program that uses predefined algorithms to encode, or compress audio or video data for storage or transmission use. A circuit that is used to convert between digital video and analog video.

Define Decoder?
A device or program that translates encoded data into its original format (e.g. it decodes the data). The term is often used in reference to MPEG-2 video and sound data, which must be decoded before it is output.

What is framing?
Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet has to go and the sender address helps the recipient acknowledge the receipt.

What is Fixed Size Framing?
In fixed-size framing, there is no need for defining the boundaries of the frames. The size itself can be used as a delimiter.

Define Character Stuffing?
In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

What is Bit Stuffing?
Bit stuffing is the process of adding one extra 0 whenever five consecutive is follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

What is Flow Control?
Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

What is Error Control?
Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and re transmission.

What Automatic Repeat Request (ARQ)?
Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

What is Stop-and-Wait Protocol?
In Stop and wait protocol, sender sends one frame, waits until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.

What is Stop-and-Wait Automatic Repeat Request?
Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

What is usage of Sequence Number in Reliable Transmission?
The protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame. Since we want to minimize the frame size, the smallest range that provides unambiguous communication. The sequence numbers can wrap around.

What is Pipelining?
In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining.

What is Sliding Window?
The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, he sender and receiver need to deal with only part of the possible sequence numbers.

What is Piggy Backing?
A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

What are the two types of transmission technology available?
(i) Multipoint (ii) point-to-point

What is subnet?
A generic term for section of a large networks usually separated by a bridge or router.

What are the possible ways of data exchange?
(i) Simplex (ii) Half-duplex (iii) Full-duplex.

Define simulation?

What is the full form of NS-2?

What are the languages used in NS-2?