Maharaja Education Trust (R), Mysuru

Maharaja Institute of Technology Mysore

Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Approved by AICTE, New Delhi,
Affiliated to VTU, Belagavi & Recognized by Government of Karnataka

LAB MANUAL

COMPUTER NETWORK LAB

(18CSL57)

Prepared by

Department of Computer Science and Engineering

Maharaja Education Trust (R), Mysuru
Maharaja Institute of Technology Mysore
Belawadi, Sriranga Pattana Taluk, Mandya – 571 477

Vision/ ಆಶಯ

"To be recognized as a premier technical and management institution promoting extensive education fostering research, innovation and entrepreneurial attitude"

ಸಂಶೋಧನೆ, ಆವಿಷ್ಕಾರ ಹಾಗೂ ಉದ್ಯಮಶೀಲತೆಯನ್ನು ಉತ್ತೇಜಿಸುವ ಅಗ್ರಮಾನ್ಯ ತಾಂತ್ರಿಕ ಮತ್ತು ಆಡಳಿತ ವಿಜ್ಞಾನ ಶಿಕ್ಷಣ ಕೇಂದ್ರವಾಗಿ ಗುರುತಿಸಿಕೊಳ್ಳುವುದು.

Mission/ ಧ್ಯೇಯ

- To empower students with indispensable knowledge through dedicated teaching and collaborative learning.

  ಸಮರ್ಪಣಾ ಮನೋಭಾವದ ಬೋಧನೆ ಹಾಗೂ ಸಹಭಾಗಿತ್ವದ ಕಲಿಕಾಕ್ರಮಗಳಿಂದ ವಿದ್ಯಾರ್ಥಿಗಳನ್ನು ಅತ್ಯತ್ಕೃಷ್ಟ ಜ್ಞಾನಸಂಪನ್ನರಾಗಿಸುವುದು.

- To advance extensive research in science, engineering and management disciplines.

  ವೈಜ್ಞಾನಿಕ, ತಾಂತ್ರಿಕ ಹಾಗೂ ಆಡಳಿತ ವಿಜ್ಞಾನ ವಿಭಾಗಗಳಲ್ಲಿ ವಿಸ್ತೃತ ಸಂಶೋಧನೆಗಳೊಡನೆ ಬೆಳವಣಿಗೆ ಹೊಂದುವುದು.

- To facilitate entrepreneurial skills through effective institute - industry collaboration and interaction with alumni.

  ಉದ್ಯಮ ಕ್ಷೇತಗಳೊಡನೆ ಸಹಯೋಗ, ಸಂಸ್ಥೆಯ ಹಿರಿಯ ವಿದ್ಯಾರ್ಥಿಗಳೊಂದಿಗೆ ನಿರಂತರ ಸಂವಹನಗಳಿಂದ ವಿದ್ಯಾರ್ಥಿಗಳಿಗೆ ಉದ್ಯಮಶೀಲತೆಯ ಕೌಶಲ್ಯ ಪಡೆಯಲು ನೆರವಾಗುವುದು.

- To instill the need to uphold ethics in every aspect.

  ಜೀವನದಲ್ಲಿ ನೈತಿಕ ಮೌಲ್ಯಗಳನ್ನು ಅಳವಡಿಸಿಕೊಳ್ಳುವುದರ ಮಹತ್ವದ ಕುರಿತು ಅರಿವು ಮೂಡಿಸುವುದು.

- To mould holistic individuals capable of contributing to the advancement of the society.

  ಸಮಾಜದ ಬೆಳವಣಿಗೆಗೆ ಗಣನೀಯ ಕೊಡುಗೆ ನೀಡಬಲ್ಲ ಪರಿಪೂರ್ಣ ವ್ಯಕ್ತಿತ್ವವುಳ್ಳ ಸಮರ್ಥ ನಾಗರೀಕರನ್ನು ರೂಪಿಸುವುದು.

VISION/ ಆಶಯ

"To be a leading academic department offering computer science and engineering education, fulfilling industrial and societal needs effectively."

"ಕೈಗಾರಿಕಾ ಮತ್ತು ಸಾಮಾಜಿಕ ಅಗತ್ಯಗಳನ್ನು ಪರಿಣಾಮಕಾರಿಯಾಗಿ ಪೂರೈಸುವ ಮೂಲಕ ಕಂಪ್ಯೂಟರ್ ವಿಜ್ಞಾನ ಮತ್ತು ಎಂಜಿನಿಯರಿಂಗ್ ಶಿಕ್ಷಣವನ್ನು ನೀಡುವ ಪ್ರಮುಖ ಶೈಕ್ಷಣಿಕ ವಿಭಾಗವಾಗುವುದು."

MISSION/ ಧ್ಯೇಯ

- To enrich the technical knowledge of students in diversified areas of Computer science and engineering by adopting outcome based approaches.

  ಫಲಿತಾಂಶ ಆಧಾರಿತ ವಿಧಾನಗಳನ್ನು ಅಳವಡಿಸಿಕೊಳ್ಳುವ ಮೂಲಕ ಕಂಪ್ಯೂಟರ್ ವಿಜ್ಞಾನ ಮತ್ತು ಎಂಜಿನಿಯರಿಂಗ್‌ನ ವೈವಿಧ್ಯಮಯ ಕ್ಷೇತ್ರಗಳಲ್ಲಿನ ವಿದ್ಯಾರ್ಥಿಗಳ ತಾಂತ್ರಿಕ ಜ್ಞಾನವನ್ನು ಅಭಿವೃದ್ಧಿ ಪಡಿಸುವುದು.

- To empower students to be competent professionals maintaining ethicality.

  ನೈತಿಕತೆಯನ್ನು ಕಾಪಾಡುವ ಸಮರ್ಥ ವೃತ್ತಿಪರರಾಗಿ ವಿದ್ಯಾರ್ಥಿಗಳನ್ನು ಸಶಕ್ತಗೊಳಿಸುವುದು.

- To facilitate the development of academia-industry collaboration.

  ಶೈಕ್ಷಣಿಕ-ಉದ್ಯಮ ಸಹಯೋಗದ ಅಭಿವೃದ್ಧಿಗೆ ಅನುಕೂಲವಾಗುವಂತೆ.

- To create awareness of entrepreneurship opportunities.

  ಉದ್ಯಮಶೀಲತೆ ಅವಕಾಶಗಳ ಬಗ್ಗೆ ಜಾಗೃತಿ ಮೂಡಿಸುವುದು.

Program Outcomes

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

Maharaja Institute of Technology Mysore

Department of Computer Science and Engineering

Syllabus

Subject: COMPUTER NETWORK LABORATORY

Subject Code 18CSL57

Lab Experiments:

PART A

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. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.
5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.
6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

PART B

Implement the following in Java:

7. Write a program for error detecting code using CRC-CCITT (16- bits).
8. Write a program to find the shortest path between vertices using bellman-ford algorithm.
9. 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.
10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.
11. Write a program for simple RSA algorithm to encrypt and decrypt the data.
12. Write a program for congestion control using leaky bucket algorithm.

Maharaja Institute of Technology Mysore

Department of Computer Science and Engineering

INDEX

Subject: COMPUTER NETWORK LAB **Subject Code: 18CSL57**

SL. No.	Contents	Page No.
1	SIMULATION USING NS-2	1
2	Implement three nodes point – to – point network with duplex links between them.	6
3	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.	8
4	Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.	9
5	Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.	11
6	Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.	13
7	Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.	15
8	Write a program for error detecting code using CRC-CCITT (16- bits). import java.util.*;	17
9	Write a program to find the shortest path between vertices using bellman-ford Algorithm	19
10	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.	21
11	Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.	23
12		24

	Write a program for simple RSA algorithm to encrypt and decrypt the data.	
13	Write a program for congestion control using leaky bucket algorithm.	28
14	VIVA QUESTIONS	30

Part A - SIMULATION USING NS-2

Introduction to NS-2:
NS2 is an open-source simulation tool that runs on Linux. It is a discreet event simulator targeted at networking research and provides substantial support for simulation of routing, multicast protocols and IP protocols, such as UDP, TCP, RTP and SRM over wired and wireless (local and satellite) networks.
Widely known as NS2, is simply an event driven simulation tool.
Useful in studying the dynamic nature of communication networks.
Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2.
In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors.
Basic Architecture of NS2

TCL – Tool Command Language
Tcl is a very simple programming language. If you have programmed before, you can learn enough to write interesting Tcl programs within a few hours. This page provides a quick overview of the main features of Tcl. After reading this you'll probably be able to start writing simple Tcl scripts on your own; however, we recommend that you consult one of the many available Tcl books for more complete information.
Basic syntax
Tcl scripts are made up of commands separated by newlines or semicolons.
Commands all have the same basic form illustrated by the following example:

```
expr 20 + 10
```

This command computes the sum of 20 and 10 and returns the result, 30. You can try out this example and all the others in this page by typing them to a Tcl application such as tclsh; after a command completes, tclsh prints its result.
Each Tcl command consists of one or more words separated by spaces. In this example there are four words: expr, 20, +, and 10. The first word is the name of a command and the other words are arguments to that command. All Tcl commands consist of words,

but different commands treat their arguments differently. The expr command treats all of its arguments together as an arithmetic expression, computes the result of that expression, and returns the result as a string. In the expr command the division into words isn't significant: you could just as easily have invoked the same command as

```
expr 20+10
```

However, for most commands the word structure is important, with each word used

for a distinct purpose.
All Tcl commands return results. If a command has no meaningful result then it returns an empty string as its result.
Variables
Tcl allows you to store values in variables and use the values later in commands.
The set command is used to write and read variables. For example, the following command modifies the variable x to hold the value 32:

```
set x 32
```

The command returns the new value of the variable. You can read the value of a variable by invoking set with only a single argument:

```
set x
```

You don't need to declare variables in Tcl: a variable is created automatically the first time it is set. Tcl variables don't have types: any variable can hold any value.
To use the value of a variable in a command, use variable substitution as in the following example:

```
expr $x*3
```

When a $ appears in a command, Tcl treats the letters and digits following it as a variable name, and substitutes the value of the variable in place of the name. In this example, the actual argument received by the expr command will be 32*3 (assuming that variable x was set as in the previous example). You can use variable substitution in any word of any command, or even multiple times within a word:

```
set cmd expr
set x 11
$cmd $x*$x
```

Command substitution
You can also use the result of one command in an argument to another command. This is called command substitution:

```
set a 44
set b [expr $a*4]
```

When a [appears in a command, Tcl treats everything between it and the matching] as a nested Tcl command. Tcl evaluates the nested command and substitutes its result into the enclosing command in place of the bracketed text. In the example above the second argument of the second set command will be 176.
Quotes and braces
Double-quotes allow you to specify words that contain spaces. For example,

consider the following script:

```
set x 24
set y 18
set z "$x + $y is [expr $x + $y]"
```

After these three commands are evaluated variable z will have the value 24 + 18 is 42. Everything between the quotes is passed to the set command as a single word. Note that (a) command and variable substitutions are performed on the text between the quotes, and (b) the quotes themselves are not passed to the command. If the quotes were not present, the set command would have received 6 arguments, which would have caused an error.
Curly braces provide another way of grouping information into words. They are different from quotes in that no substitutions are performed on the text between the curly braces:

```
set z {$x + $y is [expr $x + $y]}
```

This command sets variable z to the value "$x + $y is [expr $x + $y]".
Control structures
Tcl provides a complete set of control structures including commands for conditional execution, looping, and procedures. Tcl control structures are just commands that take Tcl scripts as arguments. The example below creates a Tcl procedure called power, which raises a base to an integer power:

```
proc power {base p} {
set result 1
while {$p > 0} {
set result [expr $result * $base]
set p [expr $p - 1]
}
return $result
}
```

This script consists of a single command, proc. The proc command takes three arguments: the name of a procedure, a list of argument names, and the body of the procedure, which is a Tcl script. Note that everything between the curly brace at the end of the first line and the curly brace on the last line is passed verbatim to proc as a single argument. The proc command creates a new Tcl command named power that takes two arguments. You can then invoke power with commands like the following:

```
power 2 6
power 1.15 5
```

When power is invoked, the procedure body is evaluated. While the body is executing it can access its arguments as variables: base will hold the first argument and p will hold the second.
The body of the power procedure contains three Tcl commands: set, while, and return. The while command does most of the work of the procedure. It takes two arguments, an expression ($p > 0) and a body, which is another Tcl script. The while command evaluates its expression argument using rules similar to those of the C programming language and if the result is true (nonzero) then it evaluates the body as a Tcl script. It repeats this process over and over until eventually the expression evaluates to false (zero). In this case the body of the while command multiplied the result value by base and then decrements p. When p reaches zero the result contains the desired power of base. The return command causes the procedure to exit with the value of variable result as the procedure's result.

Where do commands come from?
As you have seen, all of the interesting features in Tcl are represented by commands. Statements are commands, expressions are evaluated by executing commands, control structures are commands, and procedures are commands.
Tcl commands are created in three ways. One group of commands is provided by the Tcl interpreter itself. These commands are called builtin commands. They include all of the commands you have seen so far and many more (see below). The builtin commands are present in all Tcl applications.
The second group of commands is created using the Tcl extension mechanism. Tcl provides APIs that allow you to create a new command by writing a command procedure in C or C++ that implements the command. You then register the command procedure with the Tcl interpreter by telling Tcl the name of the command that the procedure implements. In the future, whenever that particular name is used for a Tcl command, Tcl will call your command procedure to execute the command. The builtin commands are also implemented using this same extension mechanism; their command procedures are simply part of the Tcl library.
When Tcl is used inside an application, the application incorporates its key features into Tcl using the extension mechanism. Thus the set of available Tcl commands varies from application to application. There are also numerous extension packages that can be incorporated into any Tcl application. One of the best known extensions is Tk, which provides powerful facilities for building graphical user interfaces. Other extensions provide object-oriented programming, database access, more graphical capabilities, and a variety of other features. One of Tcl's greatest advantages for building integration applications is the ease with which it can be extended to incorporate new features or communicate with other resources.
The third group of commands consists of procedures created with the proc command, such as the power command created above. Typically, extensions are used for lower-level functions where C programming is convenient, and procedures are used for higher-level functions where it is easier to write in Tcl.

Wired TCL Script Components
Create the event scheduler
Open new files & turn on the tracing
Create the nodes
Setup the links
Configure the traffic type (e.g., TCP, UDP, etc)
Set the time of traffic generation (e.g., CBR, FTP)
Terminate the simulation
NS Simulator Preliminaries.
Initialization and termination aspects of the ns simulator.
Definition of network nodes, links, queues and topology.
Definition of agents and of applications.
The nam visualization tool.

Tracing and random variables.
Features of NS2
NS2 can be employed in most unix systems and windows. Most of the NS2 code is in C++. It uses TCL as its scripting language, Otcl adds object orientation to TCL.NS(version 2) is an object oriented, discrete event driven network simulator that is freely distributed and open source.
• Traffic Models: CBR, VBR, Web etc
• Protocols: TCP, UDP, HTTP, Routing algorithms,MAC etc
• Error Models: Uniform, bursty etc
• Misc: Radio propagation, Mobility models , Energy Models
• Topology Generation tools
• Visualization tools (NAM), Tracing
Structure of NS
● NS is an object oriented discrete event simulator
– Simulator maintains list of events and executes one event after another
– Single thread of control: no locking or race conditions
● Back end is C++ event scheduler
– Protocols mostly
– Fast to run, more control
• Front end is OTCL
Creating scenarios, extensions to C++ protocols
fast to write and change
Platforms
It can be employed in most unix systems(FreeBSD, Linux, Solaris) and Windows.
Source code
Most of NS2 code is in C++
Scripting language
It uses TCL as its scripting language OTcl adds object orientation to TCL.

Protocols implemented in NS2
Transport layer(Traffic Agent) – TCP, UDP
Network layer(Routing agent)

Interface queue – FIFO queue, Drop Tail queue, Priority queue
Logic link contol layer – IEEE 802.2, AR
How to use NS2
Design Simulation – Determine simulation scenario
Build ns-2 script using tcl.
Run simulation
Simulation with NS2
Define objects of simulation.
Connect the objects to each other
Start the source applications. Packets are then created and are transmitted through network.
Exit the simulator after a certain fixed time.
NS programming Structure

- Create the event scheduler
- Turn on tracing
- Create network topology
- Create transport connections
- Generate traffic
- Insert errors

1. Implement three nodes point – to – point network with duplex links between them.

```
#Create Simulator
set ns [new Simulator]
#Open Trace file and NAM file
set ntrace [open prog1.tr w]
$ns trace-all $ntrace
set namfile [open prog1.nam w]
$ns namtrace-all $namfile
#Finish Procedure
proc Finish {} {
global ns ntrace namfile
#Dump all the trace data and close the files
$ns flush-trace
close $ntrace
close $namfile
#Execute the nam animation file
exec nam prog1.nam &
#Show the number of packets dropped
exec echo "The number of packet drops is " &
```

```
exec grep -c "^d" prog1.tr &
exit 0
}
#Create 3 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
#Label the nodes
$n0 label "TCP Source"
$n2 label "Sink"
#Set the color
$ns color 1 blue
#Create Links between nodes
#You need to modify the bandwidth to observe the variation in packet drop
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
#Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
#Set Queue Size
#You can modify the queue length as well to observe the variation in packet drop
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10
```

```
#Set up a Transport layer connection.
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 up an Application layer Traffic
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
#Schedule Events
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "Finish"
#Run the Simulation
$ns run
```

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.

```
#Create Simulator
set ns [new Simulator]
#Use colors to differentiate the traffic
$ns color 1 Blue
$ns color 2 Red
#Open trace and NAM trace file
set ntrace [open prog3.tr w]
$ns trace-all $ntrace
set namfile [open prog3.nam w]
$ns namtrace-all $namfile
#Finish Procedure
proc Finish {} {
global ns ntrace namfile
#Dump all trace data and close the file
$ns flush-trace
close $ntrace
close $namfile
#Execute the nam animation file
exec nam prog3.nam &
#Find the number of ping packets dropped
puts "The number of ping packets dropped are "
exec grep "^d" prog3.tr | cut -d " " -f 5 | grep -c "ping" &
exit 0
}
#Create six nodes
for {set i 0} {$i < 6} {incr i} {
set n($i) [$ns node]
}
#Connect the nodes
for {set j 0} {$j < 5} {incr j} {
$ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail
}
#Define the recv function for the class 'Agent/Ping'
Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received ping answer from $from with round trip time $rtt
ms"
}
#Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]
$p0 set class_ 1
$ns attach-agent $n(0) $p0
set p1 [new Agent/Ping]
$p1 set class_ 1
$ns attach-agent $n(5) $p1
$ns connect $p0 $p1
```

```
#Set queue size and monitor the queue
#Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5
#Create Congestion
#Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0
#Apply CBR traffic over TCP
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0
#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"
#Run the Simulation
$ns run
```

3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [new Simulator]
$ns color 1 Red
$ns color 2 Blue
set na [open Lab3.nam w]
$ns namtrace-all $na
set nt [open Lab3.tr w]
$ns trace-all $nt
set ng1 [open tcp1.xg w]
set ng2 [open tcp2.xg w]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
```

```
$ns make-lan "$n0 $n1 $n2" 1Mb 10ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n3 $n4 $n5" 2Mb 10ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
set tcp1 [new Agent/TCP]
set tcp2 [new Agent/TCP]
set cbr1 [new Application/Traffic/CBR]
set cbr2 [new Application/Traffic/CBR]
$ns attach-agent $n4 $tcp1
$cbr1 attach-agent $tcp1
$ns attach-agent $n1 $tcp2
$cbr2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns attach-agent $n5 $sink2
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
proc End {} {
global ns na nt
$ns flush-trace
close $na
close $nt
exec nam Lab3.nam &
exec xgraph tcp1.xg tcp2.xg &
exit 0
}
proc Draw {Agent File} {
global ns
set Cong [$Agent set cwnd_]
set Time [$ns now]
puts $File "$Time $Cong"
$ns at [expr $Time+0.01] "Draw $Agent $File"
}
$ns at 0.0 "$cbr1 start"
$ns at 0.7 "$cbr2 start"
$ns at 0.0 "Draw $tcp1 $ng1"
$ns at 0.0 "Draw $tcp2 $ng2"
$ns at 10.0 "End"
$ns run
```

4. Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

```
set ns [new Simulator]
set na [open Lab4.nam w]
$ns namtrace-all-wireless $na 500 500
set nt [open Lab4.tr w]
$ns trace-all $nt
set topo [new Topography]
$topo load_flatgrid 500 500
$ns node-config -adhocRouting DSDV
$ns node-config -llType LL
$ns node-config -macType Mac/802_11
$ns node-config -ifqType Queue/DropTail
$ns node-config -ifqLen 50
$ns node-config -phyType Phy/WirelessPhy
$ns node-config -channelType Channel/WirelessChannel
$ns node-config -propType Propagation/TwoRayGround
$ns node-config -antType Antenna/OmniAntenna
$ns node-config -topoInstance $topo


$ns node-config -agentTrace ON
$ns node-config -routerTrace ON
create-god 4
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$n0 set X_ 250.0
$n0 set Y_ 250.0
$n0 set Z_ 0.0
$n1 set X_ 200.0
$n1 set Y_ 250.0
$n1 set Z_ 0.0
$n2 set X_ 250.0
$n2 set Y_ 250.0
$n2 set Z_ 0.0
$n3 set X_ 250.0
$n3 set Y_ 250.0
$n3 set Z_ 0.0
$ns at 0.0 "$n0 setdest 400.0 300.0 50.0"
$ns at 0.0 "$n1 setdest 50.0 100.0 20.0"
$ns at 0.0 "$n2 setdest 75.0 180.0 5.0"
$ns at 0.0 "$n3 setdest 100.0 100.0 25.0"
set tcp1 [new Agent/TCP]
$ns attach-agent $n0 $tcp1
```

```
set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n3 $sink2
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $tcp1
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $tcp2
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
proc End {} {
global ns nt na
$ns flush-trace
close $na
close $nt
exec nam Lab4.nam &
}
$ns at 0.0 "$cbr1 start"
$ns at 0.0 "$cbr2 start"
$ns at 10.0 "End"
$ns run
```

AWK Code

```
|
BEGIN{Num_of_pkts=0;}
{
if ($1 == "r" && $3 == "_1_" && $4 == "AGT" && $7 == "tcp")
{
Num_of_pkts = Num_of_pkts + $8;
}
}
END{
Throughput = Num_of_pkts * 8 / $2 /1000000;
printf("\n\n\tThroughput = %fbpms\n\n\n",Throughput);
}
```

5. Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.

```
set bwDL(gsm) 9600
set bwUL(gsm) 9600
set propDL(gsm) .500
set propUL(gsm) .500
set buf(gsm) 10
set ns [new Simulator]
set nt [open Lab5.tr w]
$ns trace-all $nt
set nodes(c1) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(c2) [$ns node]
proc cell_topo {} {
global ns nodes
$ns duplex-link $nodes(c1) $nodes(bs1) 3Mbps 10ms DropTail
$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
$ns duplex-link $nodes(bs2) $nodes(c2) 3Mbps 50ms DropTail
}
switch gsm {
gsm -
gprs -
umts {cell_topo}
}
$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL(gsm) simplex
$ns bandwidth $nodes(ms) $nodes(bs1) $bwUL(gsm) simplex
$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL(gsm) simplex
$ns bandwidth $nodes(ms) $nodes(bs2) $bwUL(gsm) simplex
$ns delay $nodes(bs1) $nodes(ms) $propDL(gsm) simplex
$ns delay $nodes(ms) $nodes(bs1) $propDL(gsm) simplex
$ns delay $nodes(bs2) $nodes(ms) $propDL(gsm) simplex
$ns delay $nodes(ms) $nodes(bs2) $propDL(gsm) simplex
$ns queue-limit $nodes(bs1) $nodes(ms) $buf(gsm)
$ns queue-limit $nodes(ms) $nodes(bs1) $buf(gsm)
$ns queue-limit $nodes(bs2) $nodes(ms) $buf(gsm)
$ns queue-limit $nodes(ms) $nodes(bs2) $buf(gsm)
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]
set tcp [new Agent/TCP]
$ns attach-agent $nodes(c1) $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $nodes(c2) $sink
set ftp [new Application/FTP]
```

```
$ftp attach-agent $tcp
$ns connect $tcp $sink
proc End {} {
global ns nt
$ns flush-trace
close $nt
exec awk -f Lab5.awk Lab5.tr &
exec xgraph -P -bar -x TIME -y DATA gsm.xg &
exit 0
}
$ns at 0.0 "$ftp start"
$ns at 10.0 "End"
$ns run
```

AWK Code

```
|
BEGIN {Total_no_of_pkts=0;}
{
if($1 == "r")
{
Total_no_of_pkts = Total_no_of_pkts + $6;
printf("%f %d\n",$2,Total_no_of_pkts) >> "gsm.xg"
}
}
END{}
```

6. Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

```
set bwDL(cdma) 384000
set bwUL(cdma) 64000
set propDL(cdma) .150
set propUL(cdma) .150
set buf(cdma) 20
set ns [new Simulator]
set nt [open Lab6.tr w]
$ns trace-all $nt
set nodes(c1) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(c2) [$ns node]
proc cell_topo {} {
global ns nodes
$ns duplex-link $nodes(c1) $nodes(bs1) 3Mbps 10ms DropTail
$ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
$ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
$ns duplex-link $nodes(bs2) $nodes(c2) 3Mbps 50ms DropTail
}
switch umts {
umts {cell_topo}
}
$ns bandwidth $nodes(bs1) $nodes(ms) $bwDL(cdma) simplex
$ns bandwidth $nodes(ms) $nodes(bs1) $bwUL(cdma) simplex
$ns bandwidth $nodes(bs2) $nodes(ms) $bwDL(cdma) simplex
$ns bandwidth $nodes(ms) $nodes(bs2) $bwUL(cdma) simplex
$ns delay $nodes(bs1) $nodes(ms) $propDL(cdma) simplex
$ns delay $nodes(ms) $nodes(bs1) $propDL(cdma) simplex
$ns delay $nodes(bs2) $nodes(ms) $propDL(cdma) simplex
$ns delay $nodes(ms) $nodes(bs2) $propDL(cdma) simplex
$ns queue-limit $nodes(bs1) $nodes(ms) $buf(cdma)
$ns queue-limit $nodes(ms) $nodes(bs1) $buf(cdma)
$ns queue-limit $nodes(bs2) $nodes(ms) $buf(cdma)
$ns queue-limit $nodes(ms) $nodes(bs2) $buf(cdma)
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
$ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]
set tcp [new Agent/TCP]
$ns attach-agent $nodes(c1) $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $nodes(c2) $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns connect $tcp $sink
```

```
proc End {} {
global ns nt
$ns flush-trace
close $nt
exec awk -f Lab6.awk Lab6.tr &
exec xgraph -P -bar -x TIME -y DATA cdma.xg &
exit 0
}
$ns at 0.0 "$ftp start"
$ns at 10.0 "End"
$ns run
```

AWK Code

```
|
BEGIN {Total_no_of_pkts=0;}
{
if($1 == "r")
{
Total_no_of_pkts = Total_no_of_pkts + $6;
printf("%f %d\n",$2,Total_no_of_pkts) >> "cdma.xg"
}
}
END{}
```

PART B

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

```
import java.util.*;

class crc {
	public static void main(String args[]) {
		Scanner scan = new Scanner(System.in);
		int n;

		System.out.println("Enter the size of the data:");
		n = scan.nextInt();
		int data[] = new int[n];
		System.out.println("Enter the data, bit by bit:");
		for(int i=0 ; i < n ; i++) {

			data[i] = scan.nextInt();
		}

		System.out.println("Enter the size of the divisor:");
		n = scan.nextInt();
		int divisor[] = new int[n];
		System.out.println("Enter the divisor, bit by bit:");
		for(int i=0 ; i < n ; i++) {
			divisor[i] = scan.nextInt();
		}

		int remainder[] = divide(data, divisor);
		for(int i=0 ; i < remainder.length-1 ; i++) {
			System.out.print(remainder[i]);
		}
		System.out.println("\nThe CRC code generated is:");

		for(int i=0 ; i < data.length ; i++) {
			System.out.print(data[i]);
		}
		for(int i=0 ; i < remainder.length-1 ; i++) {
			System.out.print(remainder[i]);
		}
		System.out.println();

		int sent_data[] = new int[data.length + remainder.length - 1];
		System.out.println("Enter the data to be sent:");
		for(int i=0 ; i < sent_data.length ; i++) {
			System.out.println("Enter bit number " + (sent_data.length-i)
                                                    + ":");
			sent_data[i] = scan.nextInt();
		}
		receive(sent_data, divisor);
```

```
}

static int[] divide(int old_data[], int divisor[]) {
        int remainder[] , i;
        int data[] = new int[old_data.length + divisor.length];
        System.arraycopy(old_data, 0, data, 0, old_data.length);
        remainder = new int[divisor.length];
        System.arraycopy(data, 0, remainder, 0, divisor.length);

        for(i=0 ; i < old_data.length ; i++) {
                System.out.println((i+1) + ".) First data bit is : "
                                                        + remainder[0]);
                System.out.print("Remainder : ");
                if(remainder[0] == 1) {
                        for(int j=1 ; j < divisor.length ; j++) {
                                remainder[j-1] = exor(remainder[j], divisor[j]);
                                System.out.print(remainder[j-1]);
                        }
                }
                else {
                        for(int j=1 ; j < divisor.length ; j++) {
                                remainder[j-1] = exor(remainder[j], 0);
                                System.out.print(remainder[j-1]);
                        }
                }
                remainder[divisor.length-1] = data[i+divisor.length];
                System.out.println(remainder[divisor.length-1]);
        }
        return remainder;
}

static int exor(int a, int b) {
        if(a == b) {
                return 0;
        }
        return 1;
}

static void receive(int data[], int divisor[]) {
        int remainder[] = divide(data, divisor);
        for(int i=0 ; i < remainder.length ; i++) {
                if(remainder[i] != 0) {
                        System.out.println("There is an error in received data...");
                        return;
                }
        }
        System.out.println("Data was received without any error.");
}
```

```
}
```

8. Write a program to find the shortest path between vertices using bellman-ford Algorithm

```
import java.util.Scanner;
public class BellmanFord
{
private int D[];
private int num_ver;
public static final int MAX_VALUE = 999;
public BellmanFord(int num_ver)
{
this.num_ver = num_ver;
D = new int[num_ver + 1];
}
public void BellmanFordEvaluation(int source, int A[][])
{
for (int node = 1; node <= num_ver; node++)
{
D[node] = MAX_VALUE;
}
D[source] = 0;
for (int node = 1; node <= num_ver - 1; node++)
{
for (int sn = 1; sn <= num_ver; sn++)
{
for (int dn = 1; dn <= num_ver; dn++)
{
if (A[sn][dn] != MAX_VALUE)
{
if (D[dn] > D[sn]+ A[sn][dn])
D[dn] = D[sn] + A[sn][dn];
}
}
}
}
for (int sn = 1; sn <= num_ver; sn++)
{
for (int dn = 1; dn <= num_ver; dn++)
{
if (A[sn][dn] != MAX_VALUE)
{
if (D[dn] > D[sn]+ A[sn][dn])
System.out.println("The Graph contains negative egde cycle");
}
}
}
```

```
for (int vertex = 1; vertex <= num_ver; vertex++)
{
System.out.println("distance of source " + source + " to "+ vertex + "is" + D[vertex]);
}
}
public static void main(String[ ] args)
{
int num_ver = 0;
int source;
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the number of vertices");
num_ver = scanner.nextInt();
int A[][] = new int[num_ver + 1][num_ver + 1];
System.out.println("Enter the adjacency matrix");
for (int sn = 1; sn <= num_ver; sn++)
{
for (int dn = 1; dn <= num_ver; dn++)
{
A[sn][dn] = scanner.nextInt();
if (sn == dn)
{
A[sn][dn] = 0;
continue;
}
if (A[sn][dn] == 0)
{
A[sn][dn] = MAX_VALUE;
}
}
}
System.out.println("Enter the source vertex");
source = scanner.nextInt();
BellmanFord b = new BellmanFord (num_ver);
b.BellmanFordEvaluation(source, A);
scanner.close();
}
}
```

9. 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.

tcp program server side

```
import java.net.*;
import java.io.*;
public class server1
{
  public static void main(String args[]) throws Exception
  {                    // establishing the connection with the server
    ServerSocket sersock = new ServerSocket(4000);
    System.out.println("Server ready for connection");
    Socket sock = sersock.accept();          // binding with port: 4000
    System.out.println("Connection is successful");

                        // reading the file name from client
    InputStream istream = sock.getInputStream( );
    BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));
    String fname = fileRead.readLine( );
                      // reading file contents
    BufferedReader contentRead = new BufferedReader(new FileReader(fname) );

                    // keeping output stream ready to send the contents
    OutputStream ostream = sock.getOutputStream( );
    PrintWriter pwrite = new PrintWriter(ostream, true);

    String str;
    while((str = contentRead.readLine()) !=  null) // reading line-by-line from file
    {
pwrite.println(str);        // sending each line to client
    }

    sock.close();  sersock.close();       // closing network sockets
    pwrite.close();  fileRead.close(); contentRead.close();
  }
}
```

tcp program client side

```
import java.net.*;
import java.io.*;
public class client1
{
  public static void main( String args[ ] ) throws Exception
  {
    Socket sock = new Socket( "127.0.0.1", 4000);

            // reading the file name from keyboard. Uses input stream
```

```
    System.out.print("Enter the file name");
    BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));
    String fname = keyRead.readLine();

      // sending the file name to server. Uses PrintWriter
    OutputStream  ostream = sock.getOutputStream( );
    PrintWriter pwrite = new PrintWriter(ostream, true);
    pwrite.println(fname);
                // receiving the contents from server.  Uses input stream
    InputStream istream = sock.getInputStream();
    BufferedReader socketRead = new BufferedReader(new InputStreamReader(istream));

    String str;
    while((str = socketRead.readLine())  !=  null) // reading line-by-line
    {
       System.out.println(str);
    }
    pwrite.close(); socketRead.close(); keyRead.close();
  }
}
```

10. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.

```
import java.net.*;
   import java.io.*;
   class userver {
   public static void main(String args[])throws Exception{
   Socket s=new Socket("localhost",3333);
   DataInputStream din=new DataInputStream(s.getInputStream());
   DataOutputStream dout=new DataOutputStream(s.getOutputStream());
   BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

   String str="",str2="";
   while(!str.equals("stop")){
   str=br.readLine();
   dout.writeUTF(str);
   dout.flush();
   str2=din.readUTF();
   System.out.println("client says: "+str2);
   }

   dout.close();
   s.close();
   }}
```

udp program client side

```
 import java.net.*;
   import java.io.*;
   class uclient{
   public static void main(String args[])throws Exception{
   ServerSocket ss=new ServerSocket(3333);
   Socket s=ss.accept();
   DataInputStream din=new DataInputStream(s.getInputStream());
   DataOutputStream dout=new DataOutputStream(s.getOutputStream());
   BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

   String str="",str2="";
   while(!str.equals("stop")){
   str=din.readUTF();
   System.out.println("server says: "+str);
   str2=br.readLine();
   dout.writeUTF(str2);
   dout.flush();
   }
   din.close();
   s.close();
   ss.close();
   }}
```

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

```
import java.io.DataInputStream;

import java.io.IOException;

import java.math.BigInteger;

import java.util.Random;


public class rsa

{

   private BigInteger p;

   private BigInteger q;

   private BigInteger N;

   private BigInteger phi;

   private BigInteger e;

   private BigInteger d;

   private int        bitlength = 1024;

   private Random     r;


   public rsa()

   {

      r = new Random();

      p = BigInteger.probablePrime(bitlength, r);

      q = BigInteger.probablePrime(bitlength, r);

      N = p.multiply(q);

      phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));

      e = BigInteger.probablePrime(bitlength / 2, r);
```

```
    while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
    {
        e.add(BigInteger.ONE);
    }
    d = e.modInverse(phi);
}

public rsa(BigInteger e, BigInteger d, BigInteger N)
{
    this.e = e;
    this.d = d;
    this.N = N;
}

@SuppressWarnings("deprecation")
public static void main(String[] args) throws IOException
{
    RSA rsa = new RSA();
    DataInputStream in = new DataInputStream(System.in);
    String teststring;
    System.out.println("Enter the plain text:");
    teststring = in.readLine();
    System.out.println("Encrypting String: " + teststring);
    System.out.println("String in Bytes: "
```

```
        + bytesToString(teststring.getBytes()));
    // encrypt
    byte[] encrypted = rsa.encrypt(teststring.getBytes());
    // decrypt
    byte[] decrypted = rsa.decrypt(encrypted);
    System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
    System.out.println("Decrypted String: " + new String(decrypted));
}

private static String bytesToString(byte[] encrypted)
{
    String test = "";
    for (byte b : encrypted)
    {
        test += Byte.toString(b);
    }
    return test;
}

// Encrypt message
public byte[] encrypt(byte[] message)
{
    return (new BigInteger(message)).modPow(e, N).toByteArray();
}
```

```
    // Decrypt message
    public byte[] decrypt(byte[] message)
    {
        return (new BigInteger(message)).modPow(d, N).toByteArray();
    }
}
```

12. Write a program for congestion control using leaky bucket algorithm.

```
import java.io.*;
import java.util.*;

class Queue
{
int q[],f=0,r=0,size;
void insert(int n)
{
Scanner in = new Scanner(System.in);
q=new int[10];
for(int i=0;i<n;i++)
{
System.out.print("\nEnter " + i + " element: ");
int ele=in.nextInt();

if(r+1>10)
{
System.out.println("\nQueue is full \nLost Packet: "+ele);
break;
}
else
{
r++;
q[i]=ele;
}
}
}

void delete()
{
Scanner in = new Scanner(System.in);
Thread t=new Thread();
if(r==0)
System.out.print("\nQueue empty ");

	else
{
	for(int i=f;i<r;i++)
{
try
{
		t.sleep(1000);
}
catch(Exception e){}
System.out.print("\nLeaked Packet: "+q[i]);
```

```
f++;
}
}
System.out.println();
      }

}

class Licky extends Thread
{
public static void main(String ar[])throws Exception
{
Queue q=new Queue();
Scanner src=new Scanner(System.in);
System.out.println("\nEnter the packets to be sent:");
int size=src.nextInt();
q.insert(size);
q.delete();

}
}
```

Viva Questions

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

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