

Kammavarisangham

K.S.INSTITUTE OF TECHNOLOGY

(Approved by A.I.C.T.E affiliated to VTU Belgaum)

#14, Raghuvanahalli, kanakapura main road,Bangalore-560109



Department of Electronics & Communication Engg.

Name of the Lab: COMMUNICATION LAB II

Course Code: 21ECL55



K. S. INSTITUTE OF TECHNOLOGY

VISION

“To impart quality technical education with ethical values, employable skills and research to achieve excellence”.

MISSION

- To attract and retain highly qualified, experienced & committed faculty.
- To create relevant infrastructure.
- Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence.
- To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society.



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

VISION:

“To achieve excellence in academics and research in Electronics & Communication Engineering to meet societal need”.

MISSION:

- To impart quality technical education with the relevant technologies to produce industry ready engineers with ethical values.
- To enrich experiential learning through active involvement in professional clubs & societies.
- To promote industry-institute collaborations for research & development.



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES (PEO'S)

PEO1: To Excel in a professional career by acquiring domain knowledge.

PEO2: To pursue higher Education & research by adopting technological innovations by continuous learning through professional bodies and clubs.

PEO3: To inculcate effective communication, team work, ethics, entrepreneurship skills and leadership qualities.

PROGRAM SPECIFIC OUTCOMES (PSO'S)

PSO1: Graduate should be able to understand the fundamentals in the field of Electronics & Communication and apply the same to various areas like Signal processing, embedded systems, Communication & Semiconductor technology.

PSO2: Graduate will demonstrate the ability to design, develop solutions for Problems in Electronics & Communication Engineering using hardware and software tools with social concerns.



K S INSTITUTE OF TECHNOLOGY

PROGRAM OUTCOMES (PO'S)

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 in independent and life-long learning in the broadest context of technological change.



K S INSTITUTE OF TECHNOLOGY, BENGALURU

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

RUBRICS for Evaluation in Laboratories - 2021 Scheme

Continuous Internal Evaluation (CIE)-50 Marks=30+20			
Record, Observation & Viva: 30 Marks			
Record	Evaluation criteria		
	Good	Average	Poor
10 Marks	The Record meets all aspects of assessment- Timeliness, contents, correctness, completeness & neatness.	The Record partially meets all aspects of assessment- Timeliness, contents, correctness, completeness & neatness.	The Record written poorly, does not meet all aspects of assessment- Timeliness, contents, correctness, completeness & neatness.
	9 to 10 Marks	5 to 8 Marks	0 to 4 Marks
Observation & Conduction	Evaluation criteria		
	Good	Average	Poor
15 Marks	The Observation meets all aspects of assessment- Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is satisfactory.	The Observation partially meets all aspects of assessment- Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is partially satisfactory.	The Observation poorly written and does not meet all aspects of assessment- Timeliness, contents, correctness, completeness & neatness. Conduction of experiment is not satisfactory.
	10 to 15 Marks	5 to 9 Marks	0 to 4 Marks
Viva	Evaluation criteria		
	Good	Average	Poor
5 Marks	All questions answered Correctly	Answers are partially correct	Poorly answered
	5 Marks	3 to 4 Marks	0 to 2 Marks
Test-10+10=20 Marks (two tests)			
Write-up: 20% of maximum marks		Conduction: 40% of maximum marks	Viva: 40% of maximum marks

Note: Each test should be conducted for 100 marks and reduce to 10

Communication Lab II			
Course Code	21ECL55	CIE Marks	50
Teaching Hours/Week (L: T: P: S)	0:0:2:0	SEE Marks	50
Credits	1	Exam Hours	3
Course objectives:			
This laboratory course enables students to			
<ul style="list-style-type: none">• Design and demonstrate communication circuits for different digital modulation techniques.• To simulate Source coding Algorithms using C/C++/ MATLAB code.• To simulate Error correcting and detecting codes using C/C++/ MATLAB code.• Simulate the networking concepts and protocols using C/C++/ Network simulation tool.• Understand entropies and mutual information of different communication channels.			
Sl.No.	Experiments		
Implement the following using discrete components			
1	FSK generation and detection		
2	PSK generation and detection		
3	DPSK Transmitter and receiver		
4	QPSK Transmitter and Receiver		
Implement the following in C/C++/MATLAB/Scilab/Python or any other Suitable software			
5	Write a program to encode binary data using Huffman code and decode it.		
6	Write a program to encode binary data using a (7,4) Hamming code and decode it.		
7	Write a program to encode binary data using a ((3,1,2)/suitably designed) Convolution code and decode it.		
8	For a given data, use CRC-CCITT polynomial to obtain the CRC code. Verify the program for the cases a) Without error b) With error		
Implement the following algorithms in C/C++/MATLAB/Network simulator			
9	Write a program for congestion control using leaky bucket algorithm.		
10	Write a program for distance vector algorithm to find suitable path for transmission.		
11	Write a program for flow control using sliding window protocols.		
12	Configure a simple network (Bus/star) topology using simulation software OR Configure a simple network (Ring/Mesh) topology using simulation software.		
Demonstration Experiments (For CIE)			
13	Configure and simulate simple Wireless Local Area network.		
14	Simulate the BER performance of (2, 1, 3) binary convolutional code with generator sequences $g(1) = (1\ 0\ 1\ 1)$ and $g(2) = (1\ 1\ 1\ 1)$ on AWGN channel. Use QPSK modulation scheme. Channel decoding is to be performed through Viterbi decoding. Plot the bit error rate versus SNR (dB), i.e. $P_{e,b}$ versus E_b/N_0 . Consider binary input vector of size 3 lakh bits. Also find the coding gain.		
15	Simulate the BER performance of (7, 4) Hamming code on AWGN channel. Use QPSK modulation scheme. Channel decoding is to be performed through maximum-likelihood decoding. Plot the bit error rate versus SNR (dB), i.e. $P_{e,b}$ versus E_b/N_0 . Consider binary input vector of size 5 lakh bits. Use the following parity check matrix for the (7, 4) Hamming code. Also find the coding gain.		
	$H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$		
16	Simulate the BER performance of rate 1/3 Turbo code. Turbo encoder uses two recursive systematic encoders with $() = [1, \frac{1+D}{1+D+D^2+D^3}]$ and pseudo-random interleaver. Use QPSK modulation scheme. Channel decoding is to be performed through maximum a-posteriori (MAP) decoding algorithm. Plot the bit error rate versus SNR (dB), i.e. $P_{e,b}$ versus E_b/N_0 . Consider binary input vector of size of around 3 lakh bits and the block length as 10384 bits. Also find the coding gain.		

Course outcomes (Course Skill Set):

On the completion of this laboratory course, the students will be able to:

1. Design and test the digital modulation circuits and display the waveforms.
2. To Implement the source coding algorithm using C/C++/ MATLAB code.
3. To Implement the Error Control coding algorithms using C/C++/ MATLAB code.
4. Illustrate the operations of networking concepts and protocols using C programming and networksimulators.

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). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th week of the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

SEE marks for the practical course is 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners).

Change of experiment is allowed only once and 15% Marks allotted to the procedure part to be made zero.

The duration of SEE is 03 hours.

Rubrics suggested in Annexure-II of Regulation book

Suggested Learning Resources:

1. Simon Haykin, "Digital Communication Systems", John Wiley & sons, First Edition, 2014, ISBN 978-0-471-64735-5.
2. K Sam Shanmugam, "Digital and analog communication systems", John Wiley India Pvt. Ltd, 1996.
3. Forouzan, "Data Communications and Networking", 5th Edition, McGraw Hill, 2013, ISBN: 1-25-906475-3.



K.S. INSTITUTE OF TECHNOLOGY, BANGALORE - 560109
DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

Course: Communication laboratory II		Course Code: 21ECL55		Type: Core
Course Incharge : Saleem S Tevaramani , Sangeetha V, Priyadarshini				
No of Hours per week				
Theory (Lecture Class)	Practical/Field Work/Allied Activities	Total/Week	Total teaching hours	
0	3	3	40	
Marks				
CIE	SEE	Total	Credits	
50	50	100	1	
<u>Aim/Objective of the Course:</u>				
Course Learning Outcomes: After completing the course, the students will be able to,				Bloom's Level
21ECL55.1	Make use of the concepts of Digital modulation & regeneration schemes and Utilize it to display the waveform.			Applying (K3)
21ECL55.2	Build the concepts of source coding techniques and develop a program to encode and decode.			Applying (K3)
21ECL55.3	Develop the program for Error control coding and identify the error in the received codeword.			Applying (K3)
21ECL55.4	Apply the knowledge of congestion and flow control and identify the suitable path for transmission			Applying (K3)
21ECL55.5	Make use of the concepts for networks and develop a topology.			Applying (K3)

CO - PO MAPPING details for Communication lab-II															
CO 21ECL55	Bloom's Level	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
21ECL55.1	K3	3	3	2	-	2	-	-	-	2	2	-	2	2	2
21ECL55.2	K3	3	3	2	-	2	-	-	-	2	2	-	2	2	2
21ECL55.3	K3	3	3	2	-	2	-	-	-	2	2	-	2	2	2
21ECL55.4	K3	3	3	2	-	2	-	-	-	2	2	-	2	2	2
21ECL55.5	K3	3	3	2	-	2	-	-	-	2	2	-	-	2	2
21ECL55		3	3	2		2				2	2	-	2	2	2

CONTENTS

Sl. No.	Experiment	Page no.
1.	FSK generation and detection	
2.	PSK generation and detection	
3.	DPSK Transmitter and receiver	
4.	QPSK Transmitter and Receiver	
Implement the following in C/C++/MATLAB/Scilab/Python or any other Suitable software		
5.	Write a program to encode binary data using Huffman code and decode it.	
6.	Write a program to encode binary data using a (7,4) Hamming code and decode it.	
7.	Write a program to encode binary data using a ((3,1,2)/suitably designed) Convolution code and decode it.	
8.	For a given data, use CRC-CCITT polynomial to obtain the CRC code. Verify the program for the cases a) Without error b) With error	
Implement the following algorithms in C/C++/MATLAB/Network simulator		
9.	Write a program for congestion control using leaky bucket algorithm.	
10.	Write a program for distance vector algorithm to find suitable path for transmission.	
11.	Write a program for flow control using sliding window protocols.	
12.	Configure a simple network (Bus/star) topology using simulation software OR Configure a simple network (Ring/Mesh) topology using simulation software.	

EXPERIMENT: 1

FREQUENCY SHIFT KEYING (FSK) GENERATION AND DETECTION

AIM: To conduct an experiment to study frequency shift keying modulation and demodulation

COMPONENTS REQUIRED:

Sl.No	Apparatus	Range	Quantity
1	IC 1458		2
2	Transistor	SL 100 SK 100	1 1
3	Diode	OA 79	1
4	Resistors	As Per Design	16
5	Potentiometer	10K Ω	1
6	Capacitor	As Per Design	3

THEORY:

In binary frequency shift keying the frequency of the carrier is shifted according to the binary symbol. The phase of the carrier is unaffected. Thus we have two different frequency signals according to binary symbol. Let there be a frequency shift by Ω . Then we can write the following equations

$$\begin{aligned} s(t) &= A_c \cos 2\pi f_1 t && \text{for symbol '1'} \\ &= A_c \cos 2\pi f_2 t && \text{for symbol '0'} \end{aligned}$$

DESIGN

Let $V_c = 5$ volts peak-to-peak, $V_m = 10$ volts peak-to-peak, $f_m = 500$ Hz, $f_c = 50$ kHz.

Assume $h_{fe} = 30$, $V_{BEsat} = 0.7$ volts, $V_{CEsat} = 0.3$ volts, $I_c = 1$ mA, $I_c = I_e$.

$$V_{c\text{ peak}} = V_{CEsat} + I_e R_e$$

$$2.5 = 0.3 + (1\text{ m})R_e, \quad \Rightarrow R_e = 2.2\text{ k}\Omega$$

$$V_{m\text{ peak}} = R_b I_b + V_{BEsat} + I_e R_e$$

$$5 = R_b I_b + 0.7 + 2.2, \quad \text{where } I_b = I_c / h_{fe}$$

then $R_{bmax} = 63\text{ k}\Omega$, Choose $R_b = 22\text{ k}\Omega$

Envelope Detector:

$$1/f_m > R_d C_d > 1/f_c, \quad \text{hence } 2\text{ms} > R_d C_d > 20\mu\text{s}$$

$$\text{Let } R_d C_d = 50/f_c = 1\text{ ms}$$

Assume $C_d = 0.01\text{ }\mu\text{F}$, then $R_d = 100\text{ k}\Omega$

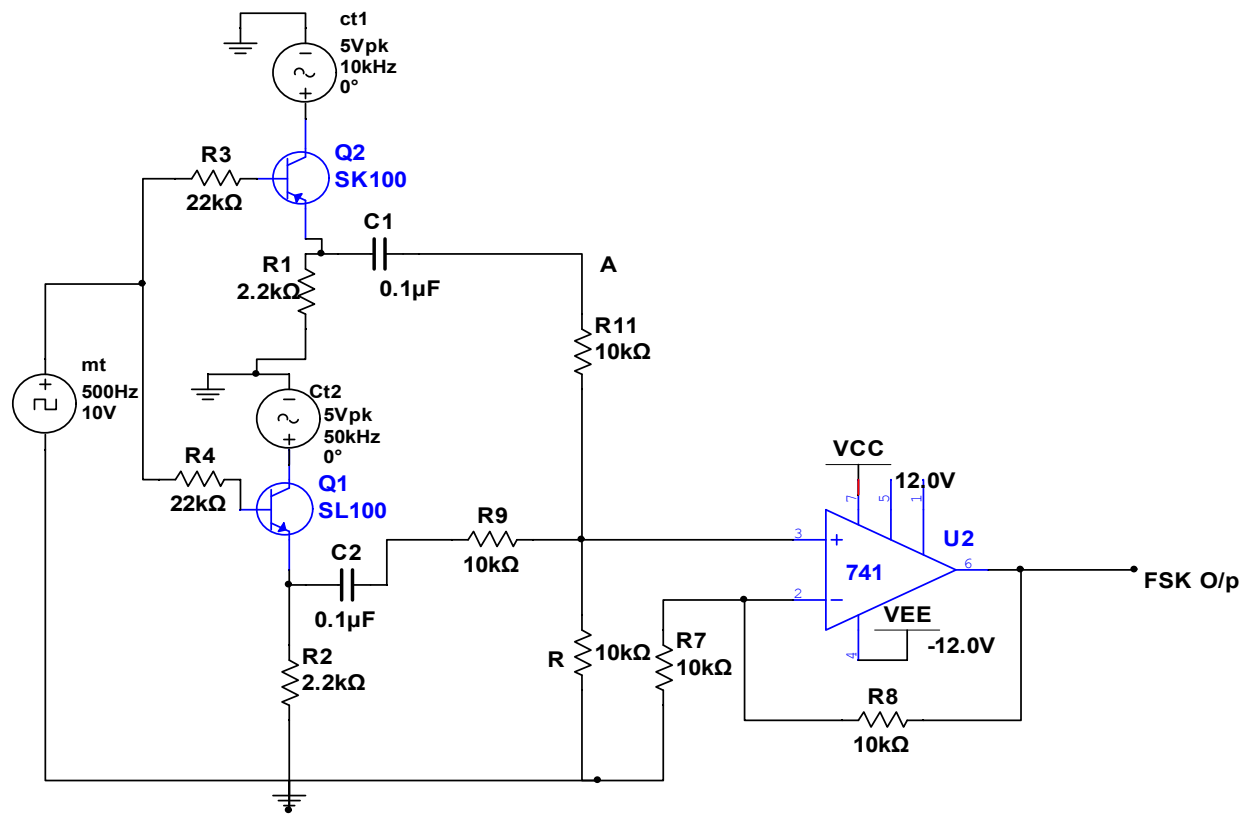
PROCEDURE:

1. Connections are made as shown in the figure.
2. Apply a square wave modulating signal
3. Apply sine wave carrier signal of given amplitude and frequency
4. Observe FSK waveform at point 'A'
5. Demodulate the FSK signal using coherent detection (Adder + Envelop detector)
{The error in the demodulated wave form can be minimized by adjusting the V_{ref} using 10K pot}

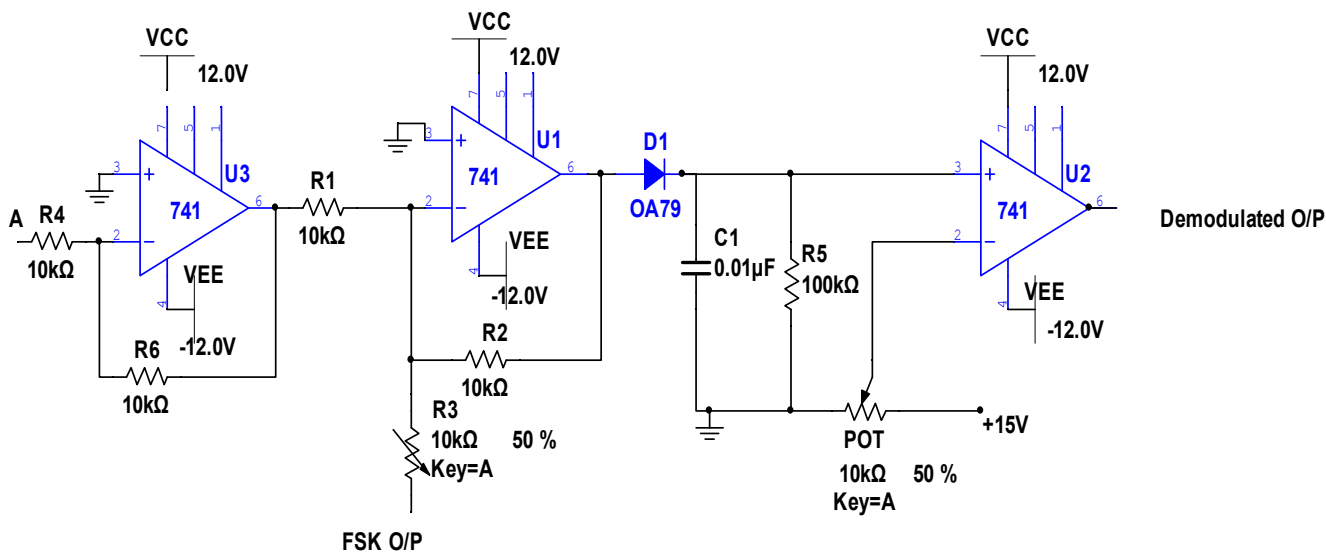
TABULAR COLUMN:

V_c (Volts)	f_c (Hz)	V_m (Volts)	f_m (Hz)	Error in detection (ms)

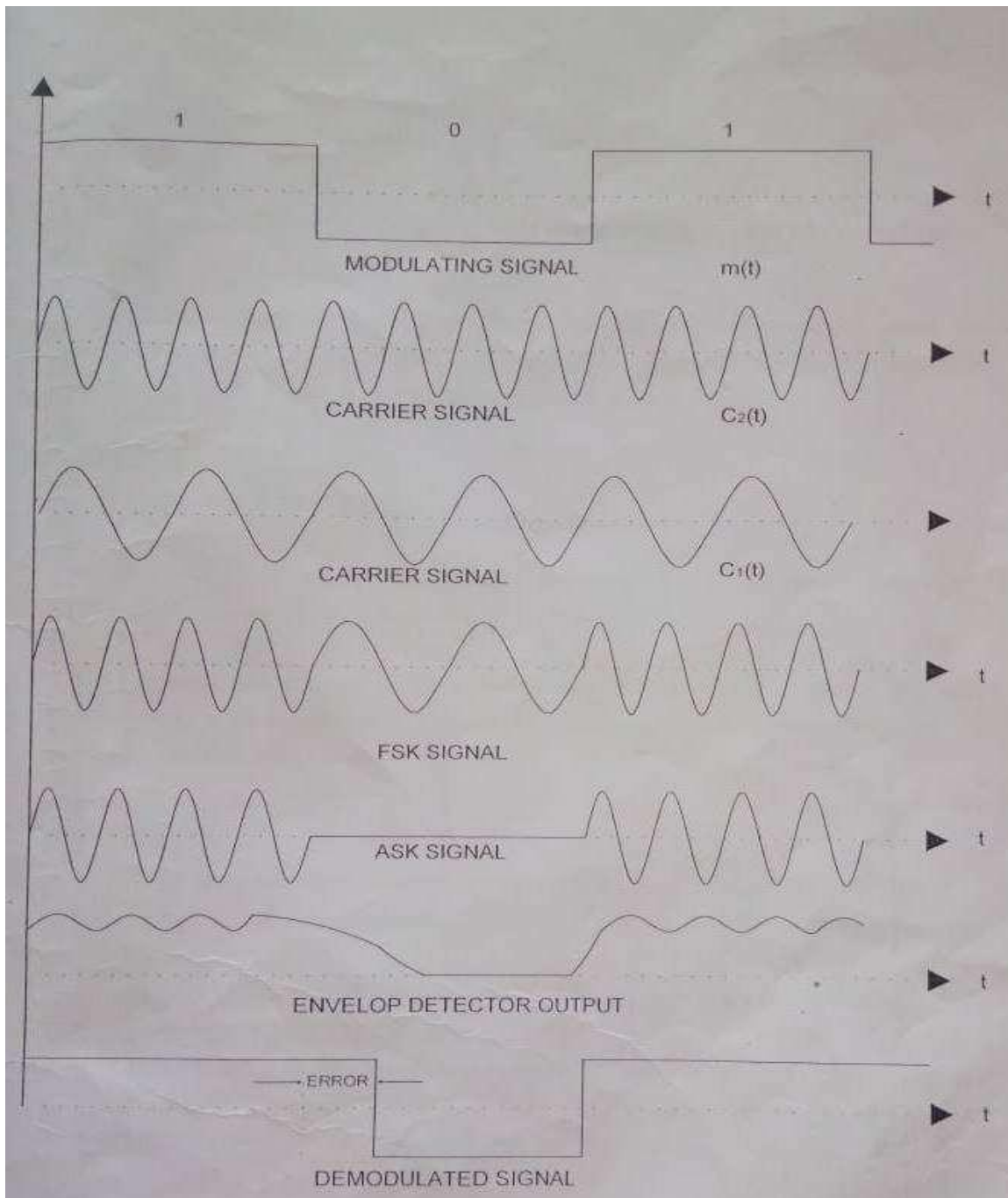
CIRCUIT DIAGRAM OF FSK MODULATOR



FSK Demodulation Circuit



WAVEFORMS OF FSK MODULATION AND DEMODULATION



RESULT:

EXPERIMENT: 2

PHASE SHIFT KEYING (PSK) GENERATION AND DETECTION

AIM: To conduct an experiment to study phase shift keying modulation and demodulation

COMPONENTS REQUIRED:

Sl. No.	Apparatus	Range	Quantity
1	IC 1458		2
2	Transistor	SL 100 SK 100	1 1
3	Diode	OA 79	1
4	Resistors	As Per Design	16
5	Potentiometer	10K Ω	1
6	Capacitor	As Per Design	3

THEORY:

In Phase shift keying the phase of the carrier is modulated by binary signal. The binary signal is used to switch the phase between 0° and 180° . Hence, it is known as phase reversal keying.(PSK) the modulated carrier is given by :

$$\begin{aligned} \text{Binary 1 : } s(t) &= A_c \cos(2\pi f_c t) \quad \text{Binary 0 : } s(t) \\ &= A_c \cos(2\pi f_c t + \pi) \end{aligned}$$

The binary PSK waveform can be described by $s(t) = D(t) A_c \cos \omega_c t$ Where, $D(t)$ is a random binary waveform with period T_b and levels + 1 and -1. The only difference between the ASK and PSK waveform is that in the ASK signaling the carrier is switched ON and OFF; whereas in the PSK signal the carrier is switched between levels + A_c and - A_c

DESIGN

Let $V_c = 5$ volts peak-to-peak, $V_m = 10$ volts peak-to-peak, $f_m = 500$ Hz, $f_c = 50$ kHz.

Assume $h_{fe} = 30$, $V_{BEsat} = 0.7$ volts, $V_{CEsat} = 0.3$ volts, $I_c = 1$ mA, $I_c = I_e$.

$$V_{c\text{ peak}} = V_{CEsat} + I_e R_e$$

$$2.5 = 0.3 + (1\text{ m})R_e, \quad \Rightarrow R_e = 2.2\text{ k}\Omega$$

$$V_{m\text{ peak}} = R_b I_b + V_{BEsat} + I_e R_e$$

$$5 = R_b I_b + 0.7 + 2.2, \quad \text{where } I_b = I_c / h_{fe}$$

Then $R_{bmax} = 63\text{ k}\Omega$, Choose $R_b = 22\text{ k}\Omega$

Envelope Detector:

$$1/f_m > R_d C_d > 1/f_c, \quad \text{hence } 2\text{ms} > R_d C_d > 20\mu\text{s}$$

$$\text{Let } R_d C_d = 50/f_c = 1\text{ ms}$$

Assume $C_d = 0.01\text{ }\mu\text{F}$, then $R_d = 100\text{ k}\Omega$

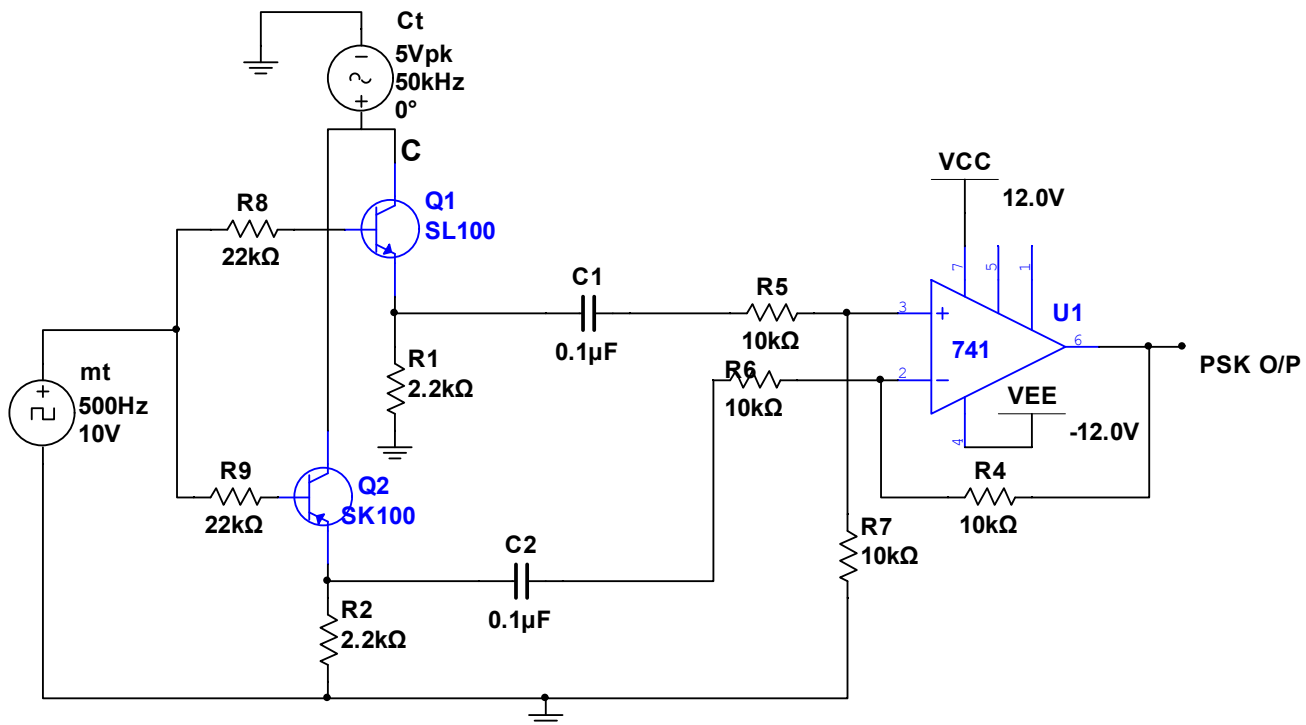
PROCEDURE:

1. Connections are made as shown in the circuit diagram.
2. Apply a square wave modulating signal of 500 Hz and 10 volts peak to peak amplitude
3. Apply sine wave carrier signal of 50 KHz of 5 volts peak to peak amplitude
4. Observe PSK waveform at point 'PSK O/P'
5. Demodulate the PSK signal using coherent detection (Adder + Envelop detector)
{The error in the demodulated wave form can be minimized by adjusting the V_{ref} using 10K pot}

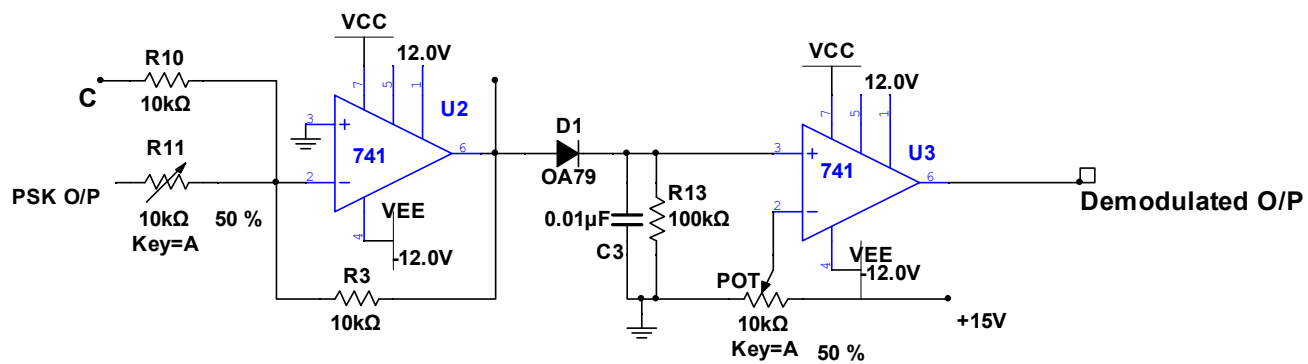
TABULAR COLUMN:

V_c (Volts)	f_c (Hz)	V_m (Volts)	F_m (Hz)	Error in detection (ms)

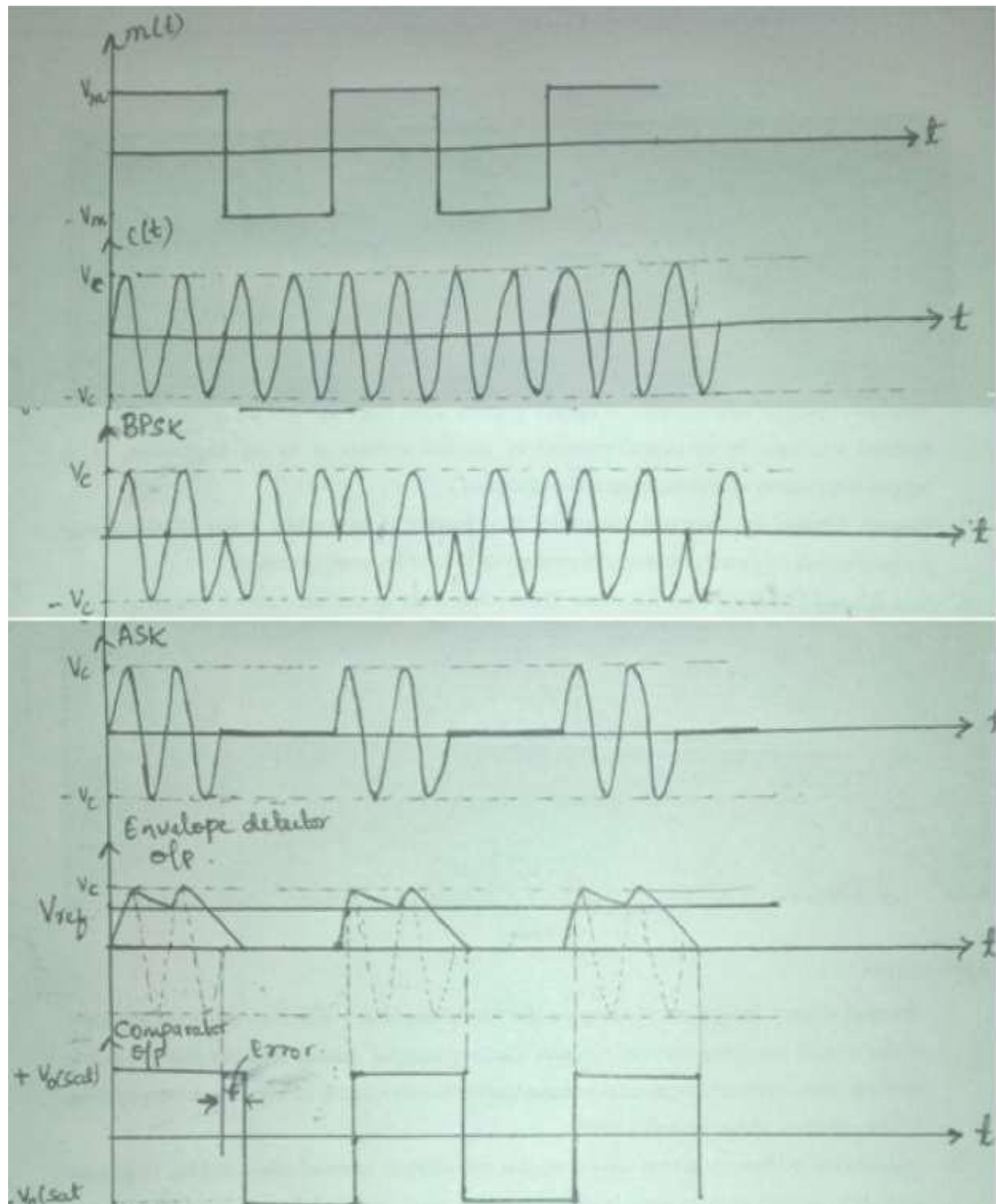
CIRCUIT DIAGRAM OF PSK MODULATION



CIRCUIT DIAGRAM OF PSK DEMODULATION



WAVEFORMS OF PSK MODULATION AND DEMODULATION



EXPERIMENT: 3

QPSK Generation & Detection

AIM: Study of Carrier Modulation Techniques by Quadrature Phase Shift Keying method.

EQUIPMENTS:

1. Experiment or Kits ADCL-02 & ADCL-03.
2. Connecting Chords.
3. Power Supply.
4. 20MHz Dual Trace Oscilloscope.

THEORY:

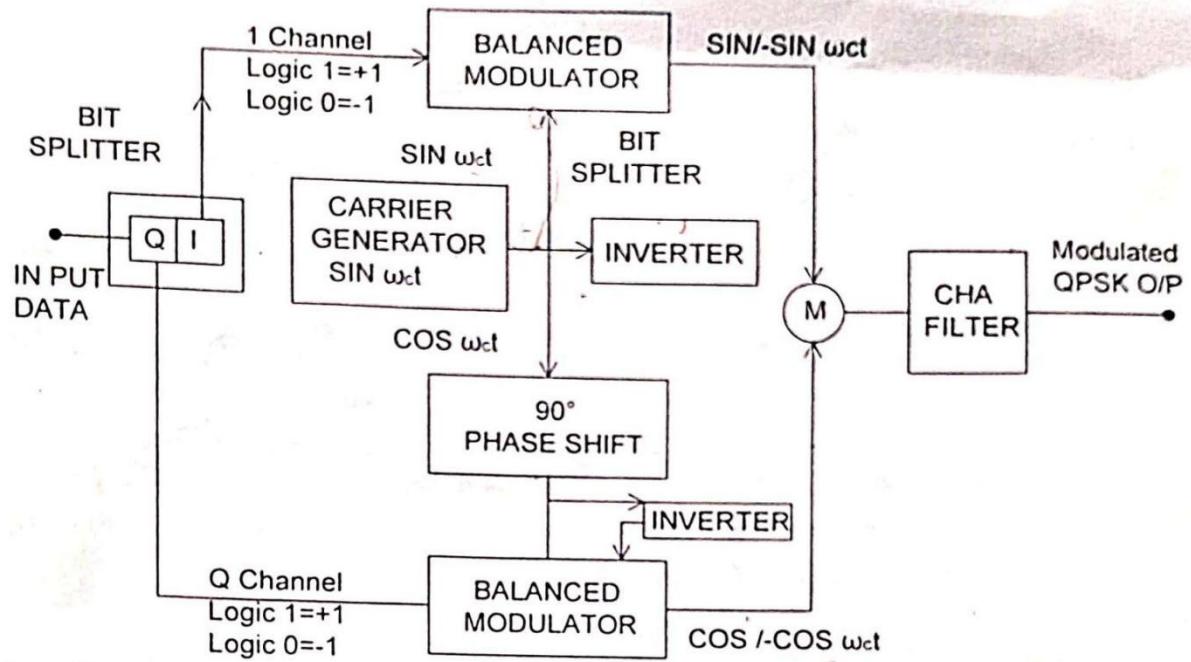
In this modulation, called Quadrature PSK (QPSK) or 4 PSK the sine carrier takes 4 phase values, separated of 90 deg. and determined by the combinations of bit pair (Dibit) of the binary data signal. The data are coded into Dibit by a circuit generating:

A data signal I (in phase) consisting in voltage levels corresponding to the value of the first bit of the considered pair, for duration equal to 2 bit intervals.

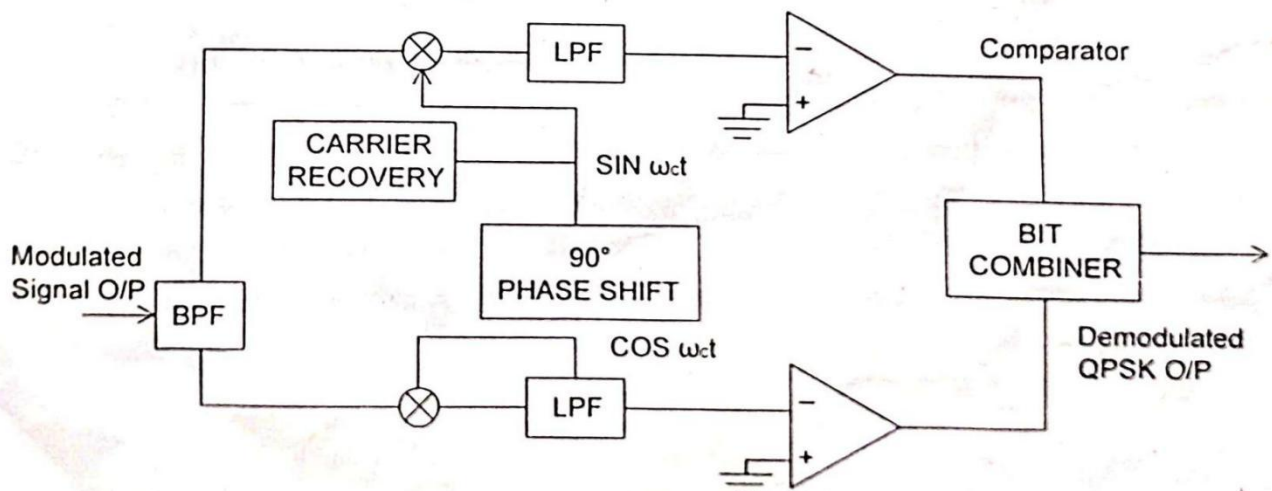
A data signal Q (in quadrature) consisting in voltage levels corresponding to the value of the second bit of the pair, for duration equal to 2 bit intervals.

The block diagram of the modulator used on the module is shown four 500KHz sine carriers, shifted between them of 90 deg, are applied to modulator. The data (signal I & Q) reach the modulator from the Dibit generator. The instantaneous value of I and Q data bit generates a symbol. Since I and Q can take either 0 or 1 value, maximum 4 possible symbols can be generated (00, 01, 10, and 11). According to the symbol generated one of the four-sine carrier will be selected. The relation between the symbol generated and sine carrier is shown in table.

QPSK TRANSMITTER



QPSK RECEIVER



DIBIT	PHASE SHIFT
00	180 deg
01	90 deg
10	270 deg
11	0 deg

A receiver for the QPSK signal is shown fig. synchronous detection is required and hence it is necessary to locally regenerate the carriers. The scheme for carrier regeneration is similar to that employed in BPSK. In that earlier case we squared the incoming the signal, extracted the waveform at twice the carrier frequency by filtering, and recovered the carrier by frequency dividing by two. In the present case, it is required that the incoming signal be raised to the fourth power after which filtering recovers a wave forms at four times the carrier.

The incoming signal also applied to the sampler followed by an adder and envelope detectors. Two adders add the sampled QPSK signal, sampled by the clock having different phases. At the output of adder the signals consisting the envelope corresponds to the I & Q bit. Envelope detector then filters the high Frequency components and recovers I & Q bit. These recovered I & Q bit. These I & Q exactly same phase & frequency compared to transmitter I & Q bit. These I & Q bits then applied to data decoder logic to recover the original NRZ-L data pattern.

NOTE: KEEP ALL THE SWITCH FAULTS IN OFF POSITION

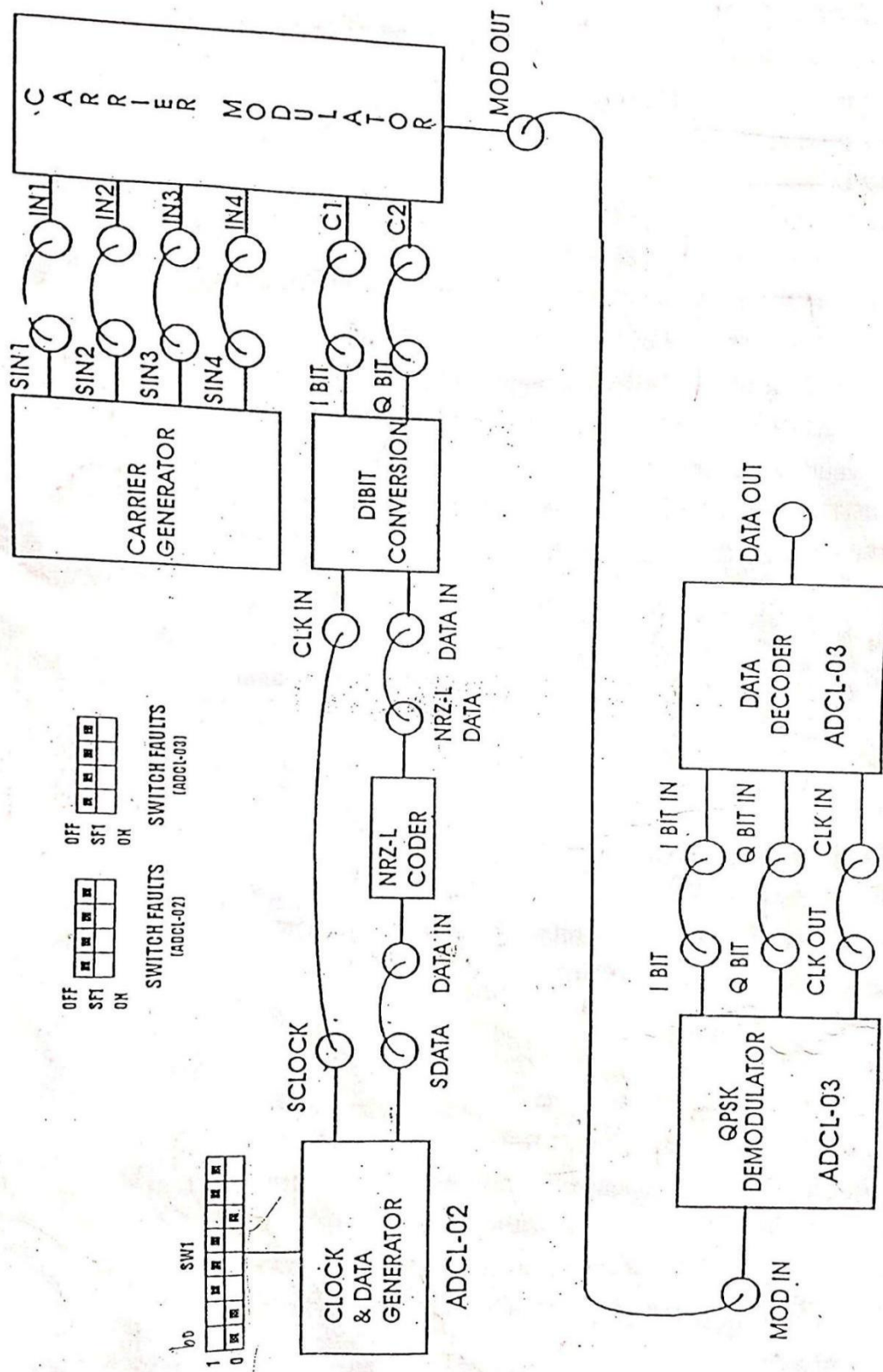


FIG. BLOCK DIAGRAM FOR QUADRATURE PHASE SHIFT KEYING MODULATION TECHNIQUE

PROCEDURE:

1. Refer to the block diagram and carry out the following connections and switch settings.
2. Connect power supply in proper polarity to the kit ADCL-02 and ADCL-03 and switch it on.
3. Select Data pattern of simulated data using switch SW1.
4. Connect SDATA generated to DATA IN of NRZ-L CODER.
5. Connect the NRZ-L DATA to DATA IN of the DIBIT CONVERSION.
6. Connect SCLOCK to CLK IN the DIBIT CONVERSION.
7. Connect the dibit data I & Q bit to control input C1 and C2 of CARRIER MODULATOR respectively. NOTE : Adjust I & Q bit as shown. By operating RST Switch on ADCL-02 before connecting it to C1 & C2.
8. Connect carrier component to input of CARRIER MODULATOR as follows:
 - a. SIN 1 to IN 1
 - b. SIN 2 to IN 2
 - c. SIN 3 to IN 3
 - d. SIN 4 to IN 4
9. Connect QPSK modulated signal MOD OUT on ADCL-02 to the MOD IN of the QPSK DEMODULATOR on ADCL-03. NOTE : Adjust Recovered I & Q bit on ADCL-03 as per ADCL-02 by RST Switch on ADCL-03.
10. Connect I BIT, Q BIT & CLK OUT outputs of QPSK Demodulator to I BIT IN, Q BIT IN & CLK IN posts of Data Decoder respectively
11. Observe various wave forms.

NOTE: If THERE IS MISMATCH IN INPUT & recovered data, THEN ADJUST THAT Data by RST Switch on ADCL-03.

OBSERVATION:

Observe the following wave forms on oscilloscope and plot it on the paper.

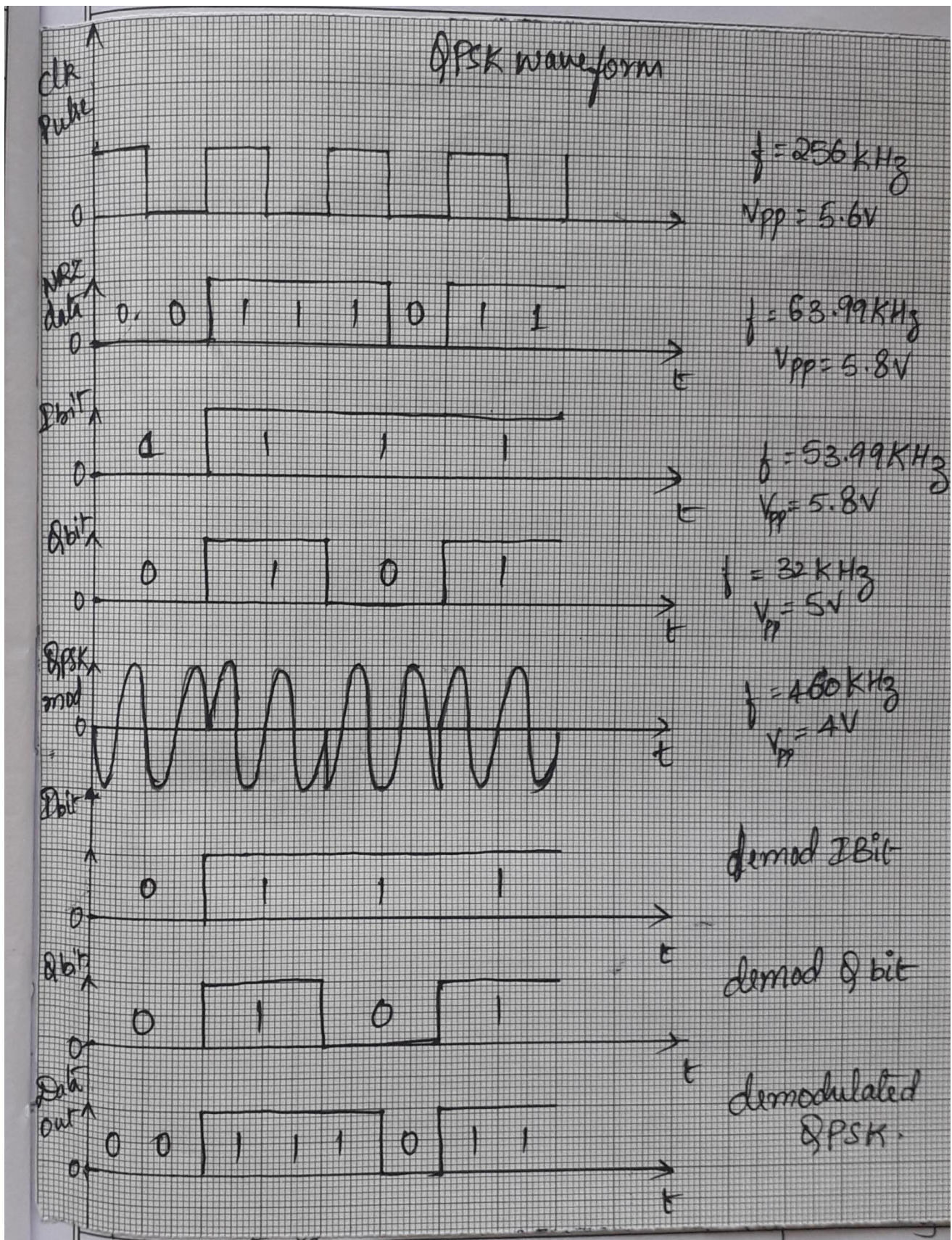
ON KIT ADCL-02

1. Input NRZ-L Data at DATA INPUT.
2. Carrier frequency SIN 1 to SIN 4
3. Dibit pair generated data I bit & Q bit at DIBIT CONVERSION.
4. QPSK modulated signal at MOD OUT.

ON KIT ADCL-03

1. Output of first squarer at SQUARER 1.
2. Output of second squarer at SQUARER 2.
3. Four sampling clocks at the output of SAMPLING CLOCK GENERATOR.
4. Two adder outputs at the output of ADDER.
5. Recovered data bits (I & Q bits) at the output of ENVELOP DETECTORS.
6. Recovered NRZ-L data from I & Q bits at the output of DATA DECODER.

Expected Wave Forms



Experiment: 4

DPSK Generation & Detection

AIM: Study of Carrier Modulation Techniques by Differential Phase Shift Keying (DPSK) method

EQUIPMENTS:

1. Experimentor Kit DPSK.
2. Connecting Chords.
3. Power Supply.
4. 20MHz Dual Trace Oscilloscope.

THEORY:

In BPSK communication system, the demodulation is made by comparing the instant phase of the BPSK signal to an absolute reference phase locally generated in the receiver.

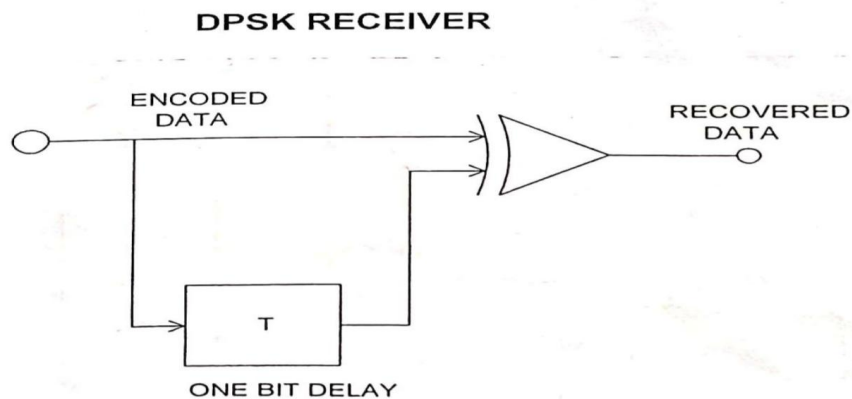
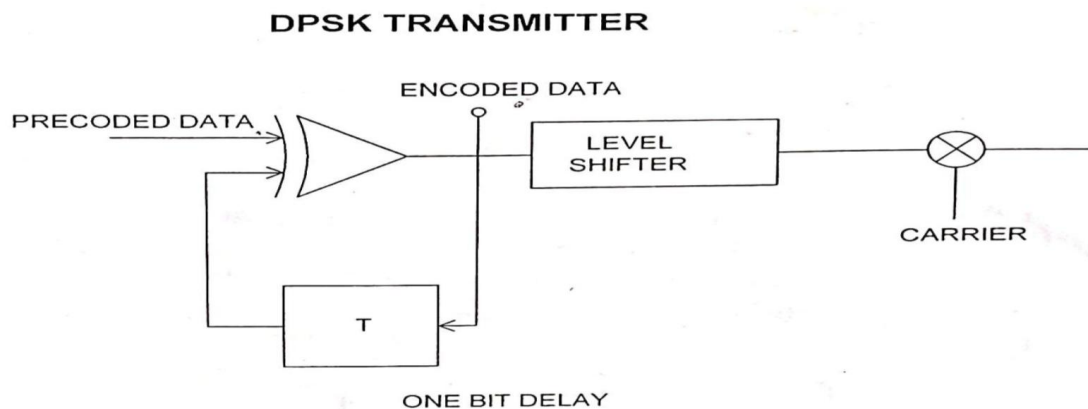
The modulation is called in this case BPSK absolute. The greatest difficulty of these systems lies in the need to keep the phase of the regenerated carrier always constant. This problem is solved with the PSK differential modulation, as the information is not contained in the absolute phase of the modulated carrier but in the phase difference between two next modulation intervals.

The Block Diagram of DPSK modulation and demodulation system. Coding is obtained by comparing the output of an EX-OR, delayed of a bit interval, with the current data bits. As total result of operation, the DPSK signal across the output of the modulator contains 180 deg. Phase variation at each data bit “1”. The demodulation is made by a normal BPSK demodulator, followed by a decision device supplying a bit “1” each time there is a variation of the logic level across its input.

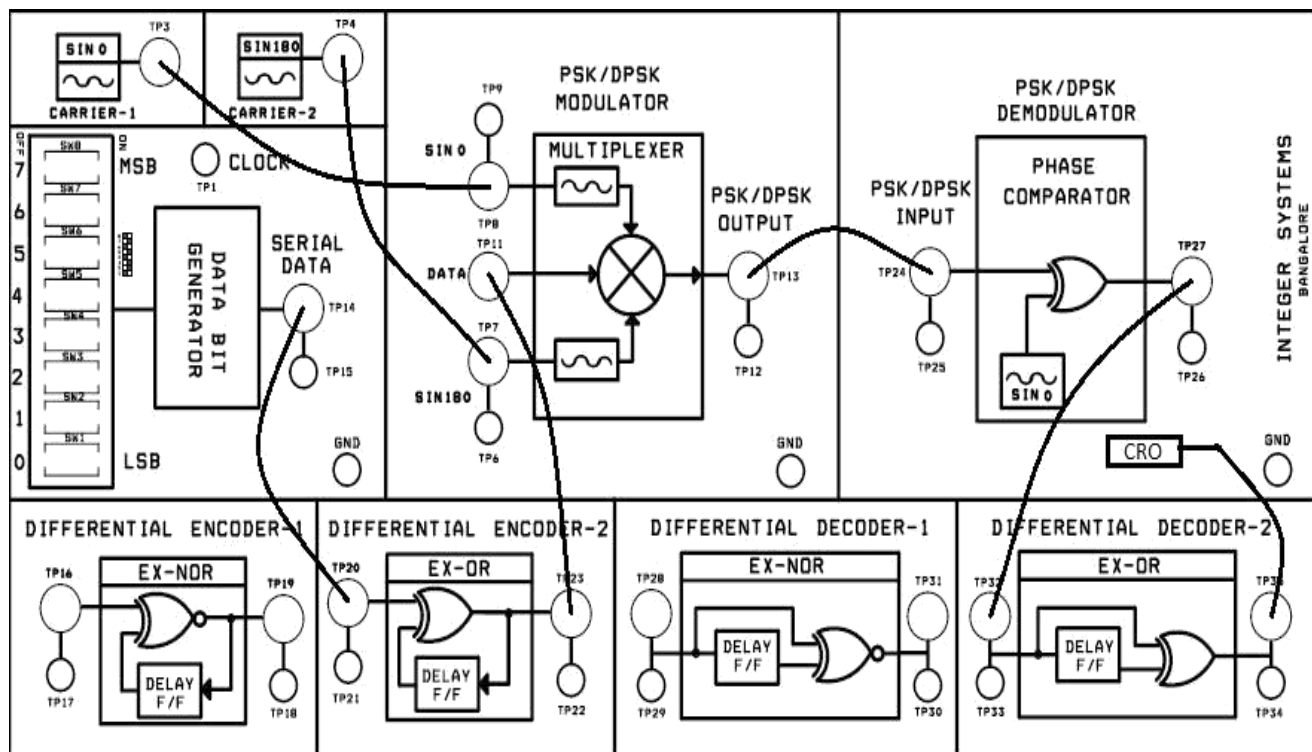
The DPSK system explained above has a clear advantage over the BPSK system in that the former avoids the need for complicated circuitry used to generate a local carrier at the receiver. To see the relative disadvantage of DPSK in comparison with PSK, consider

that during some bit interval the received signal is so contaminated by noise that in a PSK system an error would be made in the determination of whether the transmitted bit was a 1 or 0. In DPSK a bit determination is made on the basis of the signal received in two successive bit intervals. Hence noise in one bit interval may cause errors to two-bit determination. The error rate in DPSK is therefore greater than in PSK, and, as a matter of fact, there is a tendency for bit errors to occur in pairs. Single errors are still possible.

NOTE: KEEP ALL THE SWITCH FAULTS IN OFF POSITION



Connection Diagram of D PSK Modulator and demodulator with EX-OR Differential encoder and decoder



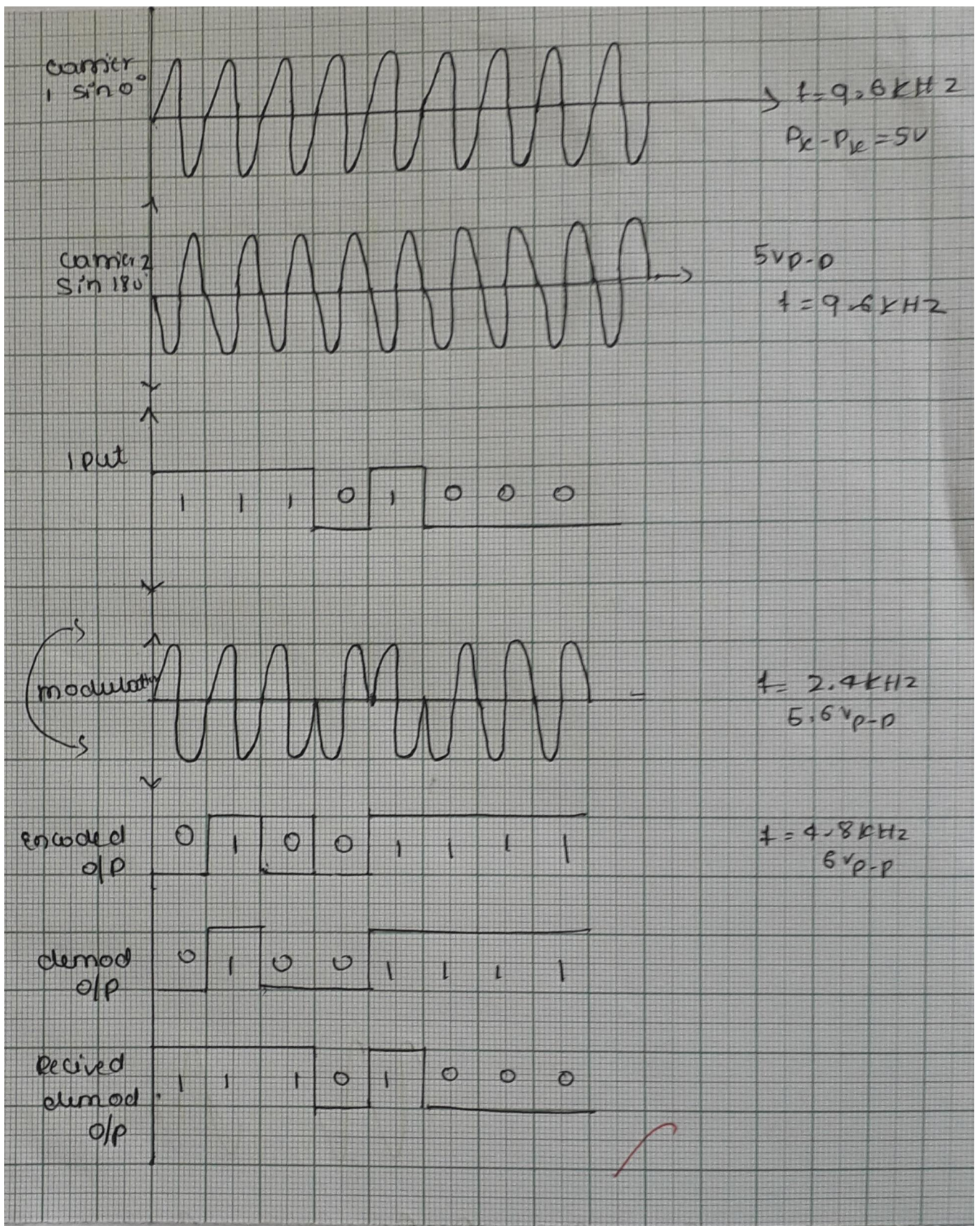
Procedure:

1. Switch on MICRO LAB INSTRUMENTS PSK/DPSK Trainer kit.
2. Connect the carrier output of carrier SIN0 (TP3) and SIN180 (T P4) degree to the PSK/DPSK modulator Block TP9 and TP7 respectively as show in the connection diagram.
3. Connect the Data Bit output (TP14) to the Differential Encoder -2(EX-OR logic) input at test point TP20.
4. Observe the differential data with EX-OR logic at the output of the Differential Encoder -2 at test point TP23.
5. Connect the Differential Encoder -2 output at Test point TP23 to Data input (TP11) of PSK/D PSK Modulator block.
6. Observe the DPSK output at the test point TP13.
7. Connect the DPSK signal to PSK/DPSK Demodulator test point (TP23)
8. Observe DPSK demodulated data at the test point TP27.

Result:

The operation of DPSK is observed and the output wave forms are verified.

Expected Waveforms:



Experiment: 5

HUFFMAN ENCODER AND DECODER

AIM: To write a MATLAB program to encode binary data using Huffman code and decode it.

SOFTWARE USED: MATLAB R2007b

THEORY:

Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. Huffman coding assigns variable length code words to fixed length input characters based on their frequencies. More frequent characters are assigned shorter code words and less frequent characters are assigned longer code words. The output from Huffman's algorithm can be displayed as a variable-length code table for encoding a source symbol. Method of Huffman can be efficiently implemented, finding a code in time linear to the number of input weights if these weights are sorted.

PROGRAM:

```
clc;
clear all;
p=input('Enter the probabilities:');
s=sort(p,'descend');
n=length(p);
symbols=[1:n];
[dict,avglen]=huffman dict(symbols,s);
disp(dict);
temp=dict;
t=dict(:,2);
for i=1:length(temp)
    temp{i,2}=num2str(temp{i,2});
end
disp('The huffman code dict:');
disp(temp);
fprintf('Enter the symbols between 1 to %d in[]',n);
sym=input(': ');
encod=huffman encode(sym,dict);
disp('The encoded output:');
disp(encod);
bits=input('Enter the bit stream in[]');
decod=huffman deco(bits,dict);
disp('The symbols are:');
```



```

disp(decod);
H=0;
for(k=1:n)
    H=H+(p(k)*log2(1/p(k)));
end
fprintf(1,'Entropy is %f bits',H);
N=H/avglen;
fprintf("\n Efficiency is:%f,N);

```

OUT PUT

Command Window

enter the probabilities[0.4 0.3 0.2 0.1]

```

[1]      [          1]
[2]      [1x2 double]
[3]      [1x3 double]
[4]      [1x3 double]

```

the huffman code dict:

```

[1]      '1'
[2]      '0 1'
[3]      '0 0 0'
[4]      '0 0 1'

```

enter the symbols 1 to 4 in []:[2 1]

sym =

```

      2      1

```

the encoded output:

```

      0      1      1

```

Experiment: 6

HAMMING ENCODER AND DECODER

AIM: To write a MATLAB program to encode binary data using [7,4] Hamming code and decode it.

SOFTWARE USED: MATLAB R2007b

THEORY:

Hamming code is a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors. In this coding method, the source encodes the message by inserting redundant bits within the message. These redundant bits are extra bits that are generated and inserted at specific positions in the message itself to enable error detection and correction. When the destination receives this message, it performs recalculations to detect errors and find the bit position that has error.

Encoding a message by Hamming Code

The procedure used by the sender to encode the message encompasses the following steps –

- Step 1 – Calculation of the number of redundant bits.
- Step 2 – Positioning the redundant bits.
- Step 3 – Calculating the values of each redundant bit.

Decoding a message in Hamming Code

Once the receiver gets an incoming message, it performs recalculations to detect errors and correct them. The steps for recalculation are –

- Step 1 – Calculation of the number of redundant bits.
- Step 2 – Positioning the redundant bits.
- Step 3 – Parity checking.
- Step 4 – Error detection and correction

PROGRAM:

```
clc;
clear all;
n=7;
k=4;
A=[1 1 1;1 1 0;1 0 1;0 1 1];
G=[eye(k) A]
H=[A' eye(n-k)]
%Encoder
msg=input('Enter the 4bit input Message:');%[1 0 1 1];
code=mod(msg*G,2)
%Channel error
code(7)=~code(7);
recd=code
%decoder
syndrome=mod(recd*H',2)
%disp("H");
disp(H');
%find position of error in code word
find=0;
for ii=1:n
    if ~find
        errvect=zeros(1,n);
errvect(ii)=1;
        search=mod(errvect*H',2);
        if search== syndrome
            find=1;
            index=ii;
        end
    end
end
disp(['Position of error in code word=',num2str(index)]);
Corrected code=recd;
Corrected code(index)=mod(recd(index)+1,2);
msg_decoded=corrected code
msg_decoded=msg_decoded(1:4);
```

OUT PUT

Command Window

G =

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

H =

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

enter the 4bit message[1 0 1 1]

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

the position of error in codeworld is 7

correctedcode =

1	0	1	1	0	0	0
---	---	---	---	---	---	---

correctedcode =

1	0	1	1	0	0	1
---	---	---	---	---	---	---

msg_decoded =

1	0	1	1	0	0	1
---	---	---	---	---	---	---

msg_decoded =

1	0	1	1
---	---	---	---

Experiment: 7

CONVOLUTIONAL ENCODER AND DECODER

AIM: To write a MATLAB program to encode binary data using Convolution code and decode it.

SOFTWARE USED: MATLAB R2007b

THEORY:

Convolutional coding is a widely used coding method which is not based on blocks of bits but rather the output code bits are determined by logic operations on the present bit in a stream and a small number of previous bits. In the encoder, data bits are input to a shift register of length K , called the constraint length. As each bit enters at the left of the register, the previous bits are shifted to the right while the oldest bit in the register is removed. Two or more binary summing operations, let's say r , create code bits which are output during one data flow period. Therefore, the code bit rate is $1/r$ times the data rate and the encoder is called a rate $1/r$ convolutional encoder of constraint length K .

PROGRAM:

```
clear all;
g=[1 1 1;1 0 1];
[n,k]=size(g);
m=k-1;
state=zeros(1,m);
%input x=[0 1 0 1 1 1 0 0 1 0 1 0 0 0 1];
Input x=[1 0 1 1 1];
[trash,h]=size(input x)
Output y=[];
for x=1:h
input=input x(1,x);
for i=1:n
output(i)=g(i,1)*input;
for j=2:k
z=g(i,j)*state(j-1);
output(i)=xor(output(i),z);
end
End

state=[input,state(1:m-1)];
Output y=[output y,output]
end
```

OUT PUT

Command Window

```
enter the input message bits=[1 0 0 1 1]
```

```
convol encoded data
```

```
Columns 1 through 9
```

```
1    1    1    0    1    1    1    1    0
```

```
Column 10
```

```
1
```

```
decoded data
```

```
1    0    0    1    1
```

```
>> |
```

Experiment: 8

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

- I) Without error.
- II) With error.

Theory: -

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

Example:

To compute an n-bit binary CRC, line the bits are representing the input in a row, and position the (n+1)-bit pattern representing the CRC's divisor (called a "polynomial") underneath the left-hand end of the row. Start with the message to be encoded: 11010011101100

This is first padded with zeroes corresponding to the bit length n of the CRC. Here is the first calculation for computing a 3-bit CRC:

11010011101100 000 : INPUT RIGHT PADDED WITH ZERO BITS

1011 : DIVISOR (4 BITS)

----- 01100011101100 000 □ RESULT

If the input bit above the leftmost divisor bit is 0, do nothing. If the input bit above the leftmost divisor bit is 1, the divisor is XORed in to the input. The divisor is then shifted one bit to the right, and the process is repeated until the divisor reaches the right-hand end of the input row. Here is the entire calculation:

```

11010011101100 000 <--- input right padded by 3 bits
1011                <--- divisor
01100011101100 000 <--- result
 1011                <--- divisor ...
00111011101100 000
 1011
00010111101100 000
 1011
00000001101100 000
 1011
00000000110100 000
 1011
00000000011000 000
 1011
00000000001110 000
 1011
00000000000101 000
 101 1
-----
00000000000000 100 <---remainder (3 bits)

```

Since the leftmost divisor bit zeroed every input bit it touched, when this process ends the only bits in the input row that can be nonzero are the n bits at the right- hand end of the row. These n bits are the remainder of the division step, and will also be the value of the CRC function (unless the chosen CRC specification calls for some post processing).

The validity of a received message can easily be verified by performing the above calculation again, this time with the check value added instead of zeroes. The remainder should equal zero if there are no detectable errors.


```

11010011101100 100 <--- input with check value
1011             <--- divisor
01100011101100 100 <--- result
 1011            <--- divisor ...
00111011101100 100

.....

000000000001110 100
      1011
00000000000101 100
      101 1
-----
0 <--- remainder

```

Program: -//crc can detect all single bit error, double bit , odd bits of error and burst error

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

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

```
char t[30],cs[30],g[40]; int
```

```
a,i,j,N;
```

```
void xor()
```

```
{
    for(j=1;j<N; j++)
        cs[j] = ((cs[j]==g[j])?'0':'1');
}
```

```
void crc()
```

```
{
for(i=0;i<N;i++)
cs[i]=t[i];
do
{
    if(cs[0]=='1')
        xor();
    for(j=0;j<N-1;j++)

        cs[j]=cs[j+1];
        cs[j]=t[i++];
} while(i<=a+N-1);
}
```

```
int main()
```

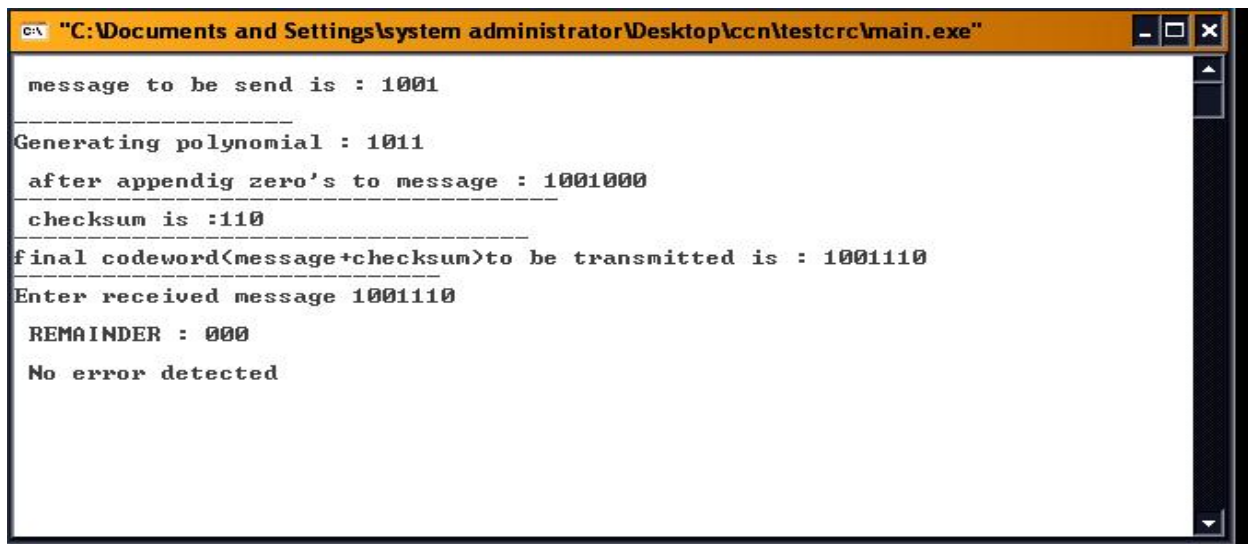
```

{
    printf("\n message to be send is :");
    scanf("%s",&t);
    printf("\n-----");
    printf("\n Generating polynomial : ");
    scanf("%s",&g);
    a=strlen(t);
    N=strlen(g);
    for(i=a;i<(a+N-1);i++)
        t[i]='0';
    printf("\n   after   appending   zero's   to
message : %s",t);printf("\n_____");
    crc();
    printf("\n checksum is :%s",cs);
    for(i=a;i<(a+N-1);i++)
        t[i]=cs[i-a];
    printf("\n_____");
    printf("\n  final  code  word(message+checksum)to  be  transmitted
is : %s",t);printf("\n_____");
    printf("\n Enter received message");
    scanf("%s",&t);
    crc();
    printf("\n REMAINDER : %s",cs);
    for(i=0;(i<N-1)&&(cs[i]!='1');i++);
    if(i<N-1)

```

```
printf("\n\n error detected\n\n");
else
printf("\n\n No error detected\n\n");
return 0;
}
```

Output without Error:



```
"C:\Documents and Settings\system administrator\Desktop\ccn\testcrc\main.exe"
message to be send is : 1001
-----
Generating polynomial : 1011
after appendig zero's to message : 1001000
checksum is :110
-----
final codeword(message+checksum)to be transmitted is : 1001110
Enter received message 1001110
REMAINDER : 000
No error detected
```

Output with error



```
"C:\Documents and Settings\system administrator\Desktop\ccn\testcrc\main.exe"
message to be send is :1001
-----
Generating polynomial : 1011
after appendig zero's to message : 1001000
checksum is :110
-----
final codeword(message+checksum)to be transmitted is : 1001110
Enter received message 1011110
REMAINDER : 110
error detected
```

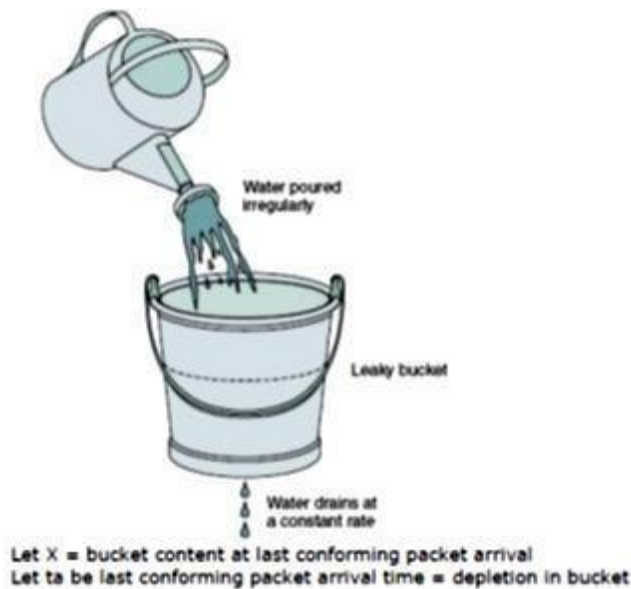
Experiment: 9

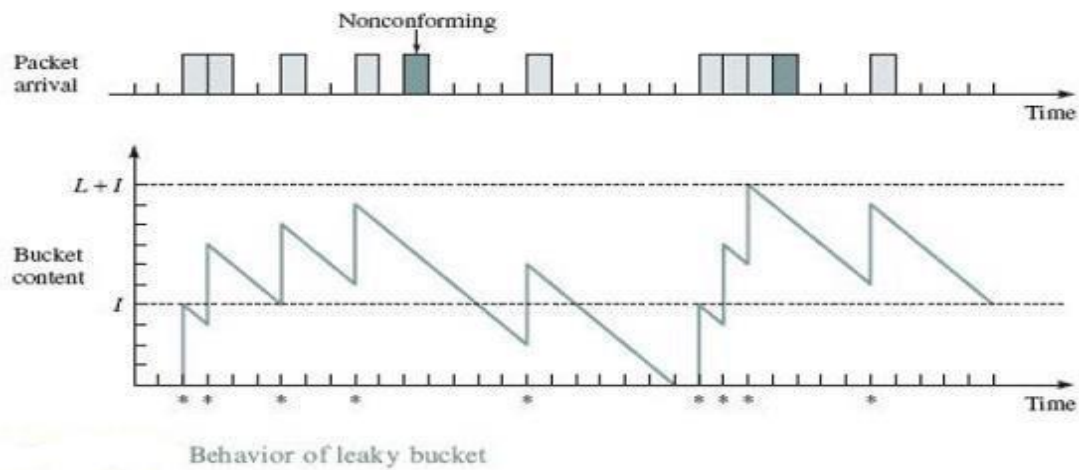
Aim: - Write a program for congestion control using leaky bucket algorithm in C language and execute the same and display the result.

Theory:

1. Network monitors traffic flows continuously to ensure they meet their traffic contract.
2. The process of monitoring and enforcing the traffic flow is called policing.
3. When a packet violates the contract, network can discard or tag the packet giving it lower priority
4. If congestion occurs, tagged packets are discarded first
5. Leaky Bucket Algorithm is the most commonly used policing mechanism
 - ❖ Bucket has specified leak rate for average contracted rate
 - ❖ Bucket has specified depth to accommodate variations in arrival rate
 - ❖ Arriving packet is conforming if it does not result in overflow

Leaky Bucket algorithm can be used to police arrival rate of a packet stream





Leaky Bucket Algorithm:

Algorithm:

- Step 1: Define the bucket size and number of packets sent
- Step 2: Enter the output rate (i.e.) the rate at which the data flows from bucket.
- Step 3: Generate random number of data in a packet & random wait time for the next packet arrival.
- Step 4: If the data content in packet is less than bucket size add the content to the bucket else reject the packet.
- Step 5: Call the bucket output subprogram which outputs the data in bucket at constant rate within the next arrival time of the next packet.
- Step 6: The Step 3 repeats (i.e.) next packet arrives with random size & random time. Which is added to the bucket along with old content if it is left in the bucket and the process continues till all the packets are entered into the bucket?
- Step 7: Check whether the content of the bucket is zero. If not output the content in a constant data rate. Till the content is less the output rate.
- Step 8: Print the remaining data in bucket which is less than output rate.

//Leacky bucket program:

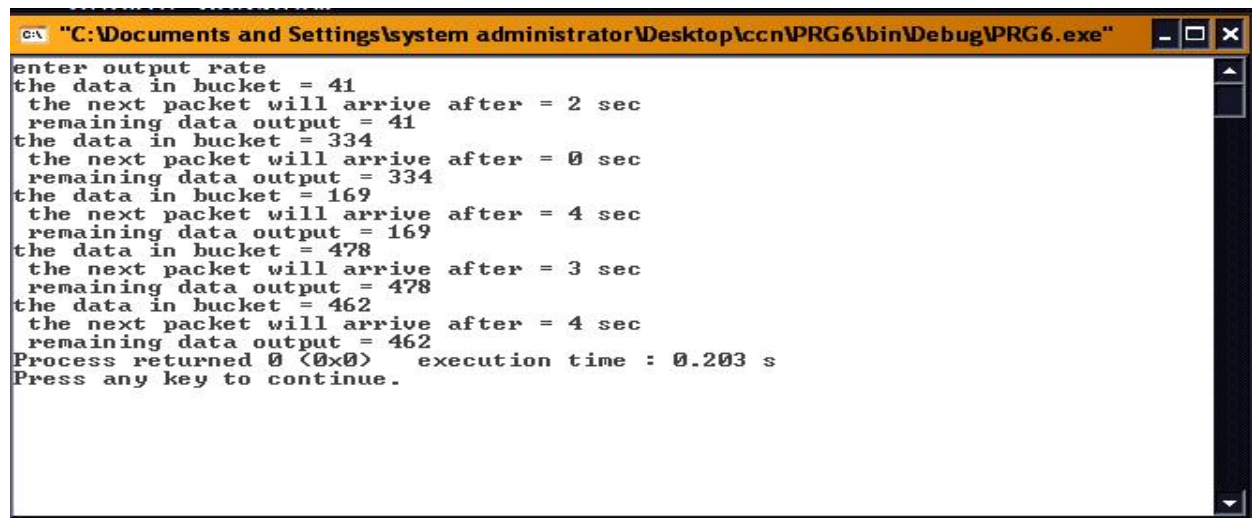
```
#include<stdio.h>      #include
<time.h> #include <windows.h>
#define bucket size 1000
#define n 5
// user defined function to output the contents of bucket at a constant rate void bucketoutput (int
*bucket,int op)
{
if(*bucket > 0 && *bucket > op) // if the no. of bytes in the bucket >output rate
{
*bucket= *bucket-op; //no. of bytes in bucket – output rate
printf("\n%d-out puted remaining is %d",op,*bucket);
}
else if(*bucket > 0) // if the bucket is not empty
{
printf("\n remaining data output = %d",*bucket);
*bucket=0;
}
}
int main()
{
int op,new pack,old pack=0,wt,i,j,bucket=0;
// op – ouput rate, wt- waiting time, bucket- no.of bytes in the bucket at anypoint of
time
printf("enter output rate"); // input the output rate scanf("%d",&op);
for(i=1;i<=n;i++)

{
New pack=rand()%500;
//new packet with random size is generated
printf("\n\n new packet size = %d",new pack); new
pack=old pack+new pack;
wt=rand()%5; // random waiting time is generated if(new pack<bucket size)if(new
pack<bucket size)
bucket=new pack;else
{
printf("\n%d = the new packet and old pack is greater thanbucket
size reject",new pack);
bucket=old pack;
}

printf("\n the data in bucket = %d",bucket);
printf("\n the next packet will arrive after = %d sec",wt);
// calling output rate function with wait time for(j=0;j<wt;j++)
```

```
{  
Bucket output(&bucket,op); Sleep(1);  
}  
Old pack=bucket;  
}  
while(bucket>0) bucket output(&bucket,op);  
return 0;  
}
```

Output



```
C:\Documents and Settings\system administrator\Desktop\ccn\PRG6\bin\Debug\PRG6.exe  
enter output rate  
the data in bucket = 41  
the next packet will arrive after = 2 sec  
remaining data output = 41  
the data in bucket = 334  
the next packet will arrive after = 0 sec  
remaining data output = 334  
the data in bucket = 169  
the next packet will arrive after = 4 sec  
remaining data output = 169  
the data in bucket = 478  
the next packet will arrive after = 3 sec  
remaining data output = 478  
the data in bucket = 462  
the next packet will arrive after = 4 sec  
remaining data output = 462  
Process returned 0 (0x0) execution time : 0.203 s  
Press any key to continue.
```

Experiment: 10

Aim:- Write a program for Distance Vector Algorithm to find suitable path for transmission.

Theory: -

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

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

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

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

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

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

Distance Vector Algorithm

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
 - ❖ It receives a distance vector from a neighbor containing different information than before.
 - ❖ It discovers that a link to a neighbor has gone down.
4. The process repeats on minimizing the cost to each destination.

Algorithm

Step 1: Initialize the no of nodes in the network and the array $g[i][j]$ to enter the routes between the nodes

Step 2: Enter the number of nodes and the cost matrix $c[i][j]$. It is the cost of going from vertex $[i]$ to vertex $[j]$. If there is no edge between vertex i & j then $c[i][j]$ is infinity which is indicated by a largest value „999“

Step 3: Update the link between the nodes from the neighboring node information. Step 4: Print the link between all the nodes.

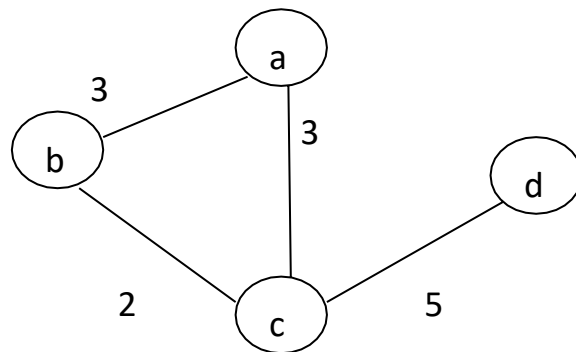
Program: -

```
/*Distance Vector Routing in this program is implemented using Bellman Ford Algorithm:-*/#include
<stdio.h>
#include <stdlib.h>
int d[10][10],via[10][10];
int main()
{
    int i,j,k,n,g[10][10];
    printf("\n Enter the no of nodes:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter the record for route %c\n",i+97);
        for(j=0;j<n;j++)
        {
            printf("(%c:%c):.",i+97,j+97);
            scanf("%d",&g[i][j]);
            if(g[i][j]!=999)
                d[i][j]=1;
        }
    }
    for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            via[i][j]=i;

    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)

            if(d[i][j]==1)
                for(k=0;k<n;k++)
                    if(g[i][j]+g[j][k]<g[i][k])
                    {
                        g[i][k]=g[i][j]+g[j][k];
                        via[i][k]=j;
                    }
    }
    for(i=0;i<n;i++)
    {
        printf("cost of route %c:\n",i+97);
        for(j=0;j<n;j++)
            printf("(%c:%d via %c\n",j+97,g[i][j],via[i][j]+97);
    }
    return 0;
}
```

Output:



Here a, b, c and d which are inside the circle are nodes of the graph, and the number between them are the distances of the graph. Now using Distance vector algorithm we can find the path between initial node and the remaining vertices.

```
C:\Documents and Settings\system administrator\Desktop\ccn\distanceprog\main.exe
Enter the no of nodes: 4
enter the record for route a
(a:a)::0 3 23 999
(a:b)::(a:c)::(a:d)::enter the record for route b
(b:a)::3 0 2 999
(b:b)::(b:c)::(b:d)::enter the record for route c
(c:a)::3 2 0 5
(c:b)::(c:c)::(c:d)::enter the record for route d
(d:a)::999 999 5 0
(d:b)::(d:c)::(d:d)::cost of route a:
a:0 via a
b:3 via a
c:5 via b
d:10 via c
cost of route b:
a:3 via b
b:0 via b
c:2 via b
d:7 via c
cost of route c:
a:3 via c
b:2 via c
c:0 via c
d:5 via c
cost of route d:
a:8 via c
b:7 via c
c:5 via d
d:0 via d
Process returned 0 (0x0)   execution time : 31.219 s
```

Experiment: 11

Aim: -Write a program for flow control using Sliding Window Protocol.

Theory:

In this protocol (and the next), the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers.

The range which is the concern of the sender is called the send sliding window; the range that is the concern of the receiver is called the receive sliding window. We discuss both here. The send window is an imaginary box covering the sequence numbers of the data frames which can be in transit.

In each window position, some of these sequence numbers define the frames that have been sent; others define those that can be sent. The maximum size of the window is $2^m - 1$ for reasons that we discuss later. In this chapter, we let the size be fixed and set to the maximum value, but we will see in future chapters that some protocols may have a variable window size.

Figure.1(a): Send window before sliding

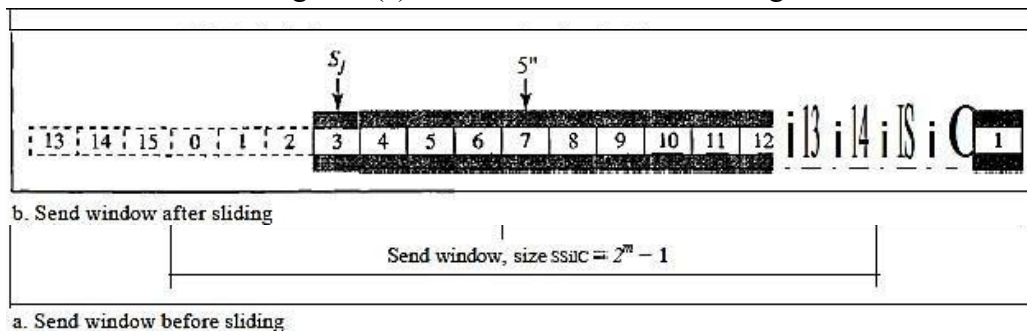


Figure.1 (b): Send window after sliding

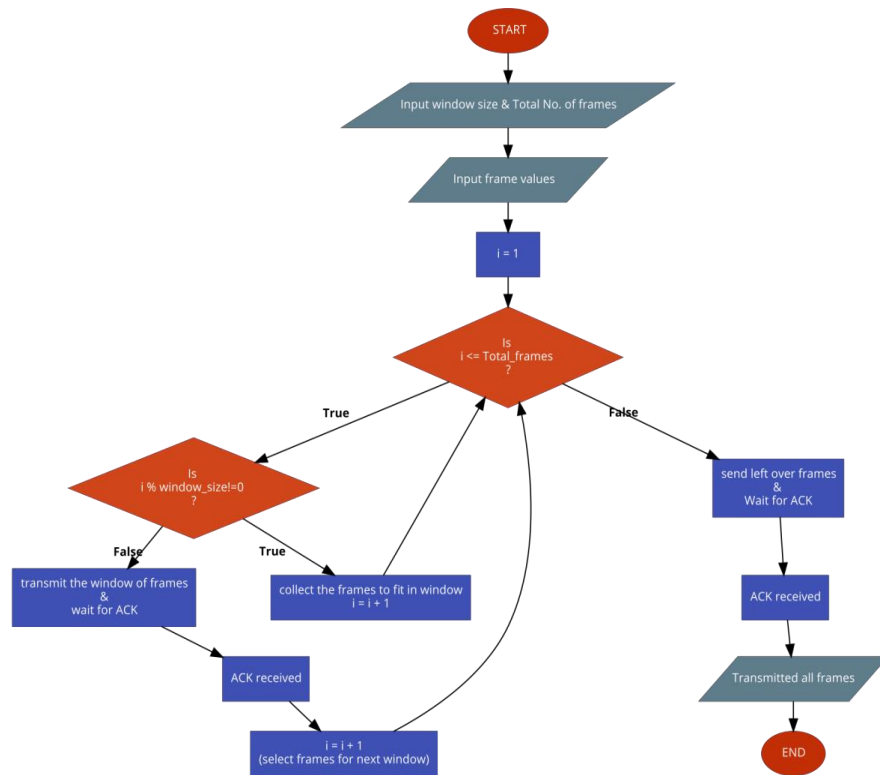
Figure.1 shows a sliding window of size 15 ($m = 4$). The window at any time divides the possible sequence numbers into four regions.

The first region, from the far left to the left wall of the window, defines the sequence numbers belonging to frames that are already acknowledged. The sender does not worry about these frames and keeps no copies of them.

The second region, colored in Figure.1 (a), defines the range of sequence numbers belonging to the frames that are sent and have an unknown status. The sender needs to wait to find out if these frames have been received or were lost. We call these outstanding frames.

The third range, white in the figure, defines the range of sequence numbers for frames that can be sent; however, the corresponding data packets have not yet been received from the network layer. Finally, the fourth region defines sequence numbers that cannot be used until the window slides.

Flow chart:



In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agree on some window size. If window size= w then after sending w frames sender waits for the acknowledgement (ack) of the first frame. As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frame are properly received, for e.g.:- if ACK of frame 3 is received it understands that frame 1 and frame 2 are received properly.

Algorithm:

Step 1: Define RTT, Time out & total no of frames to be sent.

Step 2: Assign ACK=yes (i.e.) assume frame 1 is sent and the sender is waiting for the acknowledgement from the receiver.

Step 3: Generate a random wait time if wait time < time out. Allow propagation delay time (i.e.) RTT/2 duration for the acknowledgement from receiver to reach the sender. And then print the statement ack is sent for the respective frame and assign ACK = yes.

Step 4: Repeat the above steps till all the frames are sent and ACK are received for the respective frames.

Program:

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

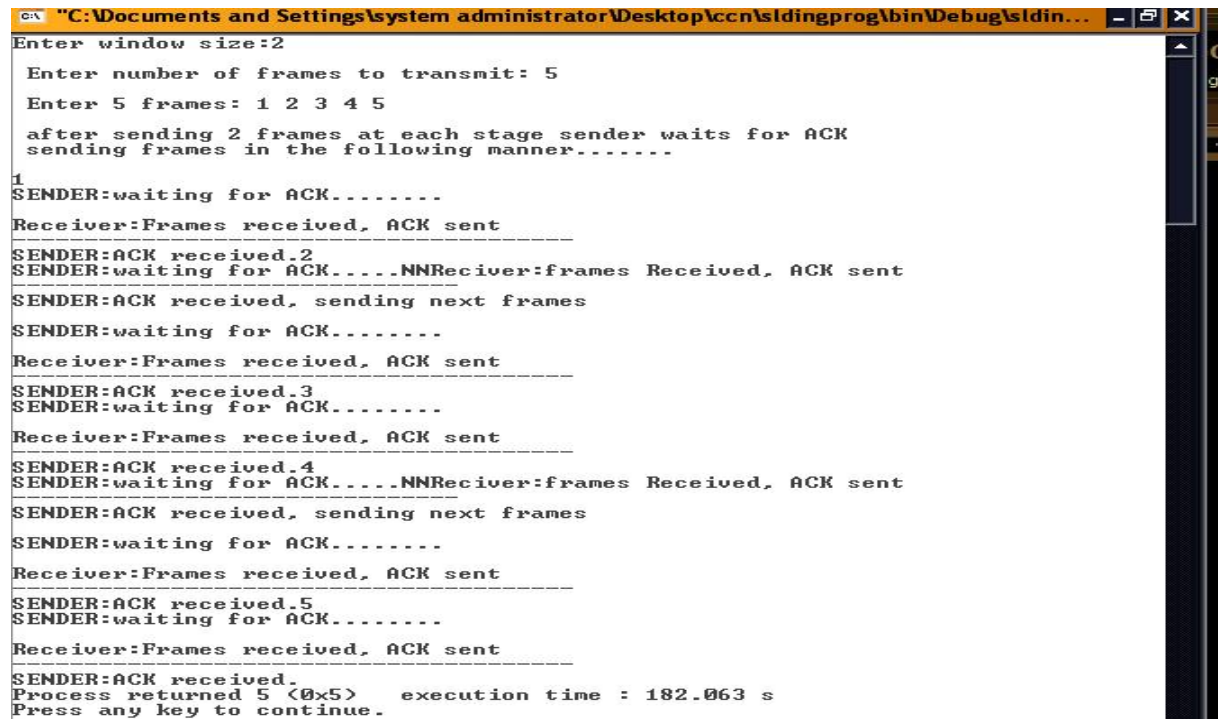
int main()
{
int W,f,frames[50]; printf("enter
window size:");scanf("%d",&W);
printf("\n enter number of frames to transmit");
scanf("%d",&f);
printf("\n enter %d frames:",f);
for(i=1;i<=f;i++)
scanf("%d",&frames[i]);
printf("\n with sliding window protocol the frames will be sent in the
following manager (assuming no corruption of frames)\n\n");
printf("after sending %d frames at each stage sender waits for acknowledgement sent by thereceiver
\n\n",W);
for(i=1;i<=f;i++)
{
if(i%w==0)
{
printf("%d\n",frames[i]);
printf("acknowledgement of above frames sent is received by sender\n\n");
}
}
```

```

else printf("%d\n",frames[i]);
} if(f%W!=0)
printf("\n acknowledgement of above frames sent is received by sender\n");return 0;
}

```

Output:



```

C:\Documents and Settings\system administrator\Desktop\ccn\slidingprog\bin\Debug\slidin...
Enter window size:2
Enter number of frames to transmit: 5
Enter 5 frames: 1 2 3 4 5
after sending 2 frames at each stage sender waits for ACK
sending frames in the following manner.....
1
SENDER:waiting for ACK.....
Receiver:Frames received, ACK sent
=====
SENDER:ACK received.2
SENDER:waiting for ACK.....NNReceiver:frames Received, ACK sent
SENDER:ACK received, sending next frames
SENDER:waiting for ACK.....
Receiver:Frames received, ACK sent
=====
SENDER:ACK received.3
SENDER:waiting for ACK.....
Receiver:Frames received, ACK sent
=====
SENDER:ACK received.4
SENDER:waiting for ACK.....NNReceiver:frames Received, ACK sent
SENDER:ACK received, sending next frames
SENDER:waiting for ACK.....
Receiver:Frames received, ACK sent
=====
SENDER:ACK received.5
SENDER:waiting for ACK.....
Receiver:Frames received, ACK sent
=====
SENDER:ACK received.
Process returned 5 (0x5)   execution time : 182.063 s
Press any key to continue.

```

Experiment: 12

Aim : Configure a simple network (Ring/Mesh) topology using simulation software

Ns2 installation steps

- Step 1: This video shows you the demonstration of ns2 in Linux Mint 14 (64Bit) OS
- Step 2: issue this command in the terminal "sudo apt-get install build-essential autoconf auto make libxmu-dev
 - " (without Quotes)
 - Step 3: if you have Ubuntu issue this command first before u try step 2 : "sudo apt-get update"
 - Step 4: Once installation succeeded, start the process of installing ns2
 - Step 5: copy the ns-allinone-2.3x.tar.gz in the home folder, in my case it is /home/pradeep
 - Step 6: go to home folder and execute this command "tar zxvf ns-allinone-2.35.tar.gz"
 - step 7: cd ns-allinone-2.35 and ./install (after these commands, wait for the successful installation), it may take 5 min to 15 minutes based on the speed of your computer.
 - Step 8: there may be errors shown, in case if errors, the errors should be corrected and reissue the command ./install
 - step 9: There is an error on ns-2.35/link state/ls.h file, so the error has to be corrected as pointed out in the video, In line number 137, change **erase to this->erase**
 - Step 10: Once installation is succeeded, you will get some path information
 - Step 11: set the paths in the .profile file which is under the home directory (**/home/hdl1(system name)/.profile**)
 - as specified in the video, copy the **PATH** url and **LD_LIBRARY_PATH** URL and paste in the .profile file.
 - Step 12: Once the path information is set, run the command to test the path "source
 - /home/pradeep/.profile) and see there should not be any errors
 - Step 16: try ns and nam individually and see whether they are working when you type ns and type enter a % symbol indicates the installation is successful when you type nam and a network animation window pop up.

Nsg2.1

Installation steps

- Step1 : sudo apt-get update"
- Step2 : sudo apt-get install default.jdk

What is NSG2? :

NS2 Scenarios Generator 2(NSG2) is a JAVA based ns2 scenarios generator. Since NSG2 is written by JAVA language, you can run NSG on any platform. NSG2 is capable of generating both wired and wireless TCL scripts for ns2. Some major functions of NSG2 are listed below:

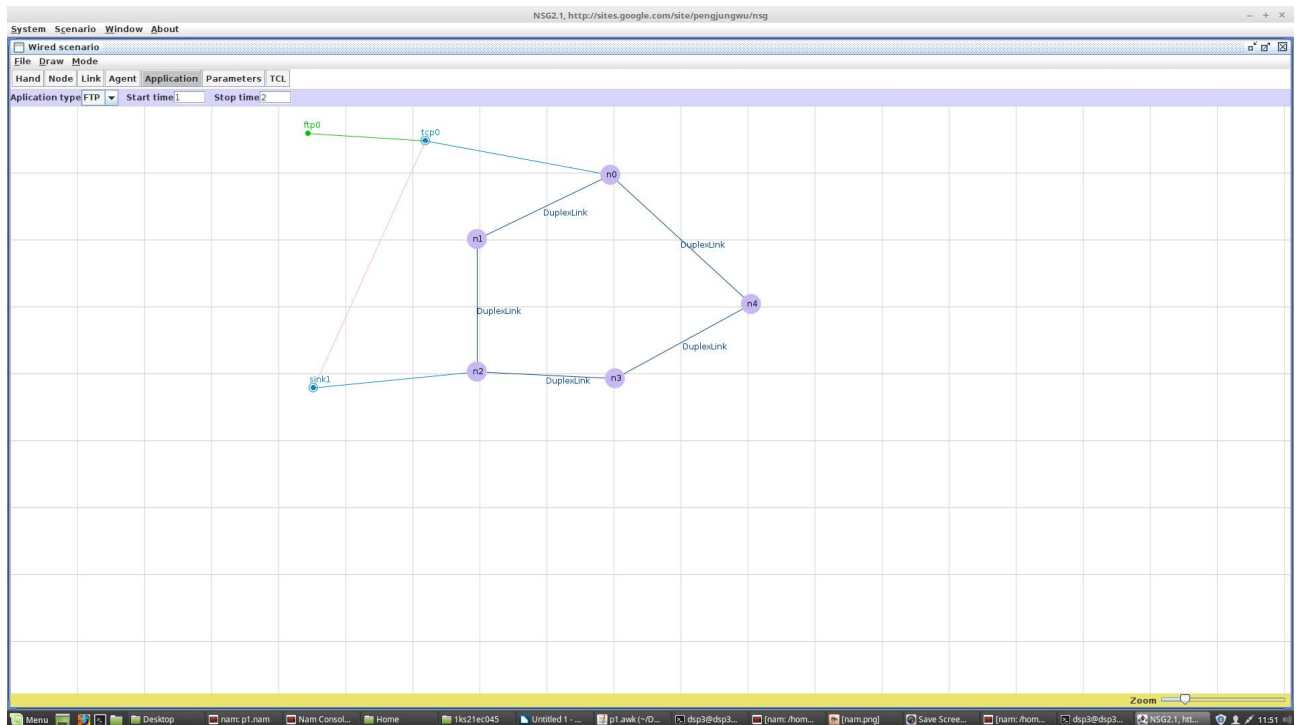
1. Creating wired and wireless nodes
2. Creating connection between nodes
3. Creating links (Duplex-Link and Simplex-Link)
4. Creating agents (TCP and UDP)
5. Creating applications (CBR and FTP)
6. Node movement

PS. A previous version of NSG that can only generate wireless scenarios can also be found [here](#).

General procedure to run the TCL script and verify the output

1. Right click in the folder where NSG is installed to open the terminal.
2. Type „ls,, and check for list of copied files and type the command “ java –jar NSG2.1.jar”
3. Go the scenario and select “ New wired /wireless scenario”
4. Click on the nodes and place the nodes for the required scenario and connect them with link between nodes by choosing any one of them below mentioned according to the requirement.
 - Duplex link -> TCP
 - Simplex link -> UDP
5. Connect the agent to the nodes suitably [TCP & UDP]
6. Fix the application for the agent as shown
 - TCP -> application -> FTP
 - UDP -> application -> CBR
7. Connect the agent between source node and destination node.
8. Click on parameters in the toolbar & rename trace file and nam file and save it.
9. Click on TCL and generate TCL script for the scenario
10. Save the TCL file with same name as trace and nam file with .tcl extension
11. Go the terminal and type command : “ns filename.tcl”
12. Run trace file and check the movement of packet from source to destination.
13. To measure the performance of the experiment we create AWK file.
14. To create AWK file go to menu->accessories-> gedit.
15. Type the awk code and save in the same location.
16. To execute awk file go to terminal and type command awk –f filename.awk filename.tr

12. Configure a simple network (Ring/Mesh) topology using simulation software



TCL Script

```
# This script is created by NSG2 beta1
```

```
# <http://wushoupong.googlepages.com/nsg>
```

```
#=====
```

```
#   Simulation parameters setup
```

```
#=====
```

```
set val(stop) 10.0           ;# time of simulation end
```

```
#=====
```

```
#   Initialization
```

```
#=====
```

```
#Create a ns simulator
```

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

```
#Open the NS trace file
```

```
set tracefile [open p1.tr w]
```

```
$ns trace-all $tracefile
```

```
#Open the NAM trace file
```

```
set namfile [open p1.nam w]
```

```
$ns namtrace-all $namfile
```

```
#=====
```

```
#    Nodes Definition
```

```
#=====
```

```
#Create 5 nodes
```

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

```
set n1 [$ns node]
```

```
set n2 [$ns node]
```

```
set n3 [$ns node]
```

```
set n4 [$ns node]
```

```
#=====
```

```
#    Links Definition
```

```
#=====
```

```
#Createlinks between nodes
```

```
$ns duplex-link $n0 $n1 100.0Mb 10ms DropTail
```

```
$ns queue-limit $n0 $n1 50
```

```

$ns duplex-link $n2 $n1 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n1 50
$ns duplex-link $n3 $n2 100.0Mb 10ms DropTail
$ns queue-limit $n3 $n2 50
$ns duplex-link $n4 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n4 $n3 50
$ns duplex-link $n0 $n4 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n4 50
#Give node position (for NAM)
$ns duplex-link-op $n0 $n1 orient left-down
$ns duplex-link-op $n2 $n1 orient left-up
$ns duplex-link-op $n3 $n2 orient left
$ns duplex-link-op $n4 $n3 orient right-down
$ns duplex-link-op $n0 $n4 orient right-down

```

```

#=====

```

```

#   Agents Definition

```

```

#=====

```

```

#Setup a TCP connection

```

```

set tcp0 [new Agent/TCP]

```

```

$ns attach-agent $n0 $tcp0

```

```

set sink1 [new Agent/TCPSink]

```

```

$ns attach-agent $n2 $sink1

```

```

$ns connect $tcp0 $sink1

```

```

$tcp0 set packetSize_ 1500

```

```

#=====

#   Applications Definition

#=====

#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ns at 1.0 "$ftp0 start"

$ns at 2.0 "$ftp0 stop"


#=====

#   Termination

#=====

#Define a 'finish' procedure

proc finish {} {

    global ns tracefile namfile

    $ns flush-trace

    close $tracefile

    close $namfile

    exec nam p1.nam &

    exit 0

}

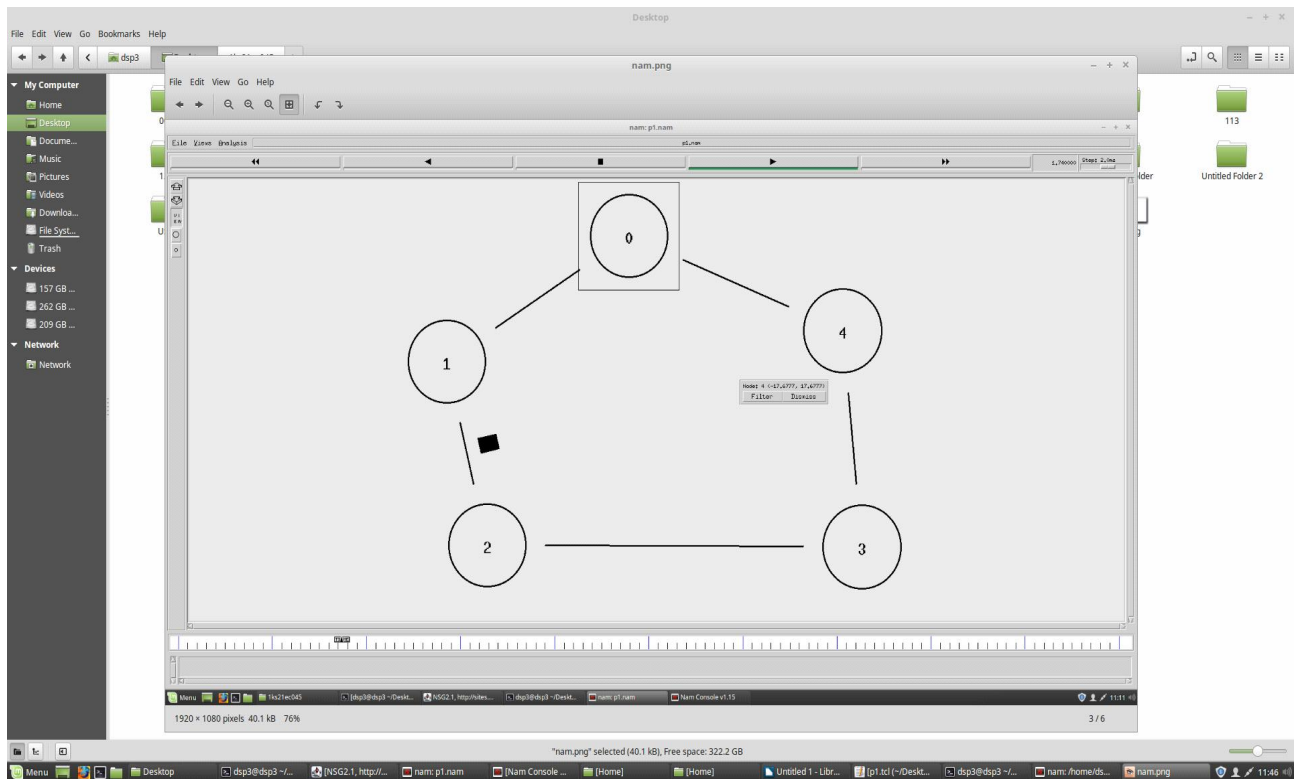
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "finish"

$ns at $val(stop) "puts \"done\" ; $ns halt"

$ns run

```



AWK Script

```
BEGIN{
a=0
}
{
if($1=="r"&&$3=="0" && $4=="3"&& $5=="tcp")
{
a++;
}
}
END{
printf("no of packects received=%d\n",a);
}
```

