

# **ATME COLLEGE OF ENGINEERING**

**13<sup>th</sup>KM Stone, Bannur Road, Mysore - 570028**



**A T M E**  
**College of Engineering**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**(ACADEMIC YEAR 2024-25)**

## **LABORATORY MANUAL**

**SUBJECT: COMPUTER NETWORKS**

**SUBJECT CODE:BCS502**

**SEMESTER:V**

**Outcome Based Education (OBE) and Choice Based Credit System (CBCS)**

**(Effective from the academic year 2021-22)**

**Composed by**

**Verified by**

**Approved by**

**Mr. Rajiv P**  
**PROGRAMMER**

**Dr. SUNITHA PATEL M S & Mr. RAGHURAM A S**  
**FACULTY CO-ORDINATOR**

**Dr. PUTTEGOWDA D**  
**HOD, CSE**

# **INSTITUTIONAL MISSION AND VISION**

## **Objectives**

- To provide quality education and groom top-notch professionals, entrepreneurs and leaders for different fields of engineering, technology and management.
- To open a Training-R & D-Design-Consultancy cell in each department, gradually introduce doctoral and postdoctoral programs, encourage basic & applied research in areas of social relevance, and develop the institute as a center of excellence.
- To develop academic, professional and financial alliances with the industry as well as the academia at national and transnational levels.
- To cultivate strong community relationships and involve the students and the staff in local community service.
- To constantly enhance the value of the educational inputs with the participation of students, faculty, parents and industry.

## **Vision**

- Development of academically excellent, culturally vibrant, socially responsible and globally competent human resources.

## **Mission**

- To keep pace with advancements in knowledge and make the students competitive and capable at the global level.
- To create an environment for the students to acquire the right physical, intellectual, emotional and moral foundations and shine as torch bearers of tomorrow's society.
- To strive to attain ever-higher benchmarks of educational excellence.

## **Department of Computer Science & Engineering**

### **Vision of the Department**

- To develop highly talented individuals in Computer Science and Engineering to deal with real world challenges in industry, education, research and society.

### **Mission of the Department**

- To inculcate professional behavior, strong ethical values, innovative research capabilities and leadership abilities in the young minds & to provide a teaching environment that emphasizes depth, originality and critical thinking.
- Motivate students to put their thoughts and ideas adoptable by industry or to pursue higher studies leading to research.

### **Program outcomes (POs)**

**Engineering Graduates will be able to:**

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage

#### **Program Educational Objectives (PEO'S):**

1. Empower students with a strong basis in the mathematical, scientific and engineering fundamentals to solve computational problems and to prepare them for employment, higher learning and R&D.
2. Gain technical knowledge, skills and awareness of current technologies of computer science engineering and to develop an ability to design and provide novel engineering solutions for software/hardware problems through entrepreneurial skills.
3. Exposure to emerging technologies and work in teams on interdisciplinary projects with effective communication skills and leadership qualities.
4. Ability to function ethically and responsibly in a rapidly changing environment by applying innovative ideas in the latest technology, to become effective professionals in Computer Science to bear a life-long career in related areas.

### **Program Specific Outcomes (PSOs)**

1. **PSO1:** Ability to apply skills in the field of algorithms, database design, web design, cloud computing and data analytics.
2. **PSO2:** Apply knowledge in the field of computer networks for building network and internet based applications.

## COMPUTER NETWORKS

Subject Code	:	BCS502	CIE Marks	:	50
Teaching Hours/Week (L:T:P: S)	:	3:0:2:0	SEE Marks	:	50
Total Hours of Pedagogy	:	40 hours Theory + 8-10 Lab Slots	Total Marks	:	100
Credits	:	04	Exam Hours	:	03
Examination type (SEE)		Theory/Practical			

### Course objectives:

This course will enable students to,

- Study the TCP/IP protocol suite, switching criteria and Medium Access Control protocols for reliable and noisy channels.
- Learn network layer services and IP versions.
- Discuss transport layer services and understand UDP and TCP protocols.
- Demonstrate the working of different concepts of networking layers and protocols

### Teaching-Learning Process (General Instructions)

These are sample Strategies; that teachers can use to accelerate the attainment of the various course outcomes.

1. Lecturer method (L) need not to be only traditional lecture method, but alternative effective teaching methods could be adopted to attain the outcomes.
2. Use of Video/Animation to explain functioning of various concepts.
3. Encourage collaborative (Group Learning) Learning in the class.
4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.
5. Adopt Problem Based Learning (PBL), which fosters student's Analytical skills, develop design thinking skills such as the ability to design, evaluate, generalize, and analyze information rather than simply recall it

## PRACTICAL COMPONENT OF IPCC

### EXPERIMENTS

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.
2. Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Develop a program for error detecting code using CRC-CCITT (16- bits).
5. Develop a program to implement a sliding window protocol in the data link layer.
6. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.

7. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present
8. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side. Develop a JAVA program to raise a custom exception (user defined exception) for DivisionByZero using try, catch, throw and finally.
9. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10. Develop a program for congestion control using a leaky bucket algorithm.

### **Course outcomes (Course Skill Set):**

At the end of the course, the student will be able to:

- **Explain** the fundamentals of computer networks.
- **Apply** the concepts of computer networks to demonstrate the working of various layers and protocols in communication network.
- **Analyze** the principles of protocol layering in modern communication systems.
- **Demonstrate** various Routing protocols and their services using tools such as Cisco packet tracer.

**Note:** For the Simulation experiments modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude using NS2 or NS3. Installation procedure of the required software must be demonstrated, carried out in groups, and documented in the report. Non simulation programs can be implemented using Java.

### **Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal valuation) and SEE (Semester End Examination) taken together.

### **CIE for the theory component of the IPCC (maximum marks 50)**

- IPCC means practical portion integrated with the theory of the course.
- CIE marks for the theory component are 25 marks and that for the practical component is 25 marks.
- 25 marks for the theory component are split into 15 marks for two Internal Assessment Tests (Two Tests, each of 15 Marks with 01-hour duration, are to be conducted) and 10 marks for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.
- Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for 25 marks).
- The student has to secure 40% of 25 marks to qualify in the CIE of the theory component of IPCC. CIE for the practical component of the IPCC
- 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.

- On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.
- The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.
- Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks.
- The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

### **SEE for IPCC**

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course (duration 03 hours)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), should have a mix of topics under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.

### **Suggested Learning Resources:**

Textbook:

1. Behrouz A. Forouzan, Data Communications and Networking, 5th Edition, Tata McGraw Hill, 2013.

### **Reference Books:**

1. Larry L. Peterson and Bruce S. Davie: Computer Networks – A Systems Approach, 4th Edition, Elsevier, 2019.
2. Nader F. Mir: Computer and Communication Networks, 2nd Edition, Pearson Education, 2015.
3. William Stallings, Data and Computer Communication 10th Edition, Pearson Education, Inc., 2014.

### **Web links and Video Lectures (e-Resources):**

1. <https://www.digimat.in/nptel/courses/video/106105183/L01.html>
2. <http://www.digimat.in/nptel/courses/video/106105081/L25.html>
3. <https://nptel.ac.in/courses/1061>

### **Activity Based Learning (Suggested Activities in Class)/ Practical Based learning**

- Implementation of various protocols using open source simulation tools. (5 marks)
- Simulation of Personal area network, Home area network, achieve QoS etc. (5 marks)



## CONTENT LIST

<b><u>SL.NO.</u></b>	<b><u>EXPERIMENT NAME</u></b>	<b><u>PAGE NO.</u></b>
1.	<b>Introduction to NS-2</b>	1-17
2.	<b>PROGRAM 1:</b> Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.	18-20
3.	<b>PROGRAM 2:</b> Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.	21-24
4.	<b>PROGRAM 3:</b> Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.	25-28
5.	<b>PROGRAM 4:</b> Develop a program for error detecting code using CRC-CCITT (16- bits).	29-33
6.	<b>PROGRAM 5:</b> Develop a program to implement a sliding window protocol in the data link layer.	34-38
7.	<b>PROGRAM 6:</b> Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.	39-42
8.	<b>PROGRAM 7:</b> Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present	43-49
9.	<b>PROGRAM 8:</b> Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.	50-54
10.	<b>PROGRAM 9:</b> Develop a program for a simple RSA algorithm to encrypt and decrypt the data.	55-59
11.	<b>PROGRAM 10:</b> Develop a program for congestion control using a leaky bucket algorithm.	60-63
12.	<b>VIVA QUESTIONS AND ANSWERS</b>	64-68

## Introduction

### Introduction to NS-2

NS-2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks. Network Simulator-2 (NS2) is a popular open source network simulator for carrying out network experimentation. Way back when it was being designed, its primary usage was to analyze the performance of congestion control algorithms implemented in Transmission control protocol (TCP).

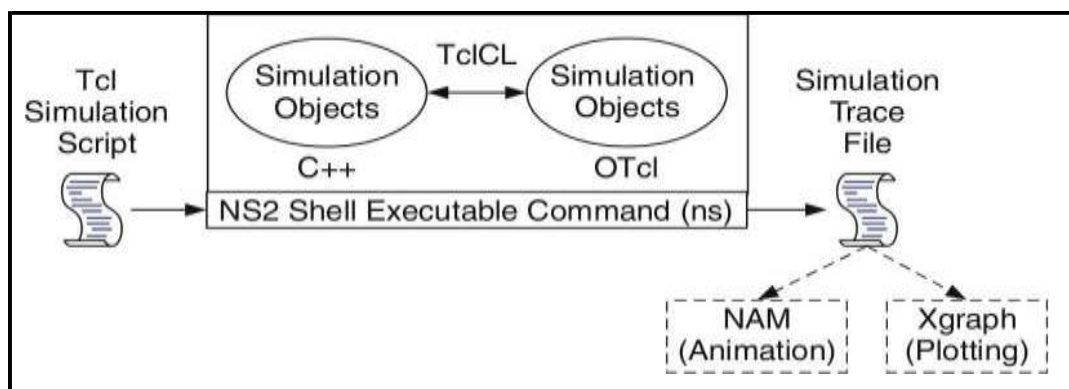
Even today, it remains the most widely

used network simulator for TCP research. Over the period of time, it gained wide acceptance in industry, and now supports simulation of latest wired as well as wireless networking protocols (e.g., routing algorithms, TCP, User Data Protocol (UDP) ) and paradigms such as Mobile Ad hoc Networks (MANETs) Vehicular Ad hoc Network (VANETs), etc.

Another simulator called ns-3 has gained a lot of popularity in the recent past. It is not a sequel of NS-2. NS-3 APIs are not compatible with those of NS-2 API. Both are completely different tools.

#### Features of NS-2:

- It is a discrete event simulator for networking research.
- It provides substantial support to simulate protocols like TCP, FTP, UDP & DSR.
- It simulates wired and wireless network.
- It is primarily UNIX based.
- Uses TCL as its scripting language.
- Otcl: Object oriented support Tcl
- TclCL: Tcl with Classes and OTcl linkage
- Discrete event scheduler



**Fig. 1 Basic Architecture of Network Simulator**

---

**Why two languages? (TCL and C++)**

- NS2 consists of two key languages: C++ and Object-oriented Tool Command Language(OTcl).
- The C++ defines the internal mechanism (i.e., a backend) of the simulation objects.
- The OTcl sets up simulation by configuring the objects as well as scheduling discrete events (i.e., a frontend).
- The C++ and the OTcl are linked together using TclCL.
- NS2 uses OTcl to create and configure a network, and uses C++ to run simulation.
- C++ is fast to run but slow to change.
- OTcl, on the other hand, is slow to run but fast to change.
- We write a Tcl simulation script and feed it as an input argument to NS2 when running simulation (e.g., executing “nsmyfirst\_ns.tcl”).
- Here, “ns” is a C++ executable file obtained from the compilation.
- myfirst\_ns.tcl is an input configuration file specifying system parameters and configuration such as nodes, link, and how they are connected.
- C++ is used for the creation of objects because of speed and efficiency.
- OTcl is used as a front-end to setup the simulator, configure objects and schedule event because of its ease of use.

**Tcl scripting**

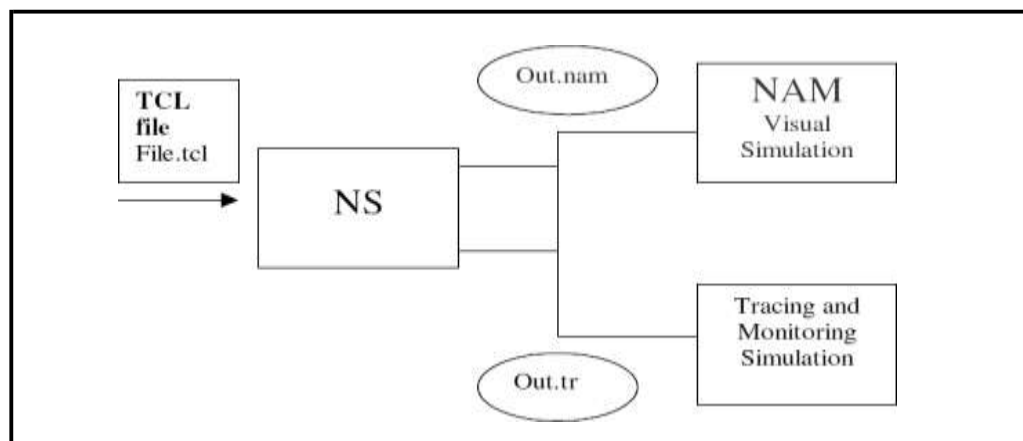
- Tcl is a general purpose scripting language.[Interpreter]
- Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- The strength of Tcl is its simplicity.
- It is not necessary to declare a data type for variable prior to the usage

**Structure of NS-2 Program:**

- Creating a Simulator Object
- Setting up files for trace & NAM
- Tracing files using their commands
- Closing trace file and starting NAM
- Creating LINK & NODE topology & Orientation of links

**Working of NS-2**

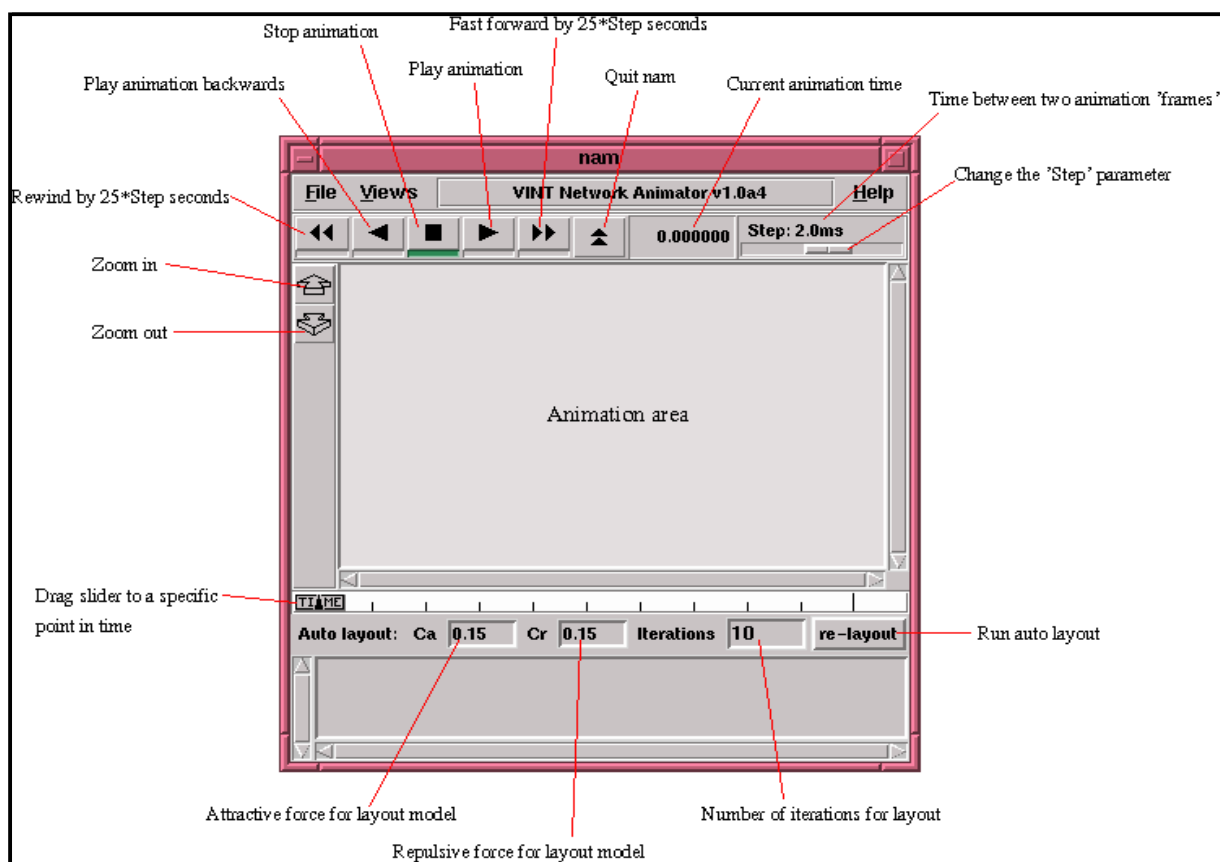
- NS2 provides users with executable command ns which takes an input argument, the name of a Tcl simulation scripting file.
- Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns.
- In most cases, a simulation trace file is created, and is used to plot graph and/or to create animation.



**Fig 2. Working of Network Simulator 2**

### Trace file and NamTrace file:

- Once the simulation is complete, we can see two files: “trace.tr”, and “nam.out”.
- The trace file (trace.tr) is a standard format used byns2.
- In ns2, each time a packet moves from one node to another, or onto a link, or into a buffer, etc., it gets recorded in this trace file.
- Each row represents one of these events and each column has its own meaning.
- Start nam with the command 'nam <nam-file>' where '<nam-file>' is the name of a nam trace file that was generated byns.



**Network Animator Window**

**Fig. 3 Details of NAM Window**

## Advantages and Disadvantages of NS2:

### Advantages

1. OpenSource
2. Complex scenarios can be easily tested.
3. Results can be quickly obtained – more ideas can be tested in a smaller timeframe.
4. Supported protocols
5. Supported platforms
6. Modularity

### Disadvantages

1. Limitation in designing large scale systems
2. May be slow compared to real time network and computationally expensive
3. Does not reflect reality in large and complex networks.
4. Statistical uncertainty in results

### Steps to create a scenario file:

**Step1:** Declare Simulator and setting output file

**Step2:** Setting Node and Link

**Step3:** Setting Agent

**Step4:** Setting Application

**Step5:** Setting Simulation time and schedules

**Step6:** Declare finish.

### **Step 1: Declare Simulator and setting**

\$ns [new Simulator]	#first line of tcl script. Creates ns object
get file [open out.tr w] \$ns trace-all \$file	#open the trace file
get namfile [open out.nam w] \$ns namtrace-all \$namfile	#open the nam file

### **Step 2: Setting Node and Link**

\$ n0 [\$ns node]	setting a node
ns duplex-link \$n0 \$n2 3Mb 5ms DropTail	#bidirectional link between n0 and n2 is declared bandwidth 3Mbps and delay 5ms. DropTail is a waiting queue type.
\$ns duplex-link-op \$n0 \$n2 orient right-down	#Sets positions of node and link for Nam. It does not affect to the result of simulation
\$ns queue-limit \$n2 \$n3 20	#The length of queue on the link from n2 to n3 is 20[packets].
\$ns duplex-link-op \$n2 \$n3 queuePos 0.5	#The position of queue is set for Nam, 0.5 is the angle between link and queue, it equals to (0.5_).

**Step 3: Setting Agent**

**UDP Agent :** To use UDP in simulation, the sender sets the Agent as UDP Agent while the receiver sets to Null Agent. Null Agents do nothing except receiving the packets.

set udp [newAgent/UDP] \$ns attach-agent \$n0 \$udp set null [newAgent/Null] \$ns attach-agent \$n3 \$null	#udp and null Agent are set for n0 and n3, respectively
\$ns connect \$udp \$null	Declares the transmission between udp and null.
\$udp set fid_ 0	Sets the number for data flow of udp. This number will be recorded to all packets which are sent from udp
\$ns color 0 blue	Mark the color to discrete packet for showing result on NAM.

**TCP Agent:** To use TCP in simulation, the sender sets the Agent as TCP Agent while the receiver sets to TCPSink Agent. When receiving a packet, TCPSink Agent will reply an acknowledgment packet (ACK). Setting Agent for TCP is similar to UDP.

set tcp [new Agent/TCP] \$ns attach-agent \$n1 \$tcp set sink [new Agent/TCPSink] \$ns attach-agent \$n3 \$sink	# tcp and sink Agent are set for n1 and n3, respectively
\$ns connect \$tcp \$sink	#declares the transmission between tcp and sink.
\$tcp set fid_ 1	#sets the number for data flow of tcp. This number will be recorded to all packet which are sent from tcp
\$ns color 1 red	#mark the color to discrete packet for showing result on Nam.

**Step 4: Setting Application**

In general, UDP Agent uses CBR Application while TCP Agent uses FTP Application.

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

**Step 5: Setting time schedule for simulation**

Time schedule of a simulation is set as below:

\$ns at 1.0 "\$cbrstart" \$ns at 3.5 "\$cbrstop"	cbr transmits data from 1.0[sec] to 3.5[sec]
\$ns at 1.5 "\$ftpstart" \$ns at 3.0 "\$ftpstop"	ftp transmits data from 1.5[sec] to 3.0[sec].

**Step 6: Declare finish**

After finish setting, declaration of finish is written at the end of file.

\$ns at 4.0 "finish"	The finish function is used to output data file at the end of simulation.
<pre> proc finish {} { global ns file namfile tcpfile \$ns flush-trace close \$file close \$namfile close \$tcpfile exit 0 } </pre>	

### Execute Simulation and start Nam

By executing below command line, simulation will be started and shows the animation of simulation

**ns sample.tcl**  
**nam out.nam**

### View trace file (out.tr)

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id
<pre> 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) </pre>											
<pre> 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 </pre>											
Trace Format Example											

**Figure 4 . Details of Trace Window**

1. The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (eg CBR or TCP)
6. Gives the packet size
7. Some flags
8. This is the flow id (fid) of IPv6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAMdisplay.
9. This is the source address given in the form of 'node.port'.
10. This is the destination address, given in the same form.
11. This is the network layer protocol's packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purposes
12. The last field shows the Unique id of the packet.

**AWK file:**

- The basic function of awk is to search files for lines (or other units of text) that contain certain patterns. When a line matches one of the patterns, awk performs specified actions on that line. awk keeps processing input lines in this way until the end of the input files are reached.
- Programs in awk are different from programs in most other languages, because awk programs are **data-driven**; that is, we describe the data to work with, and then what to do when we find it. Most other languages are **procedural**. When working with procedural languages, it is usually much harder to clearly describe the data of our program will process.
- For this reason, awk programs are often refreshingly easy to both write and read. When we run awk, we can specify an awk **program** that tells awk what to do. The program consists of a series of **rules**. (It may also contain **function definitions**, an advanced feature which we will ignore for now.
- Each rule specifies one pattern to search for, and one action to perform when that pattern is found). Syntactically, a rule consists of a pattern followed by an action. The action is enclosed in curly braces to separate it from the pattern. Rules are usually separated by newlines. Therefore, an awk program looks like this:

```
pattern { action }           pattern { action }
```

- Since we are dealing with column oriented data, AWK is probably the easiest tool we can use to format our data. AWK is a simple scripting language that scans through a file line byline. It allows to access any column in the current line by using special variables \$1, \$2, \$3, etc. for the first, second and third columns. The definition of each column of the trace file is shown above, so we can use the AWK script to check the value of each column and collect the data we need.
- The BEGIN and END sections are only executed once (before and after the file has been processed). The middle section is executed for each line of the file. The AWK script keeps three variables to store the throughput in Mb/s for flow 1, flow 2, and the total. For each line of the trace file, it checks to see if a TCP packet (\$5 == "tcp") is received (\$1 == "r") at node 3 (\$4 == "3"), which is our destination node. If so, it increments the count for the appropriate flow, using the size of the particular packet (in bytes) from column 6. After each second, it prints the total while converting bytes to Mb.

```
BEGIN { print "START" } { print } END { print "STOP" }
```

**To run awk script**

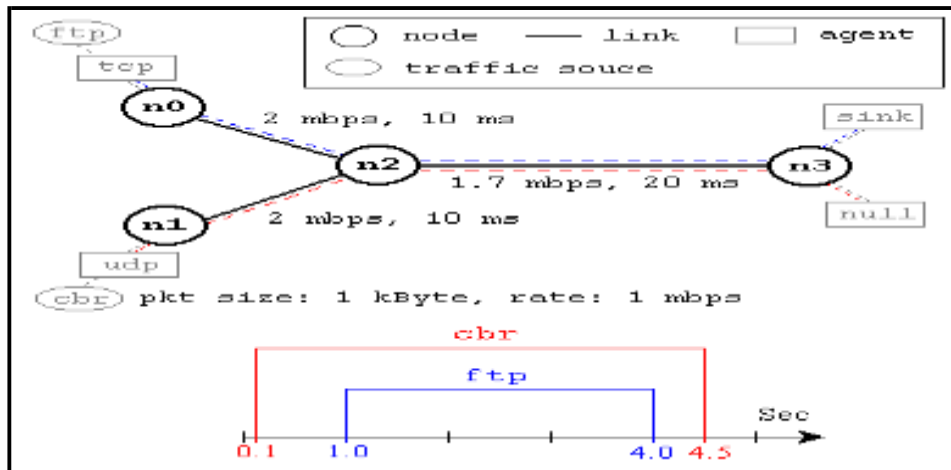
```
awk -f <filename.awk> <input_file> <output_file>
```

**XGRAPH:**

- Plotting purposes.
- Comes together with NS2 installation package.
- Running Xgraph
- 

```
Xgraph <inputfile1>...<inputfileN> -bg <color> -t <graph_title> -x <xtitle> -y <ytitle>
```



**NS Simulation Script****Fig 5. A Simple Network Topology and Simulation Scenario**

This network consists of 4 nodes (n0, n1, n2, n3) as shown in Fig. 5. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte.

A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate 1KByte packets at the rate of 1 Mbps.

The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec.

**An Example Simulation Script****#Create a simulator object**

```
set ns [new Simulator]
```

**#Define different colors for data flows (for NAM)**

```
$ns color 1 Blue
```

```
$ns color 2 Red
```

**#Open the NAM trace file**

```
set nf [open out.nam w]
```

```
$ns namtrace-all $nf
```

**#Define a 'finish' procedure**

```
proc finish {} {
```

```
    global ns nf
```

```
$ns flush-trace
```

**#Close the NAM trace file**

```
close $nf
```

**#Execute NAM on the trace file**

```
exec nam out.nam & exit 0
```

```
}
```

**#Create four nodes**

```
set n0 [$ns node]
```

---

```
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail
#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
#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
#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
#Setup a UDP connection
set udp [newAgent/UDP]

$ns attach-agent $n1 $udp set
null [newAgent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDPconnection
set cbr [newApplication/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftpstart"
$ns at 4.0 "$ftpstop"
$ns at 4.5 "$cbrstop"
$ns at 5.0 "finish"
#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]" puts "CBR interval = [$cbr set interval_]"
#Run the simulation
$ns run
```

---

## How to Download and Install NetBeans IDE on your Computer.

Go to <https://netbeans.org/downloads> to download the latest version of NetBeans IDE.

You will see the following page:



On this download page you see different download bundles. And for Java development only, we can choose either Java SE or Java EE. We'd recommend you to choose Java EE which supports comprehensive Java development (Java EE includes Java SE).

So click the Download button in the column Java EE to download NetBeans installer for Java EE development. The file name of the installer program is something like netbeans-8.2-javaee-windows.exe (on Windows).

Click on the installer file to start installing NetBeans IDE. You will be asked to install GlassFish and Apache Tomcat server



In this tutorial, you don't need any server. However you will need them later so let check both, and **click Next**.

In the next screen, **check 'I accept the terms in the license agreement'**:

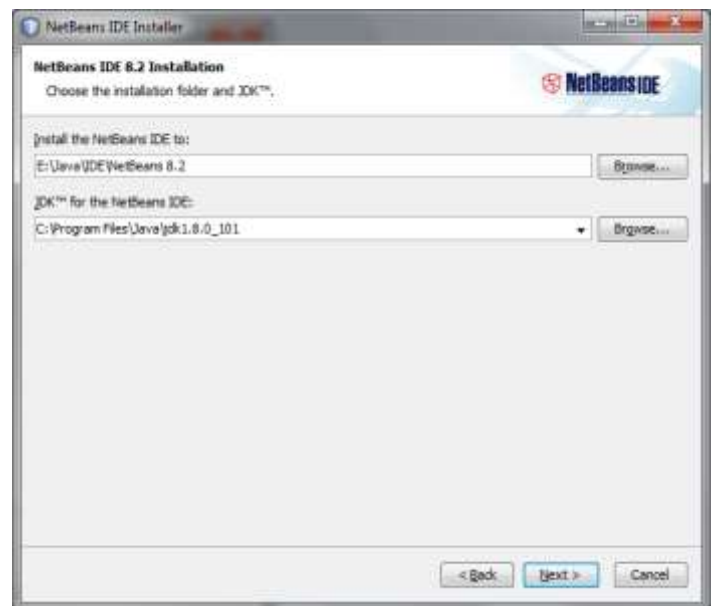


**Click Next.**

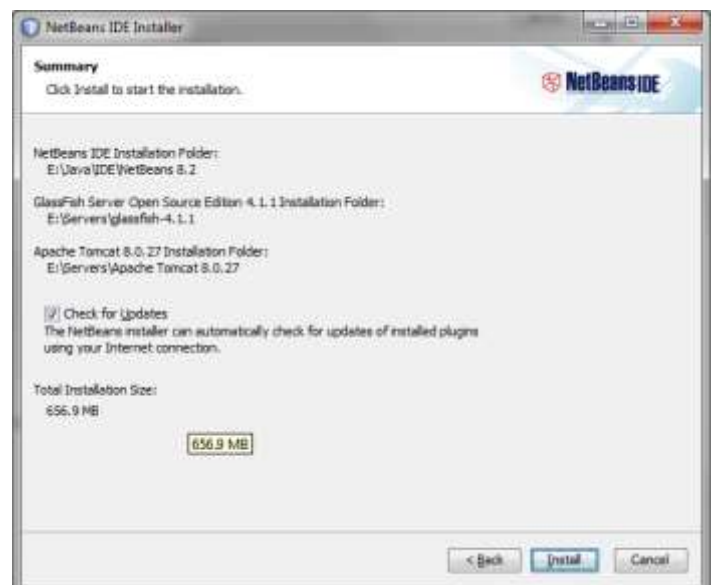
In the next screen, choose the installation directory and JDK version for the IDE:

You can keep the defaults and click **Next**.

In the next screen, choose installation directory and JDK version for GlassFish server:



Click **Next** to see the summary:



And click **Install** to start installing NetBeans with GlassFish and Tomcat servers. Wait until the setup complete:

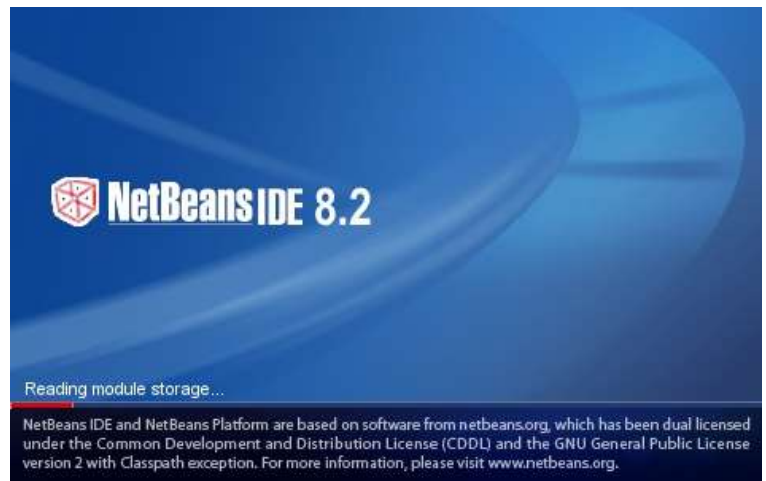


**Click Finish.**

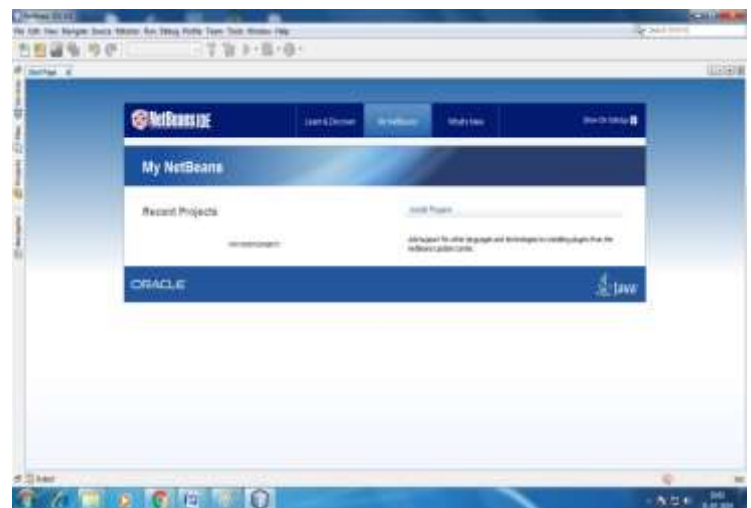
Now you can start NetBeans IDE from the start menu or Run NetBeans by clicking the NetBeans icon on desktop as shown below



The splash screen appears:

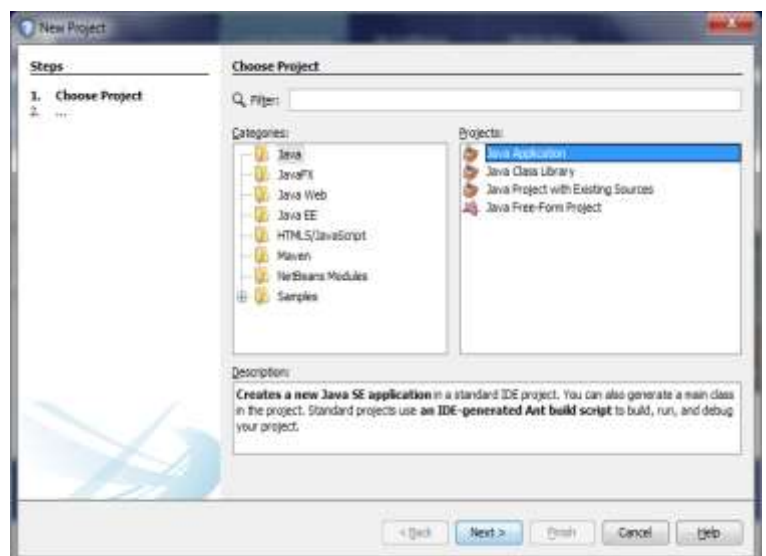


And you should see the home screen of NetBeans:

**How to Create Your First Java Project**

Now, let's create a Java project using NetBeans IDE. Go to menu **File > New Project...**

Under the New Project dialog, choose Java application as shown in the following screenshot:



New Java Application

Steps

1. Choose Project
2. Name and Location

Name and Location

Project Name: HelloWorld

Project Location: E:\Java Browse...

Project Folder: E:\Java\HelloWorld

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class: net.codejava.helloWorld

Back Next Finish Cancel Help

The screenshot shows the 'New Java Application' dialog box in the Eclipse IDE. The 'Name and Location' tab is active. The 'Project Name' field contains 'HelloWorld'. The 'Project Location' field contains 'E:\java', and the 'Project Folder' field contains 'E:\java\HelloWorld'. There are 'Browse...' buttons next to the 'Project Location' and 'Libraries Folder' fields. The 'Use Dedicated Folder for Storing Libraries' checkbox is unchecked. Below it, a note states: 'Different users and projects can share the same compilation libraries (see Help for details)'. The 'Create Main Class' checkbox is checked, and the 'Main Class' field contains 'test.code.java.helloWorld'. At the bottom, there are buttons for 'Back', 'Next >', 'Finish' (highlighted), 'Cancel', and 'Help'.

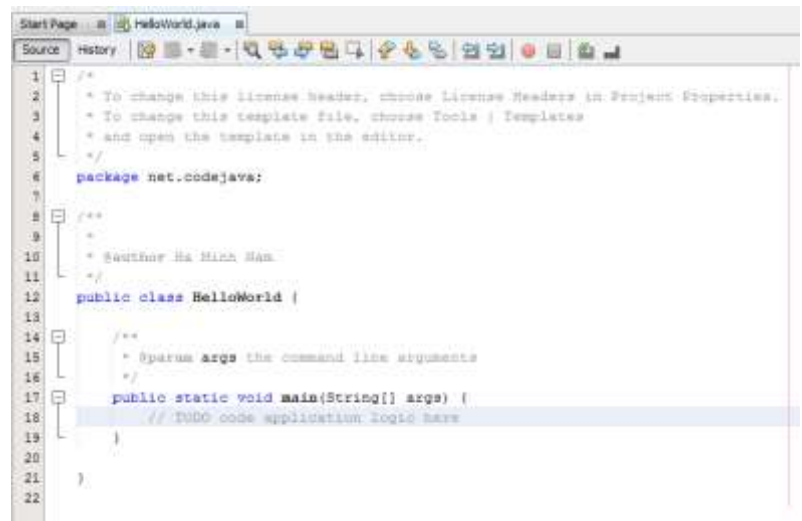
```

1  // To change this license header, choose License Header in Project Properties.
2  // To change this template file, choose Template | Configure...
3  // and open the template in the editor.
4
5  package net.codestars;
6
7
8  import java.util.*;
9
10
11
12  public class Ballometer {
13
14      //
15      //
16      //
17      //
18      //
19      //
20      //
21      //
22      //
23      //
24      //
25      //
26      //
27      //
28      //
29      //
30      //
31      //
32      //
33      //
34      //
35      //
36      //
37      //
38      //
39      //
40      //
41      //
42      //
43      //
44      //
45      //
46      //
47      //
48      //
49      //
50      //
51      //
52      //
53      //
54      //
55      //
56      //
57      //
58      //
59      //
60      //
61      //
62      //
63      //
64      //
65      //
66      //
67      //
68      //
69      //
70      //
71      //
72      //
73      //
74      //
75      //
76      //
77      //
78      //
79      //
80      //
81      //
82      //
83      //
84      //
85      //
86      //
87      //
88      //
89      //
90      //
91      //
92      //
93      //
94      //
95      //
96      //
97      //
98      //
99      //
100     //
101     //
102     //
103     //
104     //
105     //
106     //
107     //
108     //
109     //
110     //
111     //
112     //
113     //
114     //
115     //
116     //
117     //
118     //
119     //
120     //
121     //
122     //
123     //
124     //
125     //
126     //
127     //
128     //
129     //
130     //
131     //
132     //
133     //
134     //
135     //
136     //
137     //
138     //
139     //
140     //
141     //
142     //
143     //
144     //
145     //
146     //
147     //
148     //
149     //
150     //
151     //
152     //
153     //
154     //
155     //
156     //
157     //
158     //
159     //
160     //
161     //
162     //
163     //
164     //
165     //
166     //
167     //
168     //
169     //
170     //
171     //
172     //
173     //
174     //
175     //
176     //
177     //
178     //
179     //
180     //
181     //
182     //
183     //
184     //
185     //
186     //
187     //
188     //
189     //
190     //
191     //
192     //
193     //
194     //
195     //
196     //
197     //
198     //
199     //
200     //
201     //
202     //
203     //
204     //
205     //
206     //
207     //
208     //
209     //
210     //
211     //
212     //
213     //
214     //
215     //
216     //
217     //
218     //
219     //
220     //
221     //
222     //
223     //
224     //
225     //
226     //
227     //
228     //
229     //
230     //
231     //
232     //
233     //
234     //
235     //
236     //
237     //
238     //
239     //
240     //
241     //
242     //
243     //
244     //
245     //
246     //
247     //
248     //
249     //
250     //
251     //
252     //
253     //
254     //
255     //
256     //
257     //
258     //
259     //
260     //
261     //
262     //
263     //
264     //
265     //
266     //
267     //
268     //
269     //
270     //
271     //
272     //
273     //
274     //
275     //
276     //
277     //
278     //
279     //
280     //
281     //
282     //
283     //
284     //
285     //
286     //
287     //
288     //
289     //
290     //
291     //
292     //
293     //
294     //
295     //
296     //
297     //
298     //
299     //
300     //
301     //
302     //
303     //
304     //
305     //
306     //
307     //
308     //
309     //
310     //
311     //
312     //
313     //
314     //
315     //
316     //
317     //
318     //
319     //
320     //
321     //
322     //
323     //
324     //
325     //
326     //
327     //
328     //
329     //
330     //
331     //
332     //
333     //
334     //
335     //
336     //
337     //
338     //
339     //
340     //
341     //
342     //
343     //
344     //
345     //
346     //
347     //
348     //
349     //
350     //
351     //
352     //
353     //
354     //
355     //
356     //
357     //
358     //
359     //
360     //
361     //
362     //
363     //
364     //
365     //
366     //
367     //
368     //
369     //
370     //
371     //
372     //
373     //
374     //
375     //
376     //
377     //
378     //
379     //
380     //
381     //
382     //
383     //
384     //
385     //
386     //
387     //
388     //
389     //
390     //
391     //
392     //
393     //
394     //
395     //
396     //
397     //
398     //
399     //
400     //
401     //
402     //
403     //
404     //
405     //
406     //
407     //
408     //
409     //
410     //
411     //
412     //
413     //
414     //
415     //
416     //
417     //
418     //
419     //
420     //
421     //
422     //
423     //
424     //
425     //
426     //
427     //
428     //
429     //
430     //
431     //
432     //
433     //
434     //
435     //
436     //
437     //
438     //
439     //
440     //
441     //
442     //
443     //
444     //
445     //
446     //
447     //
448     //
449     //
450     //
451     //
452     //
453     //
454     //
455     //
456     //
457     //
458     //
459     //
460     //
461     //
462     //
463     //
464     //
465     //
466     //
467     //
468     //
469     //
470     //
471     //
472     //
473     //
474     //
475     //
476     //
477     //
478     //
479     //
480     //
481     //
482     //
483     //
484     //
485     //
486     //
487     //
488     //
489     //
490     //
491     //
492     //
493     //
494     //
495     //
496     //
497     //
498     //
499     //
500     //
501     //
502     //
503     //
504     //
505     //
506     //
507     //
508     //
509     //
510     //
511     //
512     //
513     //
514     //
515     //
516     //
517     //
518     //
519     //
520     //
521     //
522     //
523     //
524     //
525     //
526     //
527     //
528     //
529     //
530     //
531     //
532     //
533     //
534     //
535     //
536     //
537     //
538     //
539     //
540     //
541     //
542     //
543     //
544     //
545     //
546     //
547     //
548     //
549     //
550     //
551     //
552     //
553     //
554     //
555     //
556     //
557     //
558     //
559     //
560     //
561     //
562     //
563     //
564     //
565     //
566     //
567     //
568     //
569     //
570     //
571     //
572     //
573     //
574     //
575     //
576     //
577     //
578     //
579     //
580     //
581     //
582     //
583     //
584     //
585     //
586     //
587     //
588     //
589     //
590     //
591     //
592     //
593     //
594     //
595     //
596     //
597     //
598     //
599     //
600     //
601     //
602     //
603     //
604     //
605     //
606     //
607     //
608     //
609     //
610     //
611     //
612     //
613     //
614     //
615     //
616     //
617     //
618     //
619     //
620     //
621     //
622     //
623     //
624     //
625     //
626     //
627     //
628     //
629     //
630     //
631     //
632     //
633     //
634     //
635     //
636     //
637     //
638     //
639     //
640     //
641     //
642     //
643     //
644     //
645     //
646     //
647     //
648     //
649     //
650     //
651     //
652     //
653     //
654     //
655     //
656     //
657     //
658     //
659     //
660     //
661     //
662     //
663     //
664     //
665     //
666     //
667     //
668     //
669     //
670     //
671     //
672     //
673     //
674     //
675     //
676     //
677     //
678     //
679     //
680     //
681     //
682     //
683     //
684     //
685     //
686     //
687     //
6
```



## Write Your First Java Code

You can see a code editor for the HelloWorld.java file as shown in the following screenshot:

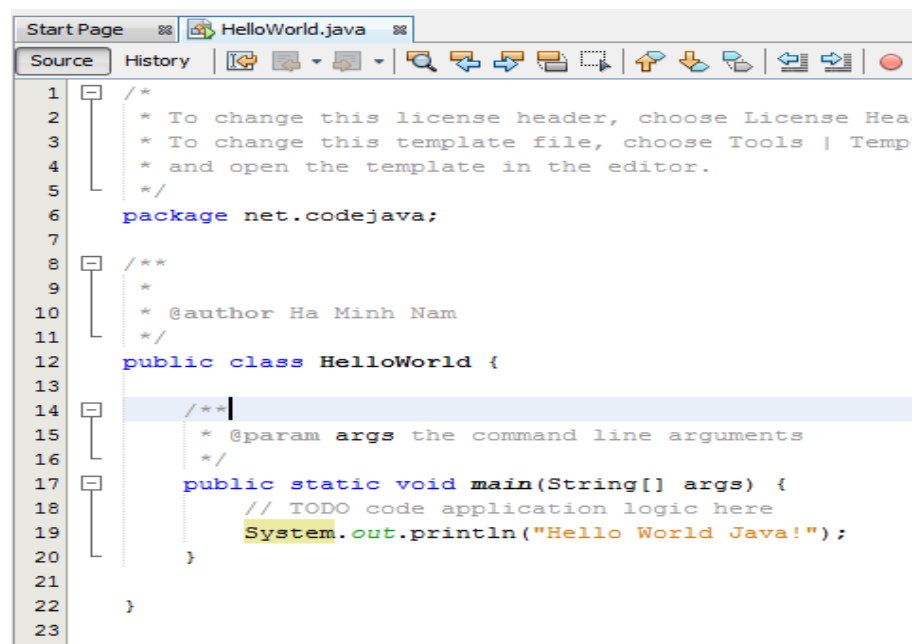


The method main() is the main entry to a Java application. All Java programs start from the main() method. Now, let's type some code in this method to print "Hello World Java!" on the screen:

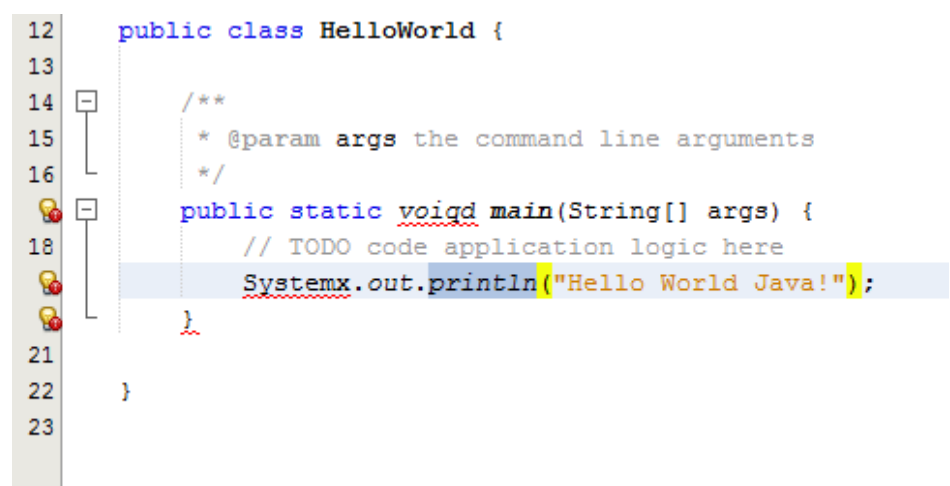
```
1 | System.out.println("Hello World Java!");
```

The whole program should look like this:

NetBeans is very smart, as it compiles the code instantly while you are typing the code. So if there's any error, the IDE will inform you by underlining the errors with red color, as shown in the following screenshot:



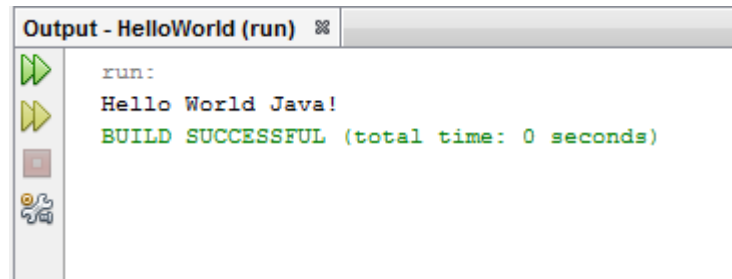
If there's no red marks like this, the code is fine and we're ready to run the program.



### Run Your First Java Program

To run the HelloWorld program above, there are several ways:

- Go to menu **Run > Run Project** <ProjectName>
- Click **Run Project** icon in the toolbar.
- Press **F6** key.
- Right click in the code editor, and select **Run File** (or press **Shift + F6**).



You should see the output of this program like this:

That's it! The HelloWorld program has run and printed the output "Hello World" .

you have successfully created and run your first Java program with NetBeans IDE

## Sample Networking Programs in Java

### UDP SAMPLE PROGRAMS

#### UDP SAMPLE PROGRAM 1

**AIM:** This Program Demonstrates Unidirectional Communication Between Udpclient And Udp Server. Message is sent from Udpclient to UdpServer and message is displayed on UdpServer side

#### UDP Client Program

```
import java.io.*;
import java.net.*;
public class UDPClient1
{
    public static void main(String[] args)
    {
        try
        {
            {ss
                //create client socket
                DatagramSocket clientsocket=new DatagramSocket();
                // get ipaddress of host machine
                InetAddress IPAddress=InetAddress.getByName("localhost");
                // create buffer for sending data
                byte[] outdata=new byte[1024];
                // read input through keyboard
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                System.out.println("enter some text to send it to UDP server");
                String str=br.readLine();
                // convert string to packet data
                outdata=str.getBytes();
```



---

```
DatagramPacket sendpacket=new DatagramPacket(outdata,outdata.length,IPAddress,2222);
    clientsocket.send(sendpacket);
    System.out.println("packet sent to server");
    clientsocket.close();
    }
    catch(Exception ex)
    {
        System.out.println(ex.getMessage());
    }
}
```

### **UDPServerProgram**

```
import java.io.*;
import java.net.*;
public class UDPServer1
{
    public static void main(String[] args)
    {
        try
        {
            // create server object
            DatagramSocket srvobj=new DatagramSocket(2222);
            // array to store data which comes from client
            byte[] InData=new byte[1024];
            System.out.println("server is running");
            while(true)
            {
                // create datagram encapsulated datagram packet
                DatagramPacket receivepacket=new DatagramPacket(InData,InData.length);
                // server receives packet from client
                srvobj.receive((receivepacket));
                String s1=new String(receivepacket.getData());
                System.out.println("data received from client is "+s1);

            }
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

---

```
}
```

## Output

### Step1

#### UDPServer Side

First run UDPServer program to start UDPServer

You will get following message that server is started

**server is running**

### Step2

#### UDPClient Side

then run UDPClient program to send msg from UDPClient TO UDPServer

Below is message when you typed will be sent from UDPClient to UDPServer

**enter some text to send it to UDP server**

**hello atme college of engineering**

**packet sent to server**

### Step3

Finally message is received from UDPClient program to UDPServer program. Message is displayed on UDPServer Program side as follows

**server is running**

**data received from client is hello atme college of engineering.**

---

**Program No. 1: Implement three nodes point – to – point network with duplex links between them for different topologies. 1 Set the queue size, vary the bandwidth and find the number of packets dropped for various iterations.**

**Program Objective:**

- Understand the Implementation of the Duplex link between the network.

**Theory:**

- Create a simulator object.
- We open a file for writing that is going to be used for the trace data.
- We now attach the agent to the nodes.
- Now we attach the application to run on top of these nodes
- We now connect the agent and the application for its working
- Set the simulation time
- The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
set ns [new Simulator]
set ntrace [open prog1.tr w]
$ns trace-all $ntrace
set namfile [open prog1.nam w]
$ns namtrace-all $namfile

proc Finish { } {
    global ns ntrace namfile
    $ns flush-trace
    close $ntrace
    close $namfile
    exec nam prog1.nam &
    puts "The number of packet drops is"
    exec grep -c "^d" prog1.tr &
    exit 0
}

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

$ns duplex-link $n0 $n1 10Mb 10ms DropTail
$ns duplex-link $n1 $n2 5Mb 10ms DropTail

$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 05

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
```

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $n2 $sink0
$ns connect $tcp0 $sink0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set type_ CBR
$cbr0 set packetSize_ 100
$cbr0 set rate_ 1Mb
$cbr0 set random_ false
$cbr0 attach-agent $tcp0
$tcp0 set class_ 1
$ns at 1.0 "$cbr0 start"
$ns at 5.0 "Finish"
$ns run
```

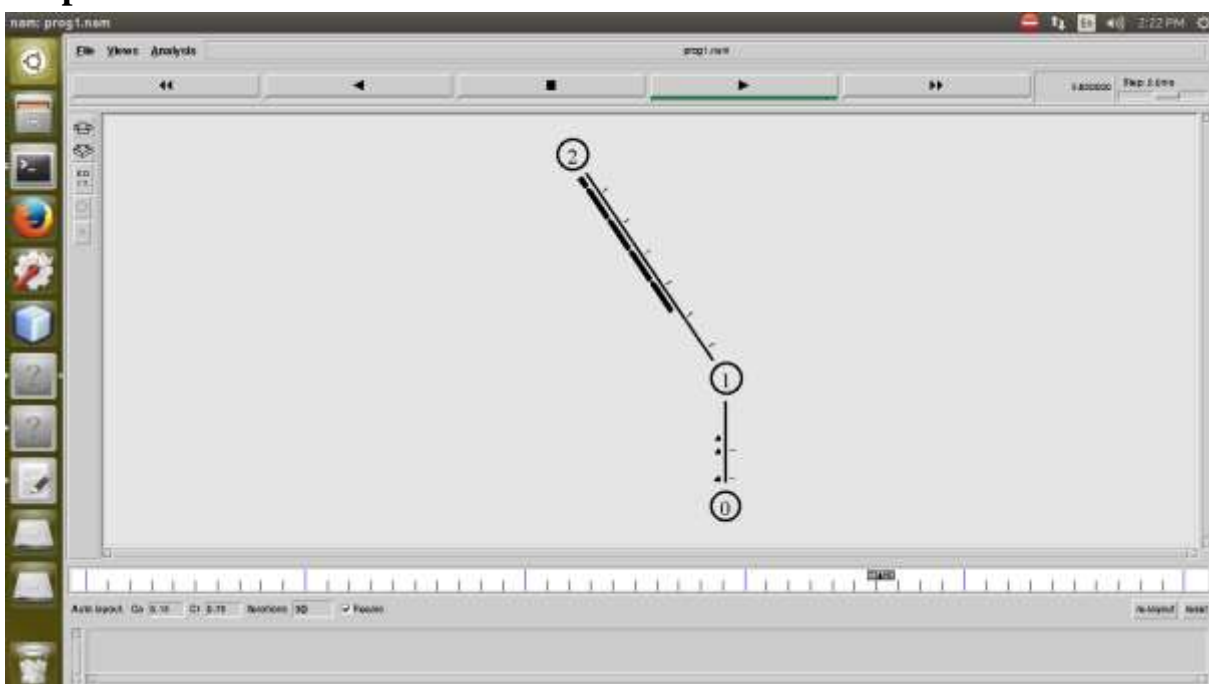
### **Output:**

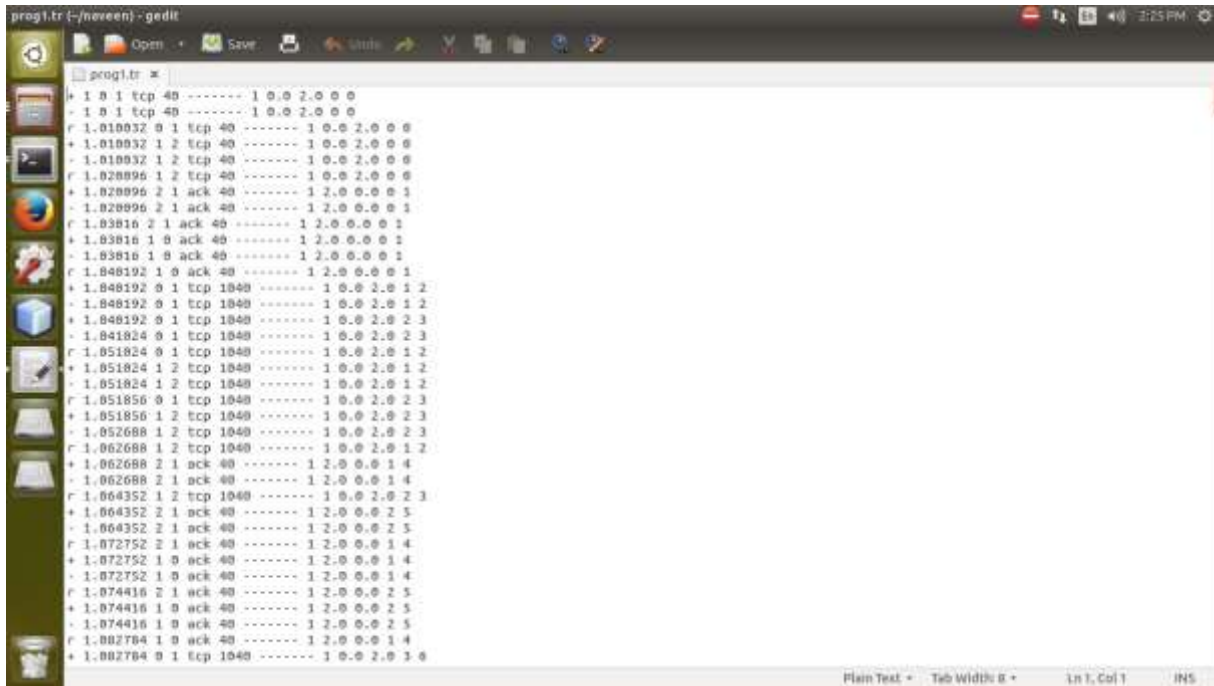
#### **Steps for execution**

1. Open gedit and type program. Program name should have the extension “.tcl”  
**student@cnpc022:~/student\$ gedit prog1.tcl**
2. Save the program.
3. Run the simulation program  
**student@cnpc022:~/ student \$ ns prog1.tcl**  
**The number of packet drops is 5**
4. Here “ns” indicates network simulator. We get the topology shown in the snapshot.
5. Now press the play button in the simulation window and the simulation will begin.
6. To see the trace file contents open the file as ,  
**student@cnpc022:~/ student \$ gedit prog1.tr**

### **Topology**

#### **Snapshot 1:**



**Snapshot 2:****Trace FileSnapshot**

```
prog1.tr *
1 0 1 tcp 40 ----- 1 0.0 2.0 0 0
1 0 1 tcp 40 ----- 1 0.0 2.0 0 0
1.010032 0 1 tcp 40 ----- 1 0.0 2.0 0 0
1.010032 1 2 tcp 40 ----- 1 0.0 2.0 0 0
1.010032 1 2 tcp 40 ----- 1 0.0 2.0 0 0
1.020096 1 2 tcp 40 ----- 1 0.0 2.0 0 0
1.020096 2 1 ack 40 ----- 1 2.0 0.0 0 1
1.020096 2 1 ack 40 ----- 1 2.0 0.0 0 1
1.03816 2 1 ack 40 ----- 1 2.0 0.0 0 1
1.03816 1 0 ack 40 ----- 1 2.0 0.0 0 1
1.03816 1 0 ack 40 ----- 1 2.0 0.0 0 1
1.040192 1 0 ack 40 ----- 1 2.0 0.0 0 1
1.040192 0 1 tcp 1040 ----- 1 0.0 2.0 1 2
1.040192 0 1 tcp 1040 ----- 1 0.0 2.0 1 2
1.040192 0 1 tcp 1040 ----- 1 0.0 2.0 2 3
1.041824 0 1 tcp 1040 ----- 1 0.0 2.0 2 3
1.051824 0 1 tcp 1040 ----- 1 0.0 2.0 1 2
1.051824 1 2 tcp 1040 ----- 1 0.0 2.0 1 2
1.051824 1 2 tcp 1040 ----- 1 0.0 2.0 1 2
1.051856 0 1 tcp 1040 ----- 1 0.0 2.0 2 3
1.051856 1 2 tcp 1040 ----- 1 0.0 2.0 2 3
1.052688 1 2 tcp 1040 ----- 1 0.0 2.0 2 3
1.062688 1 2 tcp 1040 ----- 1 0.0 2.0 1 2
1.062688 2 1 ack 40 ----- 1 2.0 0.0 1 4
1.062688 2 1 ack 40 ----- 1 2.0 0.0 1 4
1.064352 1 2 tcp 1040 ----- 1 0.0 2.0 2 3
1.064352 2 1 ack 40 ----- 1 2.0 0.0 2 5
1.064352 2 1 ack 40 ----- 1 2.0 0.0 2 5
1.072752 2 1 ack 40 ----- 1 2.0 0.0 1 4
1.072752 1 0 ack 40 ----- 1 2.0 0.0 1 4
1.072752 1 0 ack 40 ----- 1 2.0 0.0 1 4
1.074416 2 1 ack 40 ----- 1 2.0 0.0 2 5
1.074416 1 0 ack 40 ----- 1 2.0 0.0 2 5
1.074416 1 0 ack 40 ----- 1 2.0 0.0 2 5
1.082704 1 0 ack 40 ----- 1 2.0 0.0 1 4
1.082704 0 1 tcp 1040 ----- 1 0.0 2.0 3 0
```

**Program Outcome :**

- Implement the Duplex link between the networks.

**Viva Questions:**

- Define Duplex link
- Define Bandwidth

---

**Program No. 2: Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion in the network.**

**Program Objective:**

- Understand the Implementation of the network topology consisting of n nodes.

```
set ns [new Simulator]
set ntrace [open prog4.tr w]
$ns trace-all $ntrace
set namfile [open prog4.nam w]
$ns namtrace-all $namfile

proc finish {} {
    global ns ntrace namfile
    $ns flush-trace
    close $ntrace
    close $namfile
    exec nam prog4.nam &
    puts "the number of ping packets dropped are"
    exec grep -c "^d" prog4.tr &
    exit 0
}

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]

$ns duplex-link $n1 $n0 1Mb 120ms DropTail
$ns duplex-link $n2 $n0 1Mb 10ms DropTail
$ns duplex-link $n3 $n0 1Mb 10ms DropTail
$ns duplex-link $n4 $n0 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail
$ns duplex-link $n6 $n0 1Mb 11ms DropTail

Agent/Ping instproc recv {from rtt} {
    $self instvar node_
    puts "node [$node_ id] received ping answer from $from round-trip-time $rtt ms"
}
set p1 [new Agent/Ping]
```

---

```
set p2 [new Agent/Ping]
set p3 [new Agent/Ping]
set p4 [new Agent/Ping]
set p5 [new Agent/Ping]
set p6 [new Agent/Ping]

$ns attach-agent $n1 $p1

$ns attach-agent $n2 $p2
$ns attach-agent $n3 $p3
$ns attach-agent $n4 $p4
$ns attach-agent $n5 $p5
$ns attach-agent $n6 $p6

$ns queue-limit $n0 $n4 3
$ns queue-limit $n0 $n5 1
$ns queue-limit $n0 $n6 1
$ns connect $p1 $p4
$ns connect $p2 $p5
$ns connect $p3 $p6

$ns at 0.1 "$p1 send"
$ns at 0.3 "$p2 send"
$ns at 0.5 "$p3 send"
$ns at 1.0 "$p4 send"
$ns at 1.2 "$p5 send"
$ns at 1.4 "$p6 send"
$ns at 2.0 "finish"
$ns run
```

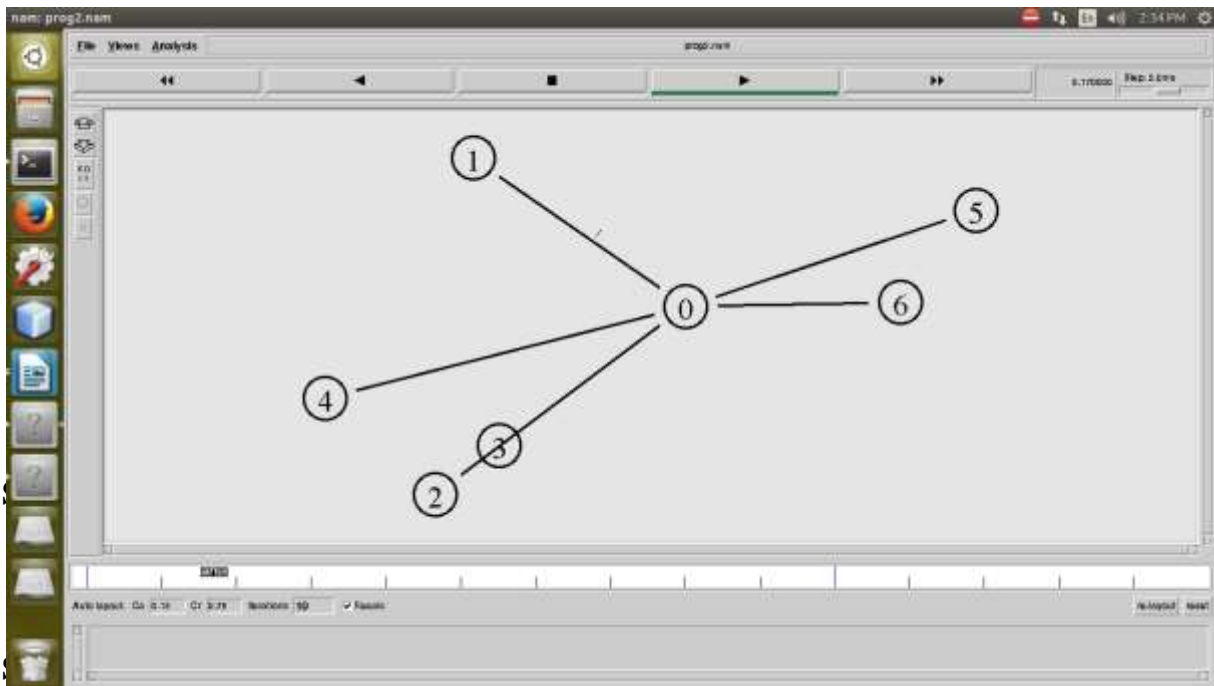
### **Output:**

#### **Steps for execution**

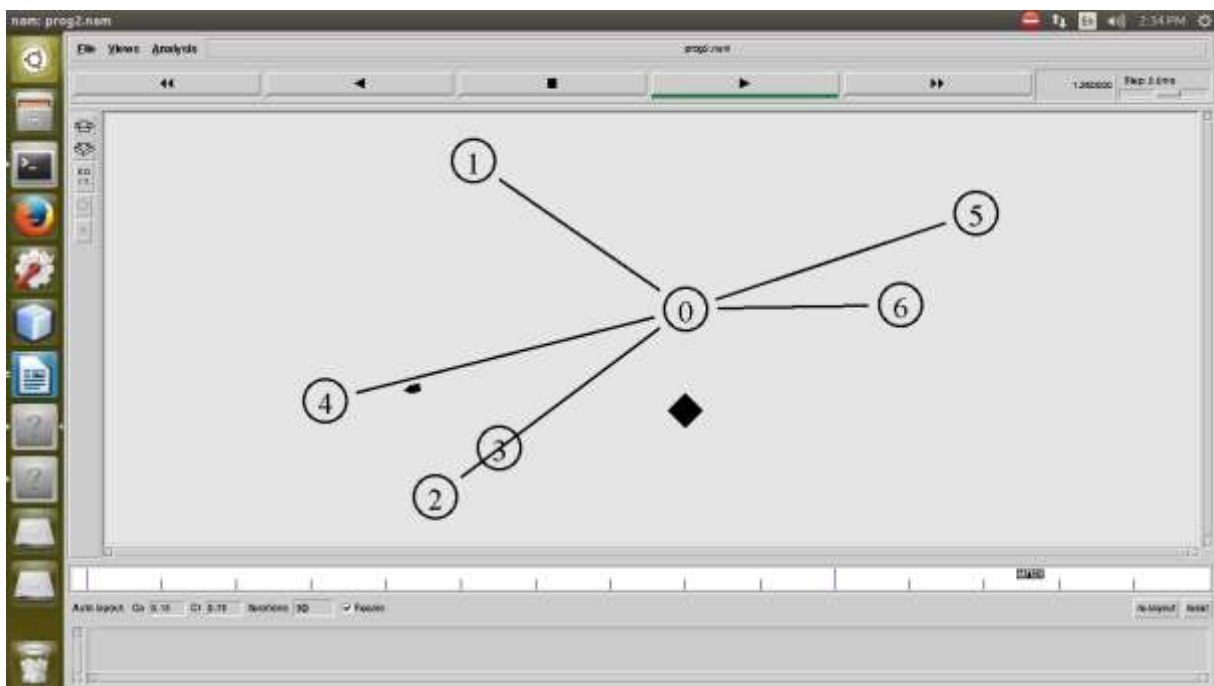
1. Open gedit and type program. Program name should have the extension “.tcl”  
**student@cnp022:~/ student\$ gedit prog4.tcl**
2. Save the program.
3. Run the simulation program  
**student@cnp022:~/ student\$ ns prog4.tcl**  
**node 1 received ping answer from 4 round-trip-time 262.0 ms**  
**node 4 received ping answer from 1 round-trip-time 262.0 ms**  
**the number of ping packets dropped are 4**
4. Here “ns” indicates network simulator. We get the topology shown in the snapshot.
5. Now press the play button in the simulation window and the simulation will begin.
6. To see the trace file contents open the file as ,  
**student@cnp022:~/ student\$ gedit prog4.tr**

## Topology

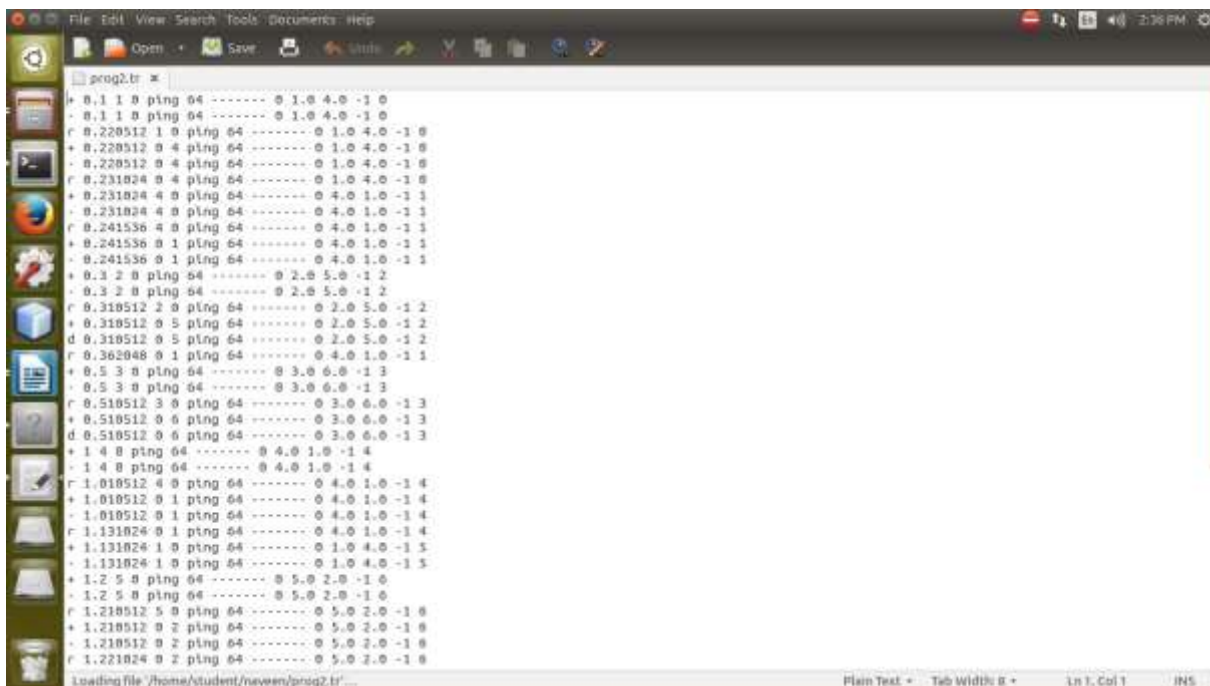
### Snapshot 1:



### Snapshot 2:





**Snapshot 3:****Trace FileSnapshot**

```
File Edit View Search Tools Documents Help
prog2.tr
0.1 1 0 ping 64 ----- 0 1.0 4.0 -1 0
0.1 1 0 ping 64 ----- 0 1.0 4.0 -1 0
0.220512 1 0 ping 64 ----- 0 1.0 4.0 -1 0
0.220512 0 4 ping 64 ----- 0 1.0 4.0 -1 0
0.220512 0 4 ping 64 ----- 0 1.0 4.0 -1 0
0.231024 0 4 ping 64 ----- 0 1.0 4.0 -1 0
0.231024 4 0 ping 64 ----- 0 4.0 1.0 -1 1
0.231024 4 0 ping 64 ----- 0 4.0 1.0 -1 1
0.241536 4 0 ping 64 ----- 0 4.0 1.0 -1 1
0.241536 0 1 ping 64 ----- 0 4.0 1.0 -1 1
0.241536 0 1 ping 64 ----- 0 4.0 1.0 -1 1
0.3 2 0 ping 64 ----- 0 2.0 5.0 -1 2
0.3 2 0 ping 64 ----- 0 2.0 5.0 -1 2
0.310512 2 0 ping 64 ----- 0 2.0 5.0 -1 2
0.310512 0 5 ping 64 ----- 0 2.0 5.0 -1 2
0.310512 0 5 ping 64 ----- 0 2.0 5.0 -1 2
0.362048 0 1 ping 64 ----- 0 4.0 1.0 -1 1
0.5 3 0 ping 64 ----- 0 3.0 6.0 -1 3
0.5 3 0 ping 64 ----- 0 3.0 6.0 -1 3
0.510512 3 0 ping 64 ----- 0 3.0 6.0 -1 3
0.510512 0 6 ping 64 ----- 0 3.0 6.0 -1 3
0.510512 0 6 ping 64 ----- 0 3.0 6.0 -1 3
1 4 0 ping 64 ----- 0 4.0 1.0 -1 4
1 4 0 ping 64 ----- 0 4.0 1.0 -1 4
1.010512 4 0 ping 64 ----- 0 4.0 1.0 -1 4
1.010512 0 1 ping 64 ----- 0 4.0 1.0 -1 4
1.010512 0 1 ping 64 ----- 0 4.0 1.0 -1 4
1.131024 0 1 ping 64 ----- 0 4.0 1.0 -1 4
1.131024 1 0 ping 64 ----- 0 1.0 4.0 -1 5
1.131024 1 0 ping 64 ----- 0 1.0 4.0 -1 5
1.2 5 0 ping 64 ----- 0 5.0 2.0 -1 6
1.2 5 0 ping 64 ----- 0 5.0 2.0 -1 6
1.210512 5 0 ping 64 ----- 0 5.0 2.0 -1 6
1.210512 0 2 ping 64 ----- 0 5.0 2.0 -1 6
1.210512 0 2 ping 64 ----- 0 5.0 2.0 -1 6
1.221024 0 2 ping 64 ----- 0 5.0 2.0 -1 6
Loading file 'C:\home\student\neveen\prog2.tr' ...
Plain Text * Tab Width: 8 * Ln 1, Col 1 INS
```

**Program Outcome :**

- Implement network topology consisting of n nodes.

**Viva Questions:**

- What is network?
- Define Topology
- Explain different types of Topology
- What is congestion?

---

**Program No. 3: Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.**

---

**Program Objective:**

- Understand the Implementation of the Ethernet LAN using n nodes.

```
#Create Simulator
set ns [new Simulator]
#Use colors to differentiate the traffics
$ns color 1 Blue
$ns color 2 Red
set ntrace [open prog6.tr w]
$ns trace-all $ntrace
set namfile [open prog6.nam w]
$ns namtrace-all $namfile
#Use some flat file to create congestion graph windows
set File0 [open WinFile0 w]
set File1 [open WinFile1 w]

#Finish Procedure
proc Finish {} {
    global ns ntrace namfile
    $ns flush-trace
    close $ntrace
    close $namfile
    exec nam prog6.nam &
    #Plot the Congestion Window graph using xgraph
    exec xgraph WinFile0 WinFile1 &
    exit 0
}

#Plot Window Procedure
proc PlotWindow {tcpSource file} {
    global ns
    set time 10.0
    set now [$ns now]
    set cwnd [$tcpSource set cwnd_]
    puts $file "$now $cwnd"
    $ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}

#Create 6 nodes
for {set i 0} {$i<6} {incr i} {
    set n($i) [$ns node]
}
```

---

```
#Create duplex links between the nodes
$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.6Mb 100ms DropTail
#Nodes n(3) , n(4) and n(5) are considered in a LAN

#Creates a Lan from a set of nodes, Bandwidth, delay characteristics along with link
#layer,
#interface queue, Mac Layer and channel type for the LAN are defined

set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3
Channel]

#Orientation to the nodes
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right

#Setup queue between n(2) and n(3) and monitor the queue
$ns queue-limit $n(2) $n(3) 20

#Set error model on link n(2) to n(3)
set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

#Set up the TCP connection between n(0) and n(4)
set tcp0 [new Agent/TCP/Newreno]
$tcp0 set fid_ 1
$tcp0 set window_ 8000
$tcp0 set packetSize_ 552
$ns attach-agent $n(0) $tcp0

set sink0 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply FTP Application over TCP
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

#Set up another TCP connection between n(5) and n(1)
set tcp1 [new Agent/TCP/Newreno]
```

---

```
$tcp1 set fid_ 2
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
$ns attach-agent $n(5) $tcp1

set sink1 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(1) $sink1
$ns connect $tcp1 $sink1

#Apply FTP application over TCP
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set type_ FTP
#Schedule Events
$ns at 0.1 "$ftp0 start"
$ns at 0.1 "PlotWindow $tcp0 $File0"
$ns at 0.5 "$ftp1 start"
$ns at 0.5 "PlotWindow $tcp1 $File1"
$ns at 25.0 "$ftp0 stop"
$ns at 25.1 "$ftp1 stop"
$ns at 25.2 "Finish"
#Run the simulation
$ns run
```

## **Output**

### **Steps for execution**

1. Open gedit and type program. Program name should have the extension “.tcl”  
**student@cnp022:~/ student\$ gedit prog6.tcl**
2. Save the program.
3. Run the simulation program  
**student@cnp022:~/ student\$ ns prog6.tcl**
4. Here “ns” indicates network simulator. We get the topology shown in the snapshot.
5. Now press the play button in the simulation window and the simulation will begin.
6. To see the trace file contents open the file as ,  
**student@cnp022:~/ student\$ gedit prog6.tr**

### Snapshot 1:



- Implement Ethernet LAN using n nodes.

- What is LAN? Explain different types of network?
- What do you mean by congestion window?

---

**Program No. 4. Write a program for error detecting code using CRC-CCITT (16-bits).**


---

**Program Objective:**

- Understand the operation of CRC-CCITT.

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check.

To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or *cyclic redundancy check* was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC. Also each data block on your hard disk has a CRC value attached to it.

Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation— by dividing it by 19 or 10011. Please note that 19 is an odd number.

This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.

$$\begin{array}{r}
 \phantom{10011} 101 = 5 \\
 10011 \overline{) 1101101} \\
 \underline{10011} \phantom{00} \\
 10000 \phantom{00} \\
 \underline{00000} \phantom{00} \\
 10000 \phantom{00} \\
 \underline{10011} \phantom{00} \\
 1110 = 14 = \text{remainder}
 \end{array}$$

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra

to check only costs one binary comparison and in 50% of the cases one binary subtraction.

You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "Lammert"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an *exclusive or* operation. Though some differences exist in the specifics across different CRC formulas, the basic mathematical process is always the same:

- The message bits are appended with  $c$  zero bits; this *augmented message* is the dividend
- A predetermined  $c+1$ -bit binary sequence, called the *generator polynomial*, is the divisor
- The checksum is the  $c$ -bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	CRC-16	CRC-32
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	110000000000000101	100000100110000010001110110110111

### International Standard CRC Polynomials

#### Algorithm:-

1. Given a bit string, append 0<sup>s</sup> to the end of it (the number of 0<sup>s</sup> is the same as the degree of the generator polynomial) let  $B(x)$  be the polynomial corresponding to  $B$ .
2. Divide  $B(x)$  by some agreed on polynomial  $G(x)$  (generator polynomial) and determine the remainder  $R(x)$ . This division is to be done using Modulo 2 Division.

3. Define  $T(x) = B(x) - R(x)$   
 $(T(x)/G(x) \Rightarrow \text{remainder } 0)$
4. Transmit  $T$ , the bit string corresponding to  $T(x)$ .  
 Let  $T'$  represent the bit stream the receiver gets and  $T'(x)$  the associated polynomial. The receiver divides  $T'(x)$  by  $G(x)$ . If there is a 0 remainder, the receiver concludes  $T = T'$  and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission

```

/* CRC */
import java.util.*;
public class Crc
{
    void div(int a[],int k)
    {
        int gp[]={ 1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
        int count=0;
        for(int i=0;i<k;i++)
        {
            if(a[i]==gp[0])
            {
                for(int j=i;j<17+i;j++)
                {
                    a[j]=a[j]^gp[count++];
                }
                count=0;
            }
        }
    }
    public static void main(String args[])
    {
        int a[]=new int[100];
        int b[]=new int[100];
        int len,k;
        Crc ob=new Crc();
        System.out.println("Enter the length of Data Frame:");
        Scanner sc=new Scanner(System.in);
        len=sc.nextInt();
        int flag=0;
        System.out.println("Enter the Message:");
        for(int i=0;i<len;i++)
        {
            a[i]=sc.nextInt();
        }
    }
}

```



---

```
for(int i=0;i<16;i++)
{
    a[len++]=0;
}
k=len-16;
for(int i=0;i<len;i++)
{
    b[i]=a[i];
}
ob.div(a,k);
for(int i=0;i<len;i++)
    a[i]=a[i]^b[i];
System.out.println("Data to be transmitted: ");
for(int i=0;i<len;i++)
{
    System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("Enter the Reveived Data: ");
for(int i=0;i<len;i++)
{
    a[i]=sc.nextInt();
}
ob.div(a, k);
for(int i=0;i<len;i++)
{
    if(a[i]!=0)
    {
        flag=1;
        break;
    }
}
if(flag==1)
    System.out.println("error in data");
else
    System.out.println("no error");
}
```

**Output1**

Enter the length of Data Frame:

5

Enter the Message:

1 1 1 0 1

Data to be transmitted:

---

1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0

Enter the Reveived Data:

1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0

no error

## Output2

Enter the length of Data Frame:

5

Enter the Message:

1 1 1 0 1

Data to be transmitted:

1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0

Enter the Reveived Data:

1 1 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 0 0

error in data

### Program Outcome

- Identify and apply the operation of CRC-CCITT.

### Viva Questions:

- Explain the features of JAVA?
- What is CRC-CCITT(16bits)?
- How CRC will detect error in a program?

---

**Program No. 5. Develop a program to implement a sliding window protocol in the data link layer.****Program Objective:**

- Understand the operation of sliding window protocol in the data link layer

Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames. The sliding window is also used in Transmission Control Protocol.

In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver. The term sliding window refers to the imaginary boxes to hold frames. Sliding window method is also known as windowing.

**Working Principle**

In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window.

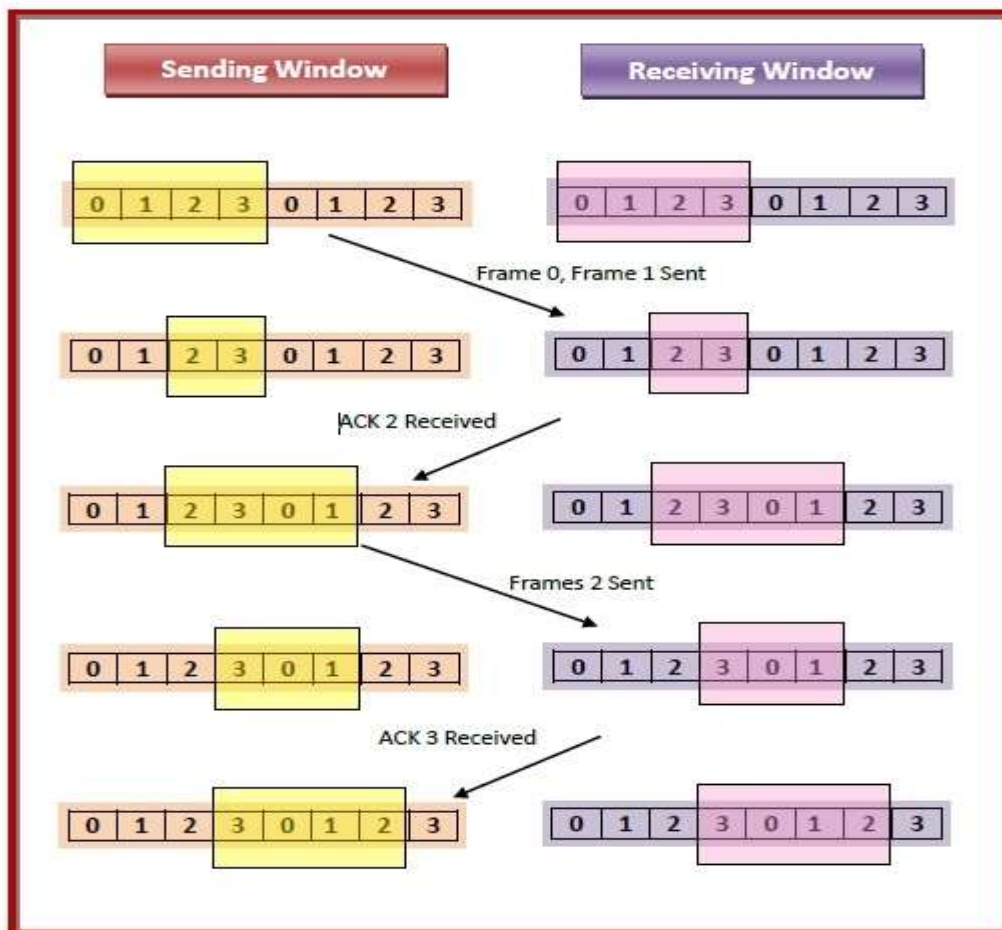
The size of the sending window determines the sequence number of the outbound frames. If the sequence number of the frames is an  $n$ -bit field, then the range of sequence numbers that can be assigned is 0 to  $2n-1$ . Consequently, the size of the sending window is  $2n-1$ . Thus in order to accommodate a sending window size of  $2n-1$ , a  $n$ -bit sequence number is chosen.

The sequence numbers are numbered as modulo- $n$ . For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

**Example**

Suppose that we have sender window and receiver window each of size 4. So the sequence numbering of both the windows will be 0,1,2,3,0,1,2 and so on. The following diagram shows the positions of the windows after sending the frames and receiving acknowledgments.



### Types of Sliding Window Protocols

The Sliding Window ARQ (Automatic Repeat reQuest) protocols are of two categories –

#### Go – Back – N ARQ

Go – Back – N ARQ provides for sending multiple frames before receiving the acknowledgment for the first frame. It uses the concept of sliding window, and so is also called sliding window protocol. The frames are sequentially numbered and a finite number of frames are sent. If the acknowledgment of a frame is not received within the time period, all frames starting from that frame are retransmitted.



#### Selective Repeat ARQ

This protocol also provides for sending multiple frames before receiving the acknowledgment for the first frame. However, here only the erroneous or lost frames are retransmitted, while the good frames are received and buffered.

---

**/\* Sliding window protocol in the data link layer \*/**

### **1. Sender Implementation:**

**File Name: Sender.java**

```
import java.io.*;
import java.net.*;
public class Sender {
    public static void main(String[] args) throws Exception {
        DatagramSocket socket = new DatagramSocket();
        InetAddress receiverAddress = InetAddress.getByName("localhost");
        int receiverPort = 9876;
        int windowSize = 4;
        byte[] data = "Hello, Sliding Window!".getBytes();
        int base = 0;
        while (base < data.length) {
            for (int i = base; i < base + windowSize && i < data.length; i++) {
                DatagramPacket packet = new DatagramPacket(data, i, 1, receiverAddress,
receiverPort);
                socket.send(packet);
            }
            DatagramPacket ackPacket = new DatagramPacket(new byte[1], 1);
            socket.receive(ackPacket);
            int ack = ackPacket.getData()[0];
            if (ack >= base) {
                base = ack + 1;
            }
        }
        socket.close();
    }
}
```

### **Output:**

```
Sent: H
Sent: e
Sent: l
Sent: l
Received ACK: 0
Sent: o
Sent: ,
Sent: S
Received ACK: 1
Sent: l
```

---

Sent: i  
Sent: d  
Received ACK: 2  
Sent: i  
Sent: n  
Received ACK: 3  
Sent: g  
Sent:  
Received ACK: 4  
Sent: W  
Sent: i  
Sent: n  
Sent: d  
Received ACK: 5  
Sent: o  
Sent: w  
Sent: !  
Received ACK: 6

## 2. Receiver Implementation:

**File Name: Receiver.java**

```
import java.io.*;
import java.net.*;
public class Receiver {
    public static void main(String[] args) throws Exception {
        DatagramSocket socket = new DatagramSocket(9876);
        int expectedSeqNum = 0;
        while (true) {
            DatagramPacket packet = new DatagramPacket(new byte[1], 1);
            socket.receive(packet);
            int seqNum = packet.getData()[0];
            if (seqNum == expectedSeqNum) {
                System.out.println("Received: " + seqNum);
                DatagramPacket ackPacket = new DatagramPacket(new byte[] { (byte)
seqNum }, 1, packet.getAddress(), packet.getPort());
                socket.send(ackPacket);
                expectedSeqNum++;
            }
        }
    }
}
```

---

**Output:**

Received: 0  
Received: 1  
Received: 2  
Received: 3  
Received: 4  
Received: 5  
Received: 6

**Program Outcome**

- Identify and apply the operation of CRC-CCITT.

**Viva Questions:**

- Explain the Sliding window protocol in the data link layer?
- What are the types of Sliding Window Protocols?

---

**Program No. 6: Write a program to find the shortest path between vertices using bellman-ford algorithm.****Program Objective:**

- Understand the Implementation of the shortest path for bellman-ford algorithm.

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Routers that use this algorithm have to maintain the distance tables (which is a one-dimension array -- "a vector"), which tell the distances and shortest path to sending packets to each node in the network. The information in the distance table is always up date by exchanging information with the neighboring nodes. The number of data in the table equals to that of all nodes in networks (excluded itself).

The columns of table represent the directly attached neighbors whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance/or time to transmit on that path (we call this as "cost"). The measurements in this algorithm are the number of hops, latency, the number of outgoing packets, etc.

The Bellman-Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers. Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm.

If a graph contains a "negative cycle" (i.e. a cycle whose edges sum to a negative value) that is reachable from the source, then there is no cheapest path: any path that has a point on the negative cycle can be made cheaper by one more walk around the negative cycle. In such a case, the Bellman-Ford algorithm can detect negative cycles and report their existence

**Implementation Algorithm:**

1. send my routing table to all my neighbors whenever my link table changes
2. when I get a routing table from a neighbor on port P with link metric M:
3.
  - a. add L to each of the neighbor's metrics
  - b. for each entry (D, P', M') in the updated neighbor's table:
    - i. if I do not have an entry for D, add (D, P, M') to my routing table
    - ii. if I have an entry for D with metric M'', add (D, P, M') to my routing table if  $M' < M''$
4. if my routing table has changed, send all the new entries to all my neighbor



---

```
/* Bellman-Ford */
import java.util.*;
public class Belmanford
{
    private int D[];
    private int n;
    public static final int max_value=999;
    public Belmanford(int n)
    {
        this.n=n;
        D=new int[n+1];
    }
    public void shortest(int s,int a[][])
    {
        for(int i=1;i<=n;i++)
        {
            D[i]=max_value;
        }
        D[s]=0;
        for(int k=1;k<=n-1;k++)
        {
            for(int i=1;i<=n;i++)
            {
                for(int j=1;j<=n;j++)
                {
                    if(a[i][j]!=max_value)
                    {
                        if(D[j]>D[i]+a[i][j])
                        {
                            D[j]=D[i]+a[i][j];
                        }
                    }
                }
            }
        }
        for (int i=1;i<=n;i++)
        {
            for (int j=1;j<=n;j++)
            {
                if(a[i][j]!=max_value)
                {
                    if(D[j]>D[i]+a[i][j])
                    {
                        System.out.println("the graph contains -ve edge cycle");
                        return;
                    }
                }
            }
        }
    }
}
```

---

```

    }
    }
}
for (int i=1;i<=n;i++)
{
    System.out.println("distance of source"+s+"to"+i+"is"+D[i]);
}
}
public static void main(String[] args)
{
    int n=0,s;
    Scanner sc=new Scanner(System.in);
    System.out.println("enter the no.of values");
    n=sc.nextInt();
    int a[][]=new int [n+1][n+1];
    System.out.println("enter the weighted matrix:");
    for (int i=1;i<=n;i++)
    {
        for (int j=1;j<=n;j++)
        {
            a[i][j]=sc.nextInt();
            if(i==j)
            {
                a[i][j]=0;
                continue;
            }
            if(a[i][j]==0)
                a[i][j]=max_value;
        }
    }
    System.out.println("enter the source vertex:");
    s=sc.nextInt();
    Belmanford b=new Belmanford(n);
    b.shortest(s,a);
    sc.close();
}
}

```

**Output1**

```

enter the no.of values
4
enter the weighted matrix:
0 999 999 999
5 0 3 4

```

---

```
999 999 0 2
999 999 999 0
enter the source vertex:
2
distance of source 2 to 1 is 5
distance of source 2 to 2 is 0
distance of source 2 to 3 is 3
distance of source 2 to 4 is 4
```

**Output2:**

```
enter the no.of values
4
enter the weighted matrix:
0 4 999 5
999 0 999 999
```

```
999 -10 0 999
999 999 3 0
enter the source vertex:
1
distance of source 1 to 1 is 0
distance of source 1 to 2 is -2
distance of source 1 to 3 is 8
distance of source 1 to 4 is 5
```

**Output3**

```
enter the no.of values
4
enter the weighted matrix:
0 4 5 999
999 0 999 7
999 7 0 999
999 999 -15 0
enter the source vertex:
1
the graph contains -ve edge cycle
```

**Program Outcome**

- Implement the shortest path for bellman-ford algorithm.

**Viva Questions:**

- What is bellman ford algorithm?
- What are the advantages and applications of bell man ford algorithm?

---

**Program No. 7: Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.**

**Program Objective:**

- Understand the Implementation of the transport layer protocols.

The term network programming refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols

- **TCP** – TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP** – UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

Sockets are a protocol independent method of creating a connection between processes. Sockets can be either

- Connection based or connectionless: Is a connection established before communication or does each packet describe the destination?
- Packet based or streams based: Are there message boundaries or is it one stream?
- Reliable or unreliable: Can messages be lost, duplicated, reordered, or corrupted?

**Socket characteristics**

Sockets are characterized by their domain, type and transport protocol. Common domains are:

- AF\_UNIX: address format is UNIX pathname
- AF\_INET: address format is host and port number

Common types are:

- virtual circuit: received in order transmitted and reliably
- datagram: arbitrary order, unreliable

Each socket type has one or more protocols. Ex:

- TCP/IP (virtual circuits)
- UDP (datagram)

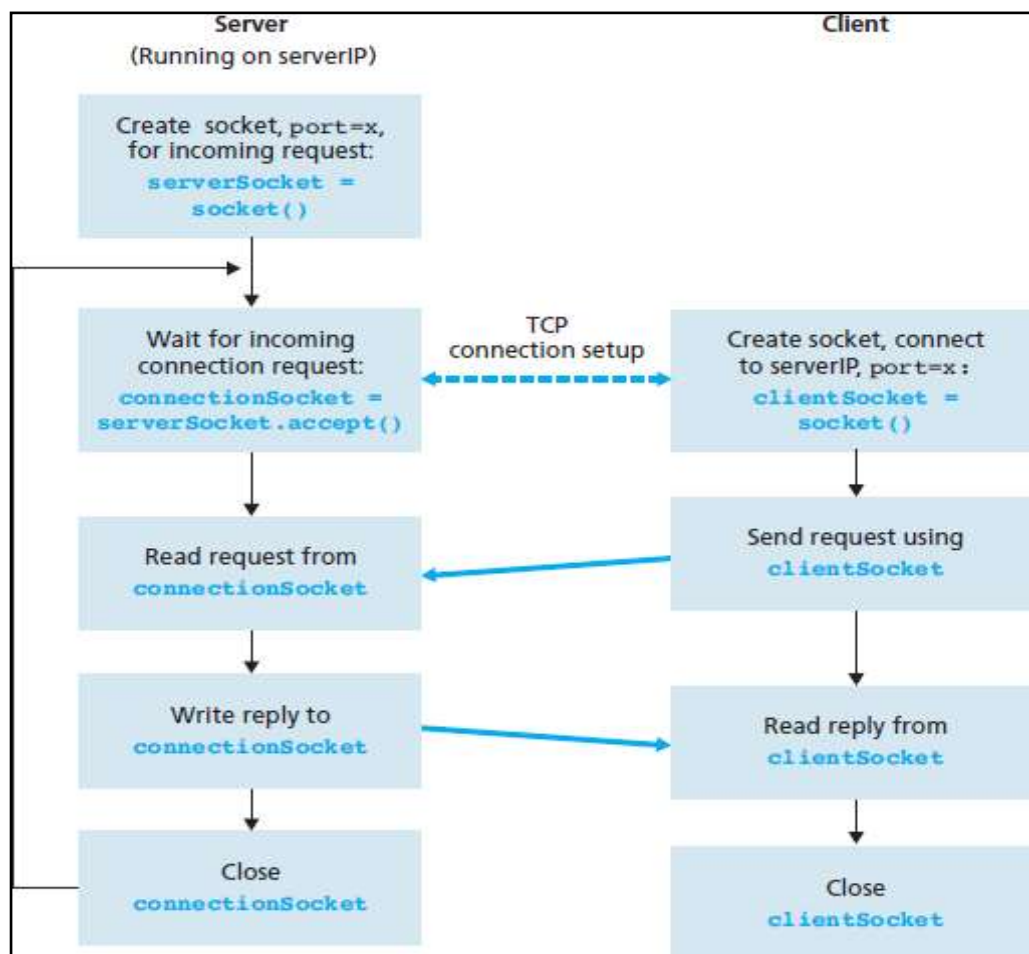
Use of sockets:

- Connection-based sockets communicate client-server: the server waits for a connection from the client
- Connectionless sockets are peer-to-peer: each process is symmetric.

Socket is an interface which enables the client and the server to communicate and pass on information from one another. Sockets provide the communication mechanism between two computers using TCP.

A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication.

The client and the server can now communicate by writing to and reading from the socket. The `java.net.Socket` class represents a socket, and the `java.net.ServerSocket` class provides a mechanism for the server program to listen for clients and establish connections with them.



**Fig. 6 The client-server application using TCP**

As shown in the Fig. 1, the following steps occur when establishing a TCP connection between two computers using sockets –

- The server instantiates a `ServerSocket` object, denoting which port number communication is to occur on.
- The server invokes the `accept()` method of the `ServerSocket` class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a `Socket` object, specifying the server name and the port number to connect to.

- The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

### **Features of TCP protocol**

**The following are the features of a TCP protocol:**

#### **Transport Layer Protocol**

**TCP is a transport layer protocol as it is used in transmitting the data from the sender to the receiver.**

#### **Reliable**

**TCP is a reliable protocol as it follows the flow and error control mechanism. It also supports the acknowledgment mechanism, which checks the state and sound arrival of the data. In the acknowledgment mechanism, the receiver sends either positive or negative acknowledgment to the sender so that the sender can get to know whether the data packet has been received or needs to resend.**

#### **Order of the data is maintained**

This protocol ensures that the data reaches the intended receiver in the same order in which it is sent. It orders and numbers each segment so that the TCP layer on the destination side can reassemble them based on their ordering.

#### **Connection-oriented**

It is a connection-oriented service that means the data exchange occurs only after the connection establishment. When the data transfer is completed, then the connection will get terminated.

#### **Full duplex**

It is a full-duplex means that the data can transfer in both directions at the same time.

#### **Stream-oriented**

TCP is a stream-oriented protocol as it allows the sender to send the data in the form of a stream of bytes and also allows the receiver to accept the data in the form of a stream of bytes. TCP creates an environment in which both the sender and receiver are connected by an imaginary tube known as a virtual circuit. This virtual circuit carries the stream of bytes across the internet.

#### **Advantages of TCP**

- It provides a connection-oriented reliable service, which means that it guarantees the delivery of data packets. If the data packet is lost across the network, then the TCP will resend the lost packets.
- It provides a flow control mechanism using a sliding window protocol.
- It provides error detection by using checksum and error control by using Go Back

---

or ARP protocol.

- It eliminates the congestion by using a network congestion avoidance algorithm that includes various schemes such as additive increase/multiplicative decrease (AIMD), slow start, and congestion window.

### **Disadvantage of TCP**

- It increases a large amount of overhead as each segment gets its own TCP header, so fragmentation by the router increases the overhead.

### Algorithm (Client Side)

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

### Algorithm (Server Side)

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

---

**TCP Client****At client side:**

```
/*TCPClient*/
import java.net.*;
import java.io.*;
public class TCPClient
{
    public static void main(String args[]) throws Exception
    {
        Socket sock=new Socket("127.0.0.1",4000);
        System.out.println("Enter the filename");
        BufferedReader keyRead=new BufferedReader(new
        InputStreamReader(System.in));
        String fname=keyRead.readLine();
        OutputStream ostream=sock.getOutputStream();
        PrintWriter pwrite=new PrintWriter(ostream,true);
        pwrite.println(fname);
        InputStream istream=sock.getInputStream();
        BufferedReader socketRead=new BufferedReader(new
        InputStreamReader(istream));
        String str;
        while((str=socketRead.readLine())!=null)
        {
            System.out.println(str);
        }
        pwrite.close();
        socketRead.close();
        keyRead.close();
    }
}
```

**TCP Server****At server side:**

```
/* TCPServer */
import java.net.*;
import java.io.*;
public class TCPServer
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket sersock=new ServerSocket(4000);
        System.out.println("Server ready for Connection");
        Socket sock=sersock.accept();
        System.out.println("Connection is Successful and waiting for chatting");
    }
}
```



```
        InputStream istream=sock.getInputStream();
        BufferedReader fileRead=new BufferedReader(new
        InputStreamReader(istream));
        String fname=fileRead.readLine();
        BufferedReader contentRead=new BufferedReader(new FileReader(fname));
        OutputStream ostream=sock.getOutputStream();
        PrintWriter pwrite=new PrintWriter(ostream,true);
        String str;
        while((str=contentRead.readLine())!=null)
        {
            pwrite.println(str);
        }
        sock.close();
        sersock.close();
        pwrite.close();
        fileRead.close();
        contentRead.close();
    }
}
```

**Note:** Create two different files TcpClient.java and TcpServer.java. Follow the steps given:

1. Open a terminal run the server program and provide the filename to send
2. Open one more terminal run the client program and provide the IP address of the server. We can give localhost address "127.0.0.1" as it is running on same machine or give the IP address of the machine.
3. Send any start bit to start sending file.

### **First Method of Executing TCP/IP socket based program**

#### **Output1**

##### **TCPServer**

first run TCPServer program .you will get below message that server is started and ready to connect with client

##### **Server ready for Connection**

##### **TCPClient**

Next run TCPClient program

##### **Enter the filename**

/home/student/user/abc.txt

hello atme college of engineering

---

## Second Method of Executing TCP/IP socket based program

### Output1

```
student@student:~/user$ javac TCPServer.java
student@student:~/user$ java TCPServer
Server ready for Connection
Connection is Successful and waiting for chatting
```

```
student@student:~/user$
student@student:~/user$ javac TCPClient.java
student@student:~/user$ java TCPClient
Enter the filename
abc.txt
atme college of engineering,mysuru
student@student:~/naveen$
```

### Program Outcome

- Implement transport layer protocols.

### Viva Questions:

- Explain the client/server model in detail.
- What do you mean by TCP/IP Socket and what is its use?

---

**Program No. 8: Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.**

**Program Objective:**

- Understand the Implementation of the data link layer protocols.
- In computer networking, the UDP stands for User Datagram Protocol. The David P. Reed developed the UDP protocol in 1980. It is defined in RFC 768, and it is a part of the TCP/IP protocol, so it is a standard protocol over the internet. The UDP protocol allows the computer applications to send the messages in the form of datagrams from one machine to another machine over the Internet Protocol (IP) network.
- The UDP is an alternative communication protocol to the TCP protocol (transmission control protocol). Like TCP, UDP provides a set of rules that governs how the data should be exchanged over the internet. The UDP works by encapsulating the data into the packet and providing its own header information to the packet. Then, this UDP packet is encapsulated to the IP packet and sent off to its destination. Both the TCP and UDP protocols send the data over the internet protocol network, so it is also known as TCP/IP and UDP/IP.
- There are many differences between these two protocols. UDP enables the process to process communication, whereas the TCP provides host to host communication. Since UDP sends the messages in the form of datagrams, it is considered the best-effort mode of communication.
- TCP sends the individual packets, so it is a reliable transport medium. Another difference is that the TCP is a connection-oriented protocol whereas, the UDP is a connectionless protocol as it does not require any virtual circuit to transfer the data.
- UDP also provides a different port number to distinguish different user requests and also provides the checksum capability to verify whether the complete data has arrived or not; the IP layer does not provide these two services.

**Features of UDP protocol**

The following are the features of the UDP protocol:

**Transport layer protocol**

UDP is the simplest transport layer communication protocol. It contains a minimum amount of communication mechanisms. It is considered an unreliable protocol, and it is based on best-effort delivery services. UDP provides no acknowledgment mechanism, which means that the receiver does not send the acknowledgment for the received packet, and the sender also does not wait for the acknowledgment for the packet that it has sent.

**Connectionless** The UDP is a connectionless protocol as it does not create a virtual path to transfer the data. It does not use the virtual path, so packets are sent in different paths between the sender and the receiver, which leads to the loss of packets or received out of order.

---

**Ordered delivery of data is not guaranteed.**

In the case of UDP, the datagrams are sent in some order will be received in the same order is not guaranteed as the datagrams are not numbered.

**Ports**

The UDP protocol uses different port numbers so that the data can be sent to the correct destination. The port numbers are defined between 0 and 1023.

**Faster transmission**

UDP enables faster transmission as it is a connectionless protocol, i.e., no virtual path is required to transfer the data. But there is a chance that the individual packet is lost, which affects the transmission quality. On the other hand, if the packet is lost in TCP connection, that packet will be resent, so it guarantees the delivery of the data packets.

**Acknowledgment mechanism**

The UDP does have any acknowledgment mechanism, i.e., there is no handshaking between the UDP sender and UDP receiver. If the message is sent in TCP, then the receiver acknowledges that I am ready, then the sender sends the data. In the case of TCP, the handshaking occurs between the sender and the receiver, whereas in UDP, there is no handshaking between the sender and the receiver.

**Segments are handled independently.**

Each UDP segment is handled individually of others as each segment takes different path to reach the destination. The UDP segments can be lost or delivered out of order to reach the destination as there is no connection setup between the sender and the receiver.

**Stateless**

It is a stateless protocol that means that the sender does not get the acknowledgement for the packet which has been sent.

**Why do we require the UDP protocol?**

As we know that the UDP is an unreliable protocol, but we still require a UDP protocol in some cases. The UDP is deployed where the packets require a large amount of bandwidth along with the actual data. For example, in video streaming, acknowledging thousands of packets is troublesome and wastes a lot of bandwidth. In the case of video streaming, the loss of some packets couldn't create a problem, and it can also be ignored.

**Datagram socket and Datagram packets**

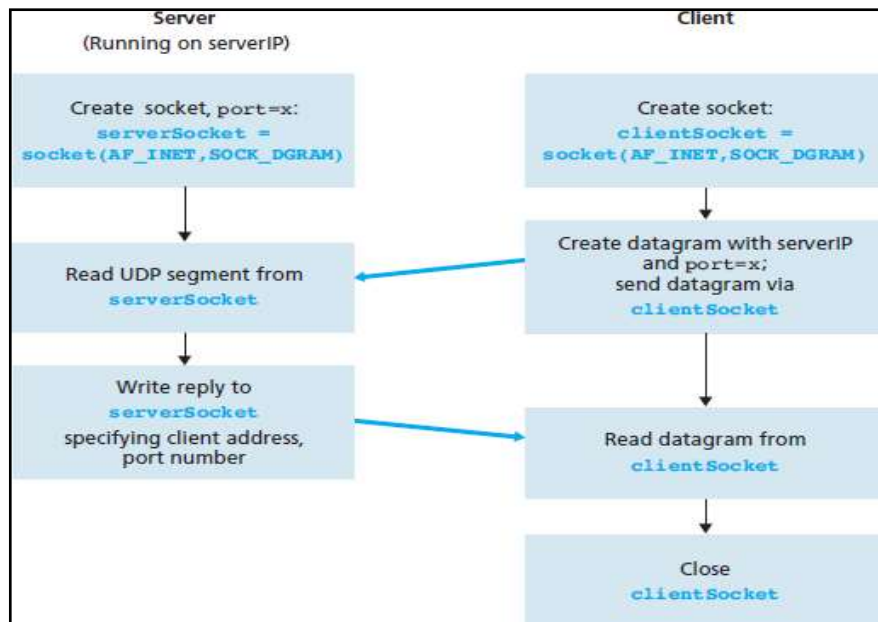
A datagram socket is the one for sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

Datagram packets are used to implement a connectionless packet delivery service supported by the UDP protocol. Each message is transferred from source machine to

destination based on information contained within that packet. That means, each packet needs to have destination address and each packet might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

Java supports datagram communication through the following classes:

- DatagramPacket
- DatagramSocket



**Fig -7-UDP client/server communication flow:**

**/\* Datagram Socket Program \*/**

**UDP Server**

```

import java.io.*;
import java.net.*;
public class UDPServer
{
    public static void main(String[] args)
    {
        DatagramSocket skt=null;
        try
        {
            System.out.println("server is started");
            skt=new DatagramSocket(6788);
            byte[] buffer = new byte[1000];
            while(true)
            {
                DatagramPacket request = new DatagramPacket(buffer,buffer.length);
                skt.receive(request);
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
  
```

---

```
String[] message = (new String(request.getData())).split(" ");
byte[] sendMsg= (message[1].toUpperCase()+ " from server to client").getBytes();
DatagramPacket reply = new
DatagramPacket(sendMsg,sendMsg.length,request.getAddress(),request.getPort());
skt.send(reply);
}
}
catch(Exception ex)
{
System.out.println(ex.getMessage());
}
}
}
```

### **UDP Client**

```
import java.io.*;
import java.net.*;
public class UDPClient
{
    public static void main(String[] args)
    {
        DatagramSocket skt;
        try
        {
            skt=new DatagramSocket();
            String msg= "atme college ";
            byte[] b = msg.getBytes();
            InetAddress host=InetAddress.getByName("127.0.0.1");
            int serverSocket=6788;
            DatagramPacket request =new DatagramPacket (b,b.length,host,serverSocket);
            skt.send(request);
            byte[] buffer =new byte[1000];
            DatagramPacket reply= new DatagramPacket(buffer,buffer.length);
            skt.receive(reply);
            System.out.println("client received:" +new String(reply.getData()));
            skt.close();
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

---

**Method of Executing datagram socket program for client/server to display the messages on client side, typed at the server side.**

### **Output1**

#### **UDPServer**

first run UDPServer program .you will get below message that server is started.

**server is started**

#### **UDPClient**

Next run UDPClient program

**client received:COLLEGE from server to client**

#### **Program Outcome**

- Implement data link layer protocols.

#### **Viva Questions:**

- What do you mean by UDP Socket and what is its use.
- What is the difference between TCP/IP and UDP connection?
- Which connection is more reliable?

---

**Program No. 9: Write a program for simple RSA algorithm to encrypt and decrypt the data.****Program Objective:**

- Understand the operation of RSA algorithm.

- **RSA** is algorithm used by modern computers to encrypt and decrypt messages. It is an asymmetric cryptographic algorithm. Asymmetric means that there are two different keys. This is also called *public key cryptography*, because one of them can be given to everyone. The other key must be kept private.
- It is based on the fact that finding the factors of an integer is hard (the factoring problem). RSA stands for **Ron Rivest, Adi Shamir** and **Leonard Adleman**, who first publicly described it in 1978. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key.
- The prime factors must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime factors can feasibly decode the message.
- RSA involves a public key and private key. The public key can be known to everyone; it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.
- The RSA algorithm's efficiency requires a fast method for performing the modular exponentiation operation. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted  $e$  and  $d$ , respectively) and taking the remainder of the division with  $N$ . A straight-forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo.
- Basically RSA is cryptographic algorithm which is meant to encrypt the data, generally used in network security applications while we are sending the data from one source to destination. The concept of RSA algorithm starts with a two key concepts, it uses two keys (asymmetric keys) one is considered as the public key and another is a private key.
- It was developed because using the symmetric encryption algorithm is easy but the key distribution is difficult, so the concept of two key concept appears to be more efficient. The whole algorithm depends on the fact that "It is not possible to judge another key when attacker gets one key" Here in two keys, one key is taken as the public key another as a private key.
- Public key is available for everyone to access, so whenever sender want to send the data to receiver, he uses the public key of receiver (as it is available for use to all) and encrypts the data using the key, this encrypted data is called cipher text, when receiver receives the cipher text, he can decrypt the data using his private key. Here even if the attacker knows the encryption algorithm, he can't do anything until the keys are available.



### A very simple example of RSA encryption

1. Select primes  $p = 11$ ,  $q = 3$ .

$$2. n = p \cdot q = 11 \cdot 3 = 33$$

$$\phi = (p-1)(q-1) = 10 \cdot 2 = 20$$

3. Choose  $e=3$

Check  $\gcd(e, p-1) = \gcd(3, 10) = 1$  (i.e. 3 and 10 have no common factors except 1), and check  $\gcd(e, q-1) = \gcd(3, 2) = 1$

$$\text{therefore } \gcd(e, \phi) = \gcd(e, (p-1)(q-1)) = \gcd(3, 20) = 1$$

4. Compute  $d$  such that  $ed \equiv 1 \pmod{\phi}$

$$\text{i.e. compute } d = e^{-1} \pmod{\phi} = 3^{-1} \pmod{20}$$

i.e. find a value for  $d$  such that  $\phi$  divides  $(ed-1)$

i.e. find  $d$  such that 20 divides  $3d-1$ . Simple testing ( $d = 1, 2, \dots$ ) gives  $d = 7$

Check:  $ed-1 = 3 \cdot 7 - 1 = 20$ , which is divisible by  $\phi$ .

5. Public key =  $(n, e) = (33, 3)$  Private key =  $(n, d) = (33, 7)$ .

**Now say we want to encrypt the message  $m = 7$ ,**

$$c = m^e \pmod{n} = 7^3 \pmod{33} = 343 \pmod{33} = 13.$$

Hence the ciphertext  $c = 13$ .

**To check decryption we compute**

$$m' = c^d \pmod{n} = 13^7 \pmod{33} = 7.$$

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

**One way of calculating  $m'$  is as follows:-**

$$m' = 13^7 \pmod{33} = 13^{(3+3+1)} \pmod{33} = 13^3 \cdot 13^3 \cdot 13 \pmod{33}$$

$$= (13^3 \pmod{33}) \cdot (13^3 \pmod{33}) \cdot (13 \pmod{33}) \pmod{33}$$

$$= (2197 \pmod{33}) \cdot (2197 \pmod{33}) \cdot (13 \pmod{33}) \pmod{33}$$

$$= 19 \cdot 19 \cdot 13 \pmod{33} = 4693 \pmod{33}$$

$$= 7.$$

**Key Generation Algorithm**

1. Generate two large random primes,  $p$  and  $q$ , of approximately equal size such that their product  $n = pq$  is of the required bit length, e.g. 1024 bits.

2. Compute  $n = pq$  and  $(\phi)$   $\phi = (p-1)(q-1)$ .

3. Choose an integer  $e$ ,  $1 < e < \phi$ , such that  $\gcd(e, \phi) = 1$ .
  4. Compute the secret exponent  $d$ ,  $1 < d < \phi$ , such that  $ed \equiv 1 \pmod{\phi}$ .
  5. The public key is  $(n, e)$  and the private key is  $(n, d)$ . The values of  $p$ ,  $q$ , and  $\phi$  should also be kept secret.
- $n$  is known as the modulus.
  - $e$  is known as the public exponent or encryption exponent.
  - $d$  is known as the secret exponent or decryption exponent.

Note: It is possible to find a smaller  $d$  by using  $\text{lcm}(p-1, q-1)$  instead of  $\phi$ ,  $\text{lcm}(p-1, q-1) = \phi / \gcd(p-1, q-1)$ .

## Encryption

Sender A does the following:-

1. Obtains the recipient B's public key  $(n, e)$ .
2. Represents the plaintext message as a positive integer  $m$ .
3. Computes the ciphertext  $c = m^e \bmod n$ .
4. Sends the ciphertext  $c$  to B.

## Decryption

Recipient B does the following:-

1. Uses his private key  $(n, d)$  to compute  $m = c^d \bmod n$ .
2. Extracts the plaintext from the integer representation  $m$ .

### **/\* RSA Key Generation \*/**

#### **Source Code:**

```
import java.util.*;
import java.io.*;
public class rsa
{
    static int gcd(int m,int n)
    {
        while(n!=0)
        {
            int r=m%n;
            m=n;
            n=r;
        }
        return m;
    }
    public static void main(String args[])
    {
```

---

```
int p=0,q=0,n=0,e=0,d=0,phi=0;
int nummes[]=new int[100];
int encrypted[]=new int[100];
int decrypted[]=new int[100];
int i=0,j=0,nofelem=0;
Scanner sc=new Scanner(System.in);
String message ;
System.out.println("Enter the Message to be encrypted:");
message= sc.nextLine();
System.out.println("Enter value of p and q\n");
p=sc.nextInt();
q=sc.nextInt();
n=p*q;
phi=(p-1)*(q-1);
for(i=2;i<phi;i++)
    if(gcd(i,phi)==1)
        break;
e=i;
for(i=2;i<phi;i++)
    if((e*i-1)%phi==0)
        break;
d=i;
for(i=0;i<message.length();i++)
{
    char c = message.charAt(i);
    int a =(int)c;
    nummes[i]=c-96;
}
nofelem=message.length();
for(i=0;i<nofelem;i++)
{
    encrypted[i]=1;
    for(j=0;j<e;j++)
        encrypted[i]=(encrypted[i]*nummes[i])%n;
}
System.out.println("\n Encrypted message\n");
for(i=0;i<nofelem;i++)
{
    System.out.print(encrypted[i]);
    System.out.print((char)(encrypted[i]+96));
}
for(i=0;i<nofelem;i++)
{
    decrypted[i]=1; for(j=0;j<d;j++)
```

```
        decrypted[i]=(decrypted[i]*encrypted[i])%n;
    }

    System.out.println("\n Decrypted message\n ");
    for(i=0;i<nofelem;i++)
        System.out.print((char)(decrypted[i]+96)); return;
    }
}
```

**Output**

Enter the text:

hello

Enter the value of P and Q :

5

7

Encrypted Text is: 8 h 10 j 17 q 17 q 15 o

Decrypted Text is: hello

**Program Outcome**

- Implement data link layer protocols. Identify and apply the operation of RSA algorithm.

**Viva Questions:**

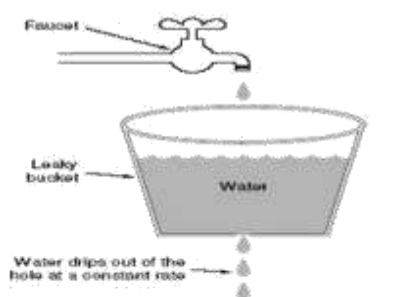
- What is RSA? Explain its algorithm.
- What do you mean by encryption and decryption of data?

**Program 10: Write a program for congestion control using leaky bucket algorithm.****Program Objective:**

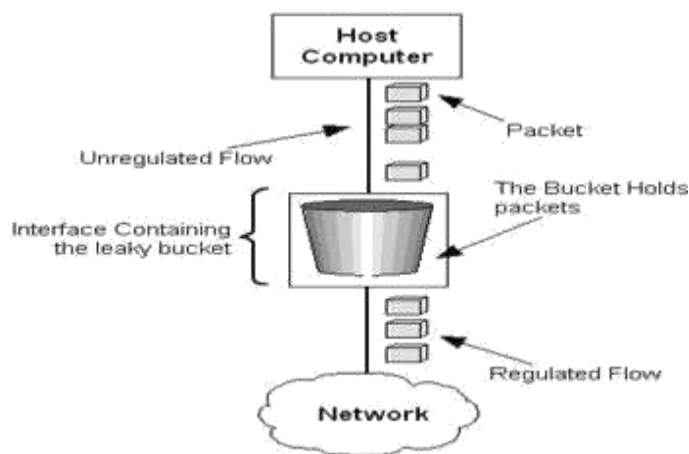
- Understand the operation of congestion control using leaky bucket algorithm.

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero.

From network perspective, leaky bucket consists of a finite queue (bucket) where all the incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets (b)).



(a) A leaky bucket with water



(b) A leaky bucket with packets

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

**The leaky-bucket algorithm:**

The algorithm can be conceptually understood as follows:

- Consider a bucket with a hole in the bottom.
- The empty space of the bucket represents an amount of credit available measured in bytes.
- The size of the bucket is  $b$  bytes. This means that if the bucket is empty,  $b$  bytes of

credit is available.

- If a packet arrives and its size is less than the available credit, the packet can be forwarded. Otherwise, it is discarded or queued depending on the application.
- The bucket leaks through the hole in its bottom at a constant rate of  $r$  bytes per second, this indicates credit accumulation.

### **/\* Leaky Bucket \*/**

```
public class LeakyBucket
{
    static int min(int x,int y)
    {
        if(x<y)
            return x;
        else
            return y;
    }
    public static void main(String[] args)
    {
        int drop=0,mini,nsec,cap,count=0,i,process;
        int inp[]=new int[25];
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter The Bucket Size\n");
        cap= sc.nextInt();
        System.out.println("Enter The Operation Rate\n");
        process= sc.nextInt();
        System.out.println("Enter The No. Of Seconds You Want To Stimulate\n");
        nsec=sc.nextInt();
        for(i=0;i<nsec;i++)
        {
            System.out.print("Enter The Size Of The Packet Entering At "+ i+1+"sec");
            inp[i] = sc.nextInt();
        }
        System.out.println("\nSecond | Packet Recieved | Packet Sent | Packet Left | Packet Dropped\n");
        //System.out.println("-----\n");
        for(i=0;i<nsec;i++)
        {
            count+=inp[i];
            if(count>cap)
            {
                drop=count-cap;
                count=cap;
            }
        }
    }
}
```

```

    }
    System.out.print(i+1);
    System.out.print("\t\t"+inp[i]);
    mini=min(count,process);
    System.out.print("\t\t"+mini);
    count=count-mini;
    System.out.print("\t\t"+count);

    System.out.print("\t\t"+drop);
    drop=0;
    System.out.println();
}
for(;count!=0;i++)
{
    if(count>cap)
    {
        drop=count-cap;
        count=cap;
    }
    System.out.print(i+1);
    System.out.print("\t\t0");
    mini=min(count,process);
    System.out.print("\t\t"+mini);
    count=count-mini;
    System.out.print("\t\t"+count);
    System.out.print("\t\t"+drop);
    System.out.println();
}
}
}

```

### Output1

Enter The Bucket Size

6

Enter The output Rate

2

Enter The No. of Seconds You Want To Stimulate

2

Enter The Size of Packet entering at 01sec

8

Enter The Size of Packet entering at 11sec

6

Second | Packet Recieved | Packet Sent | Packet Left | Packet Dropped|

1	8	2	4	2
2	6	2	4	4

---

3	0	2	2	0
4	0	2	0	0

**Output2**

Enter The Bucket Size

5

Enter The output Rate

2

Enter The No. of Seconds You Want To Stimulate

3

Enter The Size of Packet entering at 01sec

5

Enter The Size of Packet entering at 11sec

4

Enter The Size of Packet entering at 21sec

3

Second	Packet Recieved	Packet Sent	Packet Left	Packet Dropped
--------	-----------------	-------------	-------------	----------------

1	5	2	3	0
---	---	---	---	---

2	4	2	3	2
---	---	---	---	---

3	3	2	3	1
---	---	---	---	---

4	0	2	1	0
---	---	---	---	---

5	0	1	0	0
---	---	---	---	---

**Program Outcome**

- Identify and apply the operation of congestion control using leaky bucket algorithm.

**Viva Questions:**

- What is congestion control?
- Explain leaky bucket algorithm



## VIVA QUESTIONS AND ANSWERS

### 1)What is a Link?

A link refers to the connectivity between two devices. It includes the type of cables and protocols used in order for one device to be able to communicate with the other.

### 2)What are the layers of the OSI reference model?

There are 7 OSI layers: Physical Layer, Data Link Layer, Network Layer, Transport Layer, Session Layer, Presentation Layer and Application Layer.

### 3)What is a LAN?

LAN is short for Local Area Network. It refers to the connection between computers and other network devices that are located within a small physical location.

### 4)What is a node?

A node refers to a point or joint where a connection takes place. It can be computer or device that is part of a network. Two or more nodes are needed in order to form a network connection.

### 5)What are routers?

Routers can connect two or more network segments. These are intelligent network devices that store information in its routing table such as paths, hops and bottlenecks. With this info, they are able to determine the best path for data transfer. Routers operate at the OSI Network Layer.

### 6)What is point to point link?

It refers to a direct connection between two computers on a network. A point to point connection does not need any other network devices other than connecting a cable to the NIC cards of both computers.

### 7)What is subnet mask?

A subnet mask is combined with an IP address in order to identify two parts: the extended network address and the host address. Like an IP address, a subnet mask is made up of 32 bits.

### 8)What is the function of the OSI Session Layer?

This layer provides the protocols and means for two devices on the network to communicate with each other by holding a session. This includes setting up the

session, managing information exchange during the session, and tear-down process upon termination of the session.

**9)What is the importance of implementing a Fault Tolerance System? Are there limitations?**

A fault tolerance system ensures continuous data availability. This is done by eliminating a single point of failure. However, this type of system would not be able to protect data in some cases, such as in accidental deletions.

**10)What is a private IP address?**

Private IP addresses are assigned for use on intranets. These addresses are used for internal networks and are not routable on external public networks. These ensures that no conflicts are present among internal networks while at the same time the same range of private IP addresses are reusable for multiple intranets since they do not "see" each other.

**11)What is DoS?**

DoS, or Denial-of-Service attack, is an attempt to prevent users from being able to access the internet or any other network services. Such attacks may come in different forms and are done by a group of perpetrators. One common method of doing this is to overload the system server so it cannot anymore process legitimate traffic and will be forced to reset.

**12)What is OSI and what role does it play in computer networks?**

OSI (Open Systems Interconnect) serves as a reference model for data communication. It is made up of 7 layers, with each layer defining a particular aspect on how network devices connect and communicate with one another. One layer may deal with the physical media used, while another layer dictates how data is actually transmitted across the network.

**13)What are MAC addresses?**

MAC, or Media Access Control, uniquely identifies a device on the network. It is also known as physical address or Ethernet address. A MAC address is made up of 6-byte parts.

**14)What are firewalls?**

Firewalls serve to protect an internal network from external attacks. These external threats can be hackers who want to steal data or computer viruses that can wipe out data in an instant. It also prevents other users from external networks from gaining access to the private network.

**15)Describe star topology**

Star topology consists of a central hub that connects to nodes. This is one of the easiest to setup and maintain.

**16)What are gateways?**

Gateways provide connectivity between two or more network segments. It is usually a computer that runs the gateway software and provides translation services. This translation is a key in allowing different systems to communicate on the network.

**17)What is TCP/IP?**

TCP/IP is short for Transmission Control Protocol / Internet Protocol. This is a set of protocol layers that is designed to make data exchange possible on different types of computer networks, also known as heterogeneous network.

For a Class C network, the number of usable Network ID bits is 21. The number of possible network IDs is 2 raised to 21 or 2,097,152. The number of host IDs per network ID is 2 raised to 8 minus 2, or 254.

**18)What is Ping?**

Ping is a utility program that allows you to check connectivity between network devices on the network. You can ping a device by using its IP address or device name, such as a computer name.

**19)What is DNS?**

DNS is Domain Name System. The main function of this network service is to provide host names to TCP/IP address resolution.

**20)What are the maximum networks and hosts in a class A, B and C network?**

For Class A, there are 126 possible networks and 16,777,214 hosts For Class B, there are 16,384 possible networks and 65,534 hosts For Class C, there are 2,097,152 possible networks and 254 hosts

**21)What is ipconfig?**

Ipconfig is a utility program that is commonly used to identify the addresses information of a computer on a network. It can show the physical address as well as the IP address.

**22)What is client/server?**

Client/server is a type of network wherein one or more computers act as servers. Servers provide a centralized repository of resources such as printers and files. Clients refers to workstation that access the server.

**23)Describe networking.**

Networking refers to the inter connection between computers and peripherals for data communication. Networking can be done using wired cabling or through wireless link.

**24)Describe Ethernet.**

Ethernet is one of the popular networking technologies used these days. It was developed during the early 1970s and is based on specifications as stated in the IEEE. Ethernet is used in local area networks.

**25)What protocols fall under the TCP/IP Internet Layer?**

There are 4 protocols that are being managed by this layer. These are ICMP, IGMP, IP and ARP.

**26)What is IPv6?**

IPv6 , or Internet Protocol version 6, was developed to replace IPv4. At present, IPv4 is being used to control internet traffic, butis expected to get saturated in the near future. IPv6 was designed to overcome this limitation.

**27)What is mesh topology?**

Mesh topology is a setup wherein each device is connected directly to every other device on the network. Consequently, it requires that each device have at least two network connections.

**28)What is ns2?**

ns is an object-oriented,discrete event simulator targeted at networking research. ns provides substantial support for simulation of tcp, routing, andmulticast protocols over wired and wireless (local and satellite) networks. Later ns-2 (version 2) was developed at uc berkeley in c++ and otcl (object-oriented extension of tcl).

**29)What is 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.

**30) Explain basic architecture of NS-2.**

NS-2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL.

**31)What is trace file?**

The trace file (trace.tr) is a standard format used by ns2. In ns2, each time a packet moves from one node to another, or onto a link, or into a buffer, etc., it gets recorded in this trace file.

**32)What is nam file?**

A visual aid showing how packets flow along the network.

**33)Why awk file is used?**

The basic function of awk is to search files for lines (or other units of text) that contain certain patterns. When a line matches one of the patterns, awk performs specified actions on that line. awk keeps processing input lines in this way until the end of the input files are reached.

**34)What is xgraph?**

It is used for plotting purpose comes together NS2 installation package.

**35)What different layers of TCP/IP model, does link, node, agent and traffic source represent in NS-2?**

Link represents Network access layer; Node represents Internet layer; Agent represents Transport layer; and Traffic Source represents Application layer of TCP/IP model.