**Ramaiah Institute of Technology**

**(Autonomous Institute, Affiliated to VTU)**

**Department of Electronics and Communication Engineering**

**VII Semester**

**WIRELESS AND DATA COMMUNICATION (ECL 74)**

**List of Experiments**

**Data Communication**

1. Write a program for error detection using CRC-CCITT (16 bits) using C.
2. Write a program for a HLDC frame to perform bit stuffing and de-stuffing in a single frame.
3. Write a program for a HLDC frame to perform character stuffing and de-stuffing in a single frame.
4. Write a program for encryption and decryption of text.
5. Simulate a three node point-to-point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped using NS2.
6. Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP agent between n1-n3. Apply relevant applications over TCP and UDP agents by changing the parameters and determine the number of packets sent by TCP/UDP using NS2.

**Wireless Communication**

1. Analyze the performance of Quadrature Amplitude Modulation (QAM) and M-ary Phase Shift Keying (PSK) scheme in AWGN channel, and compare the results with theoretical results.
2. Compute Bit Error Rate (BER) for different digital modulation schemes in frequency-flat and slowly varying fading channel.
3. Bit error rate analysis of digital communication receivers with Maximal Ratio Combining (MRC) receives diversity in frequency-flat and slowly varying fading channel.
4. Bit error rate analysis of digital communication receivers with Equal Gain Combining (EGC) receives diversity in frequency-flat and slowly varying fading channel.
5. Simulation of Direct Sequence Spread Spectrum (DSSS) techniques.
6. (a) Measurement of numerical aperture and attenuation loss in analog fiber optic link.

(b) Data multiplexing using fiber optic link.

**Course Outcomes:**

1. Examine the performance of the algorithms of OSI model layers (POs – 1, 2, 3, 4, 5, PSO – 3)
2. Use simulators to evaluate the network performance in different layers like NS2. (POs – 1, 2, 3, 4, 5, PSO – 3)
3. Analyze the performance of the digital modulation receivers in AWGN and fading channel. (POs – 1, 2, 3, 4, 5, PSO – 3)
4. Analyze the performance of diversity receiver in multipath fading channel. (POs – 1, 2, 4, 5, PSO – 3)
5. Examine the characteristics of analog and digital optical link. (POs – 1, 2, 4, 5, PSO – 3)

**Course Contents and Lab Schedule**

| **Lab Session No** | **Experiments** | **CO** |
| --- | --- | --- |
|  | Write a program for error detection using CRC-CCITT (16 bits) using C. | CO1 |
|  | Write a program for a HLDC frame to perform bit stuffing and de-stuffing in a single frame. | CO1 |
|  | Write a program for a HLDC frame to perform character stuffing and de-stuffing in a single frame. | CO1 |
|  | Write a program for encryption and decryption of text. | CO1 |
|  | Simulate a three node point-to-point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped using NS2. | CO2 |
|  | Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP agent between n1-n3. Apply relevant applications over TCP and UDP agents by changing the parameters and determine the number of packets sent by TCP/UDP using NS2. | CO2 |
|  | **Test 1** | CO1, CO2 |
|  | Analyze the performance of Quadrature Amplitude Modulation (QAM) and M-ary Phase Shift Keying (PSK) scheme in AWGN channel, and compare the results with theoretical results. | CO3 |
|  | Compute Bit Error Rate (BER) for different digital modulation schemes in frequency-flat and slowly varying fading channel. | CO3 |
|  | Bit error rate analysis of digital communication receivers with Maximal Ratio Combining (MRC) receives diversity in frequency-flat and slowly varying fading channel. | CO3 |
|  | Bit error rate analysis of digital communication receivers with Equal Gain Combining (EGC) receives diversity in frequency-flat and slowly varying fading channel. | CO3 |
|  | Simulation of Direct Sequence Spread Spectrum (DSSS) techniques. | CO4 |
|  | (a) Measurement of numerical aperture and attenuation loss in analog fiber optic link.(b) Data multiplexing using fiber optic link. | CO5 |
|  | **Test 2** | CO3, CO4& CO5 |

**Experiment 1**

**Write a program for error detection using CRC-CCITT (16 bits) using C**.

**Aim:** To stimulate the polynomial code check sum for CRC - CCITT.

**Theory:** CRC is a method used the detection of errors when data is being transmitted. A CRC is a numeric value computed from the bits in the message to be transmitted. It is appended to the tail of message prior to transmission, the receiver defects the presence of errors in the received message by computing new CRC. This technique is based on binary division, the redundant bits used by CPC are derived by dividing the data unit by predetermined division, the remainder got by division in the CRC which is appended to the end of data unit so that the resulting data unit becomes exactly divisible by the divisor. At the receiver the incoming data unit is divided by the same no. If the remainder is zero that data unit is assumed to be intact and it is accepted. A remainder medicates the data unit has been damaged in transit and therefore must be rejected.

Ex: - 10010101110

**Algorithm:**

1. Given a bit string, append 0S to the end of it (the number of 0s is the same as the degree of the generator polynomial) let B(x) be the polynomial corresponding to B.

2. Divide B(x) by some agreed on polynomial G(x) (generator polynomial) and determine the remainder R(x). This division is to be done using Modulo 2 Division.

3. Define T(x) = B(x) –R(x) (T(x)/G(x) => remainder 0)

4. Transmit T, the bit string corresponding to T(x).

5. Let T’ represent the bit stream the receiver gets and T’(x) the associated polynomial. The receiver divides T1 (x) by G(x). If there is a 0 remainder, the receiver concludes T = T’ and no error occurred otherwise, the receiver concludes an error occurred and requires a retransmission.

#include<stdio.h>

#include<string.h>

#include<conio.h>

#define N strlen(g)

char t[128], cs[128], g[]="10110";

int a, e, c;

void xor() {

for(c=1;c<N;c++) cs[c]=((cs[c]==g[c])?'0':'1');

}

void crc() {

for(e=0;e<N;e++) cs[e]=t[e];

do {

if(cs[0]=='1') xor();

for(c=0;c<N-1;c++) cs[c]=cs[c+1];

cs[c]=t[e++];

}while(e<=a+N-1);

}

void main() {

//rscr();

printf("\nEnter poly : "); scanf("%s",t);

printf("\nGenerating Polynomial is : %s",g);

a=strlen(t);

for(e=a;e<a+N-1;e++) t[e]='0';

printf("\nModified t[u] is : %s",t);

crc();

printf("\nChecksum is : %s",cs);

for(e=a;e<a+N-1;e++) t[e]=cs[e-a];

printf("\nFinal Codeword is : %s",t);

int check,gg;

printf("\nTest Error detection 0(yes) 1(no) ? ");

scanf("%d",&check);

if(check==0){

printf("\nEnter position where you want to insert error : ");

scanf("%d",&gg);

t[gg]=(t[gg]=='1'?'0':'1');

printf("\nErrorneous data : %s",t);

printf("\nError Detected");

}

else{

printf("\nno Error Detected");

}

getch();

}

**Out Put:**

Enter poly:

Generating Polynomial is:

Modified t[u] is:

Checksum is

Final Codeword is:

Test Error detection 0(yes) 1(no) ?:

Enter position where you want to insert error:

Erroneous data:

Error detected.

**Experiment 2**

**Write a program for a HLDC frame to perform bit stuffing and de-stuffing in a single frame.**

**Aim:** To write and execute program for bit stuffing and de-stuffing

**Theory:** When a data stream is transmitted over a channel it may get deviated due to the noise, interference and other nonlinear Characteristics of the nature. Therefore for a purpose of error Detection and correction the ordinary raw data is divided into Frames and compute the checksum for that.

**Checksum:** Error detection method used by the higher level protocol check sum generator divides the data into equal segments of n bits, add the segments and complement the result called checksum field is added to the end of the original data unit as redundancy bits. At the reception the checksum is again computed locally calculated checksum are matched if both are same there is no error else there is an error.

In bit stuffing, each frame begins with a special binary code 01111110 called as flag when error data encounter 5 consecutive ones in the data ,automatically 6th bit is forcibly made 0 at the receiver it sees that 5 ones with o then automatically de stuff or it will detect the 0 to get 1.

**Example 1:**

Original Data:

Stuffed data:

**Algorithm for bit stuffing:**

1. Input the data bit stream

2. Check for 5 consecutive ones

3. If present add 0 to the sixth bit

4. If not continue to further bit

5. Add start and end frame

6. Transmit and display

#include<stdio.h>

#include<conio.h>

#include<string.h>

void main()

{

{

char ch,arr[50]={"01111110"},rec[50];

int i,j,k,len=8,cnt=0;

printf("\n enter the data:\n");

while((ch=getche())!='\r')

{

if(ch=='1')

cnt++;

else

cnt=0;

arr[len++]=ch;

if(cnt==5)

{

arr[len++]='0';

cnt=0;

}

}

strcat(arr,"01111110");

printf("\n bit stuffed stream is:\n ");

for(i=0;i<len+8;i++)

printf("%c",arr[i]);

//destuffing

cnt=0;

printf("\n the destuffed stream is: \n");

for(j=8,k=0;j<len;j++)

{

if(arr[j]=='1')

cnt++;

else

cnt=0;

rec[k++]=arr[j];

if(cnt==5&&arr[j+1]=='0')

{

j++;

cnt=0;

}

}

for(j=0;j<k;j++)

printf("%c",rec[j]);

}

getch();

}

**Output**

Original Data: 0111000111111010

Stuffed data: **01111110**011100011111**0**1010**01111110**

**Experiment 3**

**Write a program for a HLDC frame to perform character stuffing and de-stuffing in a single frame**

**Aim:** To write and execute program for character stuffing and de-stuffing.

**Theory:** In character stuffing, at the starting an ASCII sequence **DLESTX** and at the end **DLEETX** is transmitted for synchronization But when DLE occurs in the data itself then the systems adds one more DLE alone with present one so at the receiver it detects one DLE such that DLE of data will be safe.

**Example:**

Original data RAMA

Stuffed data: **DLESTX** RAMA **DLEETX**

Original data DLE RAMADLE

Stuffed data: **DLESTX DLE**DLERAMADLE **DLE DLEETX**

**Program:**

#include<stdio.h>

#include<conio.h>

#define DLE 16

#define STX 2

#define ETX 3

void main()

{

char ch,arr[50]={DLE,STX},rec[50];

int len=2,i,j;

//clrscr();

printf("enter the data stream:ctrl+p->DLE ctrl+b->STX ctrl+c->ETX \n");

while((ch=getch())!='\r')

{

if(ch==DLE)

{

arr[len++]=DLE;

printf("DLE");

}

else if(ch==STX)

printf("STX");

else if(ch==ETX)

printf("ETX");

else printf("%c",ch);

arr[len++]=ch;

}

arr[len++]=DLE;

arr[len++]=ETX;

printf("\n the stuffed stream is:\n");

for(i=0;i<len;i++)

{

ch=arr[i];

if(ch==DLE)

printf("DLE");

else if(ch==STX)

printf("STX");

else if(ch==ETX)

printf("ETX");

else printf("%c",ch);

}

//destuffing

printf("\n the destuffed data dtream is:\n");

for(j=2;j<len-2;j++)

{

ch=arr[j];

if(ch==DLE)

{

printf("DLE");

j++;

}

else if(ch==STX)

printf("STX");

else if(ch==ETX)

printf("ETX");

else printf("%c",ch);

}

getch();

}

**Output:**

Original data:

Stuffed data:

Original data:

Stuffed data:

**Experiment 4**

**Write a program for encryption and decryption of text.**

**Aim:** To write and execute a program for encryption and decryption by Using substitution method

**Theory:** Cryptography (the branch of cryptology dealing with the design of algorithm for encryption and decryption intended entirely for the secrecy and /or authentically of messages) is used for network data protection, privacy and security. In which the message to be encrypted known as plain text are transformed by a function that is a Parameterized by a key. The output of encryption process known as cipher text is then transmitted. In substitution method each letter or group of letters is replaced by another letter or group of letters to disguise it. The cipher text alphabets to be shifted by K letters. In this case K becomes a key to the generous method of circularly shifted alphabets.

**Example:**

1. Plain text**:** RAMA

Key=2

Cipher text: TCOC

2. Plain text: XYZ

Key=3

Cipher text: ABC

**Algorithm:**

1. Get the message to be encrypted into plain text

2. Initialize the number of shifts per character for encryption

3. Remove non alphabetical character from the plain text and capitalize them

4. Replace each character of the plain text by shifting letters along the forward direction and put

the encrypted message to the cipher text.

5. For decrypting initialize the key K

6. Replace each character of cipher text by shifting k letters along the reverse direction and put

the decrypted message to plain text.

#include <stdio.h>

#include <ctype.h>

#include <conio.h>

#define MAX 1000

int main()

{

{

int s, pi, ci;

char plain[MAX], cipher[MAX];

printf("\*\*\* Encryption & decryption using substitution cipher \*\*\*\n\n");

printf("Enter the plain text:\n");

gets(plain);

while(1)

{

printf("\nKey (number of shifts per character) for encryption : ");

scanf("%d", &s);

if(s < 1 || s > 25)

printf("Bad input! Enter a value between 1 and 25.");

else

break;

}

printf("\nAfter removing non alphabetical characters and capitalizing:\n");

for(ci = 0, pi = 0; plain[pi] != '\0'; pi++)

if(isalpha(plain[pi]))

{

putchar(toupper(plain[pi]));

cipher[ci++] = ((toupper(plain[pi]) - 'A') + s% 26) % 26 + 'A';

}

cipher[ci] = '\0';

printf("\n\nAfter encryption:\n%s\n", cipher);

while(1)

{

printf("\nKey for decryption : ");

scanf("%d", &s);

if(s< 1 || s> 25)

printf("Bad input! Enter a value between 1 and 25.");

else

break;

}

for(pi = 0, ci = 0; cipher[ci] != '\0'; ci++)

plain[pi++] = ((cipher[ci] - 'A') + (26 - s)) % 26 + 'A';

plain[pi] = '\0';

printf("\nAfter decryption:\n%s", plain);

//return 0;

}

getch();

}

**Output:**

Plain text**:** RAMA

Key=2

Cipher text: TCOC

Plain text: XYZ

Key=3

Cipher text: ABC

**Experiment 5**

**Simulate a three node point-to-point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped using NS3.**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"

#include "ns3/traffic-control-module.h"

#include "ns3/flow-monitor-module.h"

Network topology

10.1.1.0 10.1.2.0 10.1.3.0

n0 -------------- n1 ----------------------n2--------------------n3

point-to-point point-to-point point-to-point

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("TrafficControlExample");

int

main (int argc, char \*argv[])

{

double simulationTime = 10; //seconds

std::string transportProt = "Udp";

std::string socketType;

//CommandLine cmd;

//cmd.AddValue ("transportProt", "Transport protocol to use: Tcp, Udp", transportProt);

cmd.Parse (argc, argv);

if (transportProt.compare ("Tcp") == 0)

{

socketType = "ns3::TcpSocketFactory";

}

else

{

socketType = "ns3::UdpSocketFactory";

}

NodeContainer nodes;

nodes.Create (3);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("8Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

pointToPoint.SetQueue ("ns3::DropTailQueue", "Mode", StringValue ("QUEUE\_MODE\_PACKETS"), "MaxPackets", UintegerValue (1));

//NetDeviceContainer devices;

//devices = pointToPoint.Install (nodes);

NetDeviceContainer devices01 = pointToPoint.Install (nodes.Get(0),nodes.Get(1));

NetDeviceContainer devices12 = pointToPoint.Install (nodes.Get(1),nodes.Get(2));

InternetStackHelper stack;

stack.Install (nodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

//Ipv4InterfaceContainer interfaces = address.Assign (devices);

Ipv4InterfaceContainer interfaces01 = address.Assign (devices01);

address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces12 = address.Assign (devices12);

Ipv4GlobalRoutingHelper::PopulateRoutingTables (); //take from examples/tutorial/third.cc

//Flow

uint16\_t port = 7;

Address localAddress (InetSocketAddress (Ipv4Address::GetAny (), port));

PacketSinkHelper packetSinkHelper (socketType, localAddress);

ApplicationContainer sinkApp = packetSinkHelper.Install (nodes.Get (2));

sinkApp.Start (Seconds (0.0));

sinkApp.Stop (Seconds (simulationTime + 0.1));

uint32\_t payloadSize = 1448;

Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));

OnOffHelper onoff (socketType, Ipv4Address::GetAny ());

onoff.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));

onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));

onoff.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s

ApplicationContainer apps;

AddressValue remoteAddress (InetSocketAddress (interfaces12.GetAddress(1), port));

onoff.SetAttribute ("Remote", remoteAddress);

apps.Add (onoff.Install (nodes.Get (0)));

apps.Start (Seconds (1.0));

apps.Stop (Seconds (simulationTime + 0.1));

FlowMonitorHelper flowmon;

Ptr<FlowMonitor> monitor = flowmon.InstallAll();

Simulator::Stop (Seconds (simulationTime + 5));

Simulator::Run ();

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());

std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

std::cout << std::endl << "\*\*\* Flow monitor statistics \*\*\*" << std::endl;

std::cout << " Tx Packets: " << stats[1].txPackets << std::endl;

std::cout << " Dropped Packets: " << stats[1].lostPackets << std::endl;

Simulator::Destroy ();

return 0;

}

**Experiment 6**

**Simulate a three node point-to-point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped using NS3.**

#include "ns3/core-module.h"

#include "ns3/network-module.h"

#include "ns3/internet-module.h"

#include "ns3/point-to-point-module.h"

#include "ns3/applications-module.h"

#include "ns3/traffic-control-module.h"

#include "ns3/flow-monitor-module.h"

// // Network topology

//

// n0

// \ 10.1.2.0

// \ 10.1.3.0

// n2 -------------------------n3

// /

// / 10.1.1.0

// n1

using namespace ns3;

NS\_LOG\_COMPONENT\_DEFINE ("TrafficControlExample");

int

main (int argc, char \*argv[])

{

double simulationTime = 10; //seconds

std::string transportProt = "Udp";

std::string socketType;

CommandLine cmd;

//cmd.AddValue ("transportProt", "Transport protocol to use: Tcp, Udp", transportProt);

cmd.Parse (argc, argv);

if (transportProt.compare ("Tcp") == 0)

{

socketType = "ns3::TcpSocketFactory";

}

else

{

socketType = "ns3::UdpSocketFactory";

}

NodeContainer nodes;

nodes.Create (4);

PointToPointHelper pointToPoint;

pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));

pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));

pointToPoint.SetQueue ("ns3::DropTailQueue", "Mode", StringValue ("QUEUE\_MODE\_PACKETS"), "MaxPackets", UintegerValue (1));

//NetDeviceContainer devices;

//devices = pointToPoint.Install (nodes);

NetDeviceContainer devices01 = pointToPoint.Install (nodes.Get(0),nodes.Get(1));

NetDeviceContainer devices02 = pointToPoint.Install (nodes.Get(0),nodes.Get(2));

NetDeviceContainer devices12 = pointToPoint.Install (nodes.Get(1),nodes.Get(2));

NetDeviceContainer devices23 = pointToPoint.Install (nodes.Get(2),nodes.Get(3));

InternetStackHelper stack;

stack.Install (nodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");

//Ipv4InterfaceContainer interfaces = address.Assign (devices);

Ipv4InterfaceContainer interfaces02 = address.Assign (devices02);

address.SetBase ("10.1.2.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces12 = address.Assign (devices12);

address.SetBase ("10.1.3.0", "255.255.255.0");

Ipv4InterfaceContainer interfaces23 = address.Assign (devices23);

Ipv4GlobalRoutingHelper::PopulateRoutingTables (); //take from examples/tutorial/third.cc

//UDP Flow

uint16\_t port = 7;

Address localAddress (InetSocketAddress (Ipv4Address::GetAny (), port));

PacketSinkHelper packetSinkHelper (socketType, localAddress);

ApplicationContainer sinkApp = packetSinkHelper.Install (nodes.Get (3));

sinkApp.Start (Seconds (0.0));

sinkApp.Stop (Seconds (simulationTime + 0.1));

uint32\_t payloadSize = 1448;

Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));

OnOffHelper onoff (socketType, Ipv4Address::GetAny ());

onoff.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));

onoff.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

onoff.SetAttribute ("PacketSize", UintegerValue (payloadSize));

onoff.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s

ApplicationContainer apps;

AddressValue remoteAddress (InetSocketAddress (interfaces23.GetAddress(1), port));

onoff.SetAttribute ("Remote", remoteAddress);

apps.Add (onoff.Install (nodes.Get (0)));

apps.Start (Seconds (1.0));

apps.Stop (Seconds (simulationTime + 0.1));

//TCP Flow

uint16\_t port\_tcp = 9;

socketType = "ns3::TcpSocketFactory"; //Add this line

Address localAddress\_tcp (InetSocketAddress (Ipv4Address::GetAny (), port\_tcp));

PacketSinkHelper packetSinkHelper\_tcp (socketType, localAddress\_tcp);

ApplicationContainer sinkApp\_tcp = packetSinkHelper\_tcp.Install (nodes.Get (3));

sinkApp\_tcp.Start (Seconds (0.5));

sinkApp\_tcp.Stop (Seconds (simulationTime + 0.1));

Config::SetDefault ("ns3::TcpSocket::SegmentSize", UintegerValue (payloadSize));

OnOffHelper onoff\_tcp (socketType, Ipv4Address::GetAny ());

onoff\_tcp.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));

onoff\_tcp.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));

onoff\_tcp.SetAttribute ("PacketSize", UintegerValue (payloadSize));

onoff\_tcp.SetAttribute ("DataRate", StringValue ("50Mbps")); //bit/s

ApplicationContainer apps\_tcp;

AddressValue remoteAddress\_tcp (InetSocketAddress (interfaces23.GetAddress (1), port\_tcp));

onoff\_tcp.SetAttribute ("Remote", remoteAddress\_tcp);

apps\_tcp.Add (onoff\_tcp.Install (nodes.Get (1)));

apps\_tcp.Start (Seconds (1.5));

apps\_tcp.Stop (Seconds (simulationTime + 0.1));

FlowMonitorHelper flowmon;

Ptr<FlowMonitor> monitor = flowmon.InstallAll();

Simulator::Stop (Seconds (simulationTime + 5));

Simulator::Run ();

Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmon.GetClassifier ());

std::map<FlowId, FlowMonitor::FlowStats> stats = monitor->GetFlowStats ();

std::cout << std::endl << "\*\*\* Flow monitor statistics \*\*\*" << std::endl;

std::cout << " Tx Packets: " << stats[1].txPackets << std::endl;

std::cout << " Dropped Packets: " << stats[1].lostPackets << std::endl;

for (std::map<FlowId, FlowMonitor::FlowStats>::const\_iterator iter = stats.begin (); iter != stats.end (); ++iter)

{

Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (iter->first);

std::cout << "Flow ID: " << iter->first << " Src Addr " << t.sourceAddress << " Dst Addr " << t.destinationAddress<< std::endl;

std::cout << "Tx Packets = " << iter->second.txPackets<< std::endl;

}

Simulator::Destroy ();

return 0;

}

# PART II

# Wireless Communication Experiments

**Experiment 1**

**Aim**

Write a code to analyze the performance of Quadrature Amplitude Modulation (QAM) and M-ary Phase Shift Keying (PSK) scheme in AWGN channel, and compare the results with theoretical results.

# Algorithm

Step1: Generate random binary data

Step2: Construct the constellation symbols for the generated binary sequence

Step3: Generate complex normal random variable for additive white Gaussian noise (AWGN) Step4: Add AWGN noise with the transmitted symbol

Step5: Construct maximum likelihood (ML) receiver to decode the symbol from the received signal

Step6: Compare the decoded sequence with the original sequence to estimate the number of errors

Step7: Average bit error rate (ABER) can be calculated by computing the ratio between total number of errors and total number of bits

# Matlab Code:

clc; close all; clear all;

% Number of information bits

m= 10^5;

%Range of SNR values snr\_dB = [0:1:10];

for j=1:1:length(snr\_dB) n\_err = 0;

n\_bits = 0;

while n\_err < 100

% Generate sequence of binary bits inf\_bits=round(rand(1,m));

% BPSK Constellation symbols x=-2\*(inf\_bits-0.5);

% Noise variance N0=1/10^(snr\_dB(j)/10);

% Send over Gaussian Link to the receiver

y=x + sqrt(N0/2)\*(randn(1,length(x))+i\*randn(1,length(x)));

% Decision making at the Receiver est\_bits=y < 0;

% Calculate Bit Errors diff=inf\_bits-est\_bits;

n\_err=n\_err+sum(abs(diff)); n\_bits=n\_bits+length(inf\_bits);

end

% Calculate Bit Error Rate BER(j)=n\_err/n\_bits;

end

% AWGN Theoretical BER theoryBerAWGN=0.5\*erfc(sqrt(10.^(snr\_dB/10))); semilogy(snr\_dB,theoryBer,'-','LineWidth',2);

hold on; semilogy(snr\_dB,BER,'or','LineWidth',2); hold on;

semilogy(snr\_dB,theoryBerAWGN,'blad-','LineWidth',2); legend('AWGN Simulated', 'AWGN Theoretical');

axis([0 20 10^-5 0.5]);

xlabel('SNR (dB)');

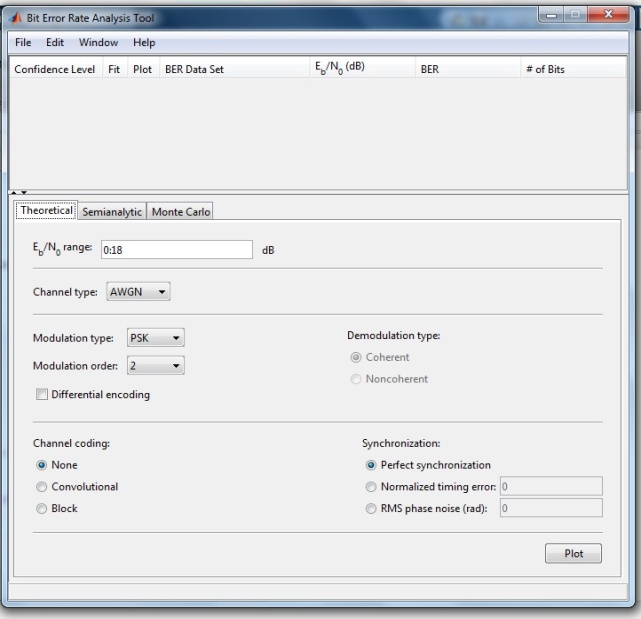
ylabel('BER'); grid on;

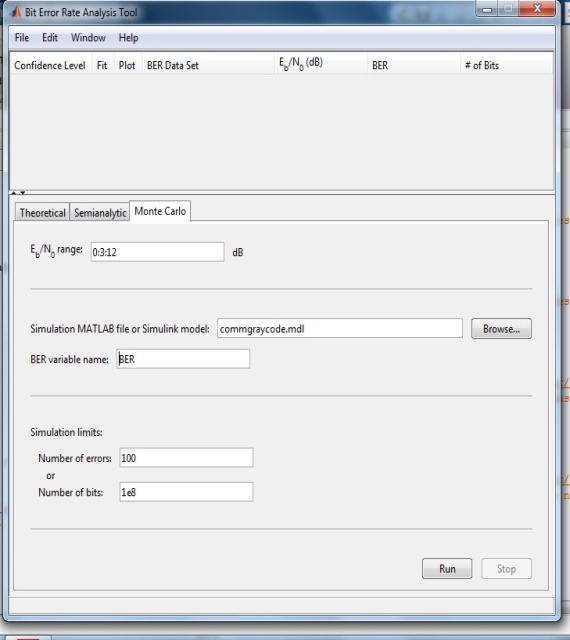
Compare the simulation results with the theoretical results, Repeat the procedure for different modulation schemes and plot the performance curve (ABER Vs SNR)

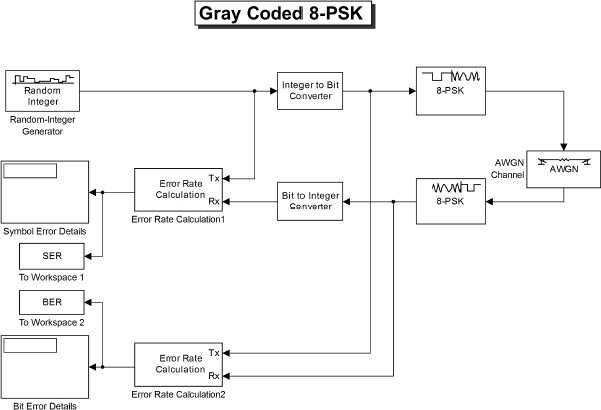
Compare and discuss the performance of different modulation schemes

# Simulink Model Procedure:

* Type Simulink in the command window to open Simulink
* Construct the communication model in a new Simulink file
* Type bertool in a command window
* Select theoretical option and choose the corresponding information for your modulation scheme
* Plot the theoretical curve
* Then, select Monte Carlo option and link the path of your saved Simulink model
* Plot the simulation curve and compare the simulation results with the theoretical results







**Experiment 2**

**Aim**

Write a code to compute Bit Error Rate (BER) for different digital modulation scheme in frequency-flat and slowly varying fading channel.

# Algorithm

Step1: Generate random binary data

Step2: Construct the constellation symbols for the generated binary sequence Step3: Generate complex channel fading coefficient

Step3: Generate complex normal random variable for additive white Gaussian noise (AWGN) Step4: Add AWGN noise with the transmitted symbol

Step5: Construct maximum likelihood (ML) receiver to decode the symbol from the received signal

Step6: Compare the decoded sequence with the original sequence to estimate the number of errors

Step7: Average bit error rate (ABER) can be calculated by computing the ratio between total number of errors and total number of bits

Repeat the procedure for different modulation schemes and plot the performance curve ABER Vs SNR

clc;

close all;

clear all;

% Number of information bits

m= 10^5;

%Range of SNR values

snr\_dB = [0:1:20];

for j=1:1:length(snr\_dB)

n\_err = 0;

n\_bits = 0;

while n\_err < 100

% Generate sequence of binary bits

inf\_bits=round(rand(1,m));

% BPSK modulator

x=-2\*(inf\_bits-0.5);

% Noise variance

N0=1/10^(snr\_dB(j)/10);

% Rayleigh channel fading

h=1/sqrt(2)\*[randn(1,length(x)) + i\*randn(1,length(x))];

% Send over Gaussian Link to the receiver

y=h.\*x + sqrt(N0/2)\*(randn(1,length(x))+i\*randn(1,length(x)));

% decision metric

y=y./h;

% Decision making at the Receiver

est\_bits=y < 0;

% Calculate Bit Errors

diff=inf\_bits-est\_bits;

n\_err=n\_err+sum(abs(diff));

n\_bits=n\_bits+length(inf\_bits);

end

% Calculate Bit Error Rate

BER(j)=n\_err/n\_bits;

end

% Rayleigh Theoretical BER

snr = 10.^(snr\_dB/10);

theoryBer=0.5.\*(1-sqrt(snr./(snr+1)));

% AWGN Theoretical BER

theoryBerAWGN=0.5\*erfc(sqrt(10.^(snr\_dB/10)));

semilogy(snr\_dB,theoryBer,'-','LineWidth',2);

hold on;

semilogy(snr\_dB,BER,'or','LineWidth',2);

hold on;

semilogy(snr\_dB,theoryBerAWGN,'blad-','LineWidth',2);

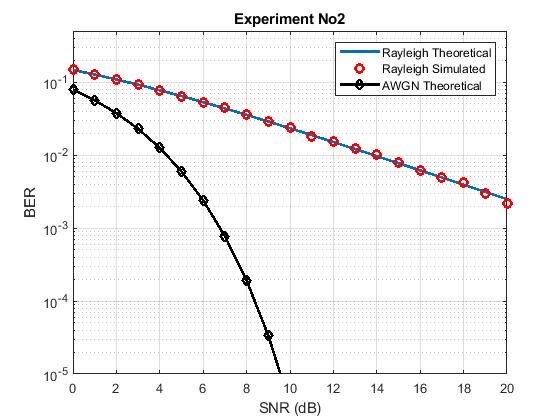
legend('Rayleigh Theoretical','Rayleigh Simulated', 'AWGN Theoretical');

axis([0 20 10^-5 0.5]);

xlabel('SNR (dB)');

ylabel('BER');

grid on;



**Experiment 3**

**Aim**

Bit error rate analysis of digital communication receivers with Maximal Ratio Combining (MRC) receive diversity in frequency-flat and slowly varying fading channel.

clc;

clear all;

close all;

%%%%%%%%%%%%%%% Initialization %%%%%%%%%%%%%%%%%%%%%%%%

N=5; % Number of trials

m = 10^6; %Number of bits in each trial

ip = rand(1,m)>0.5; % Generated bits

BPSK = 2\*ip-1; % Generated BPSK symbols

snr\_dB = 0:1:15; % range of snr values

snr = 10.^(snr\_dB/10); % snr value in the normal scale

L=2; % Number of diversity branches

% theoretical BER value for MRC combiner with 2 diversity branches

p\_R\_MRC = 1/2 - 1/2\*(1+1./snr).^(-1/2);

ber\_MRC\_ana = p\_R\_MRC.^2.\*(1+2\*(1-p\_R\_MRC));

%%%%%%%%%% Receive MRC one by Two System %%%%%%%%%%%%%%%%%%%

n\_err=zeros(1,length(snr\_dB)); % Initialize the bit error counter

for p = 1:N

for q = 1:length(snr\_dB)

% Generate white noise samples

No = 1/sqrt(2)\*[randn(L,m) + 1j\*randn(L,m)];

% Generate channel coefficient

h = 1/sqrt(2)\*[randn(L,m) + 1j\*randn(L,m)];

symbol = kron(ones(L,1),BPSK); % array of symbols

rec\_vector = h.\*symbol + 10^(-snr\_dB(q)/20)\*No;% received symbol

% Decision metric

dec\_metric = sum(conj(h).\*rec\_vector,1)./sum(h.\*conj(h),1);

ip\_hat = real(dec\_metric)>0; % Estimated symbol

n\_err(q) = n\_err(q)+size(find([ip- ip\_hat]),2); % compare input and estimated symbols

end

end

ber\_MRC\_sim = n\_err/(N\*m);

semilogy(snr\_dB,ber\_MRC\_ana,'-r\*','LineWidth',2)

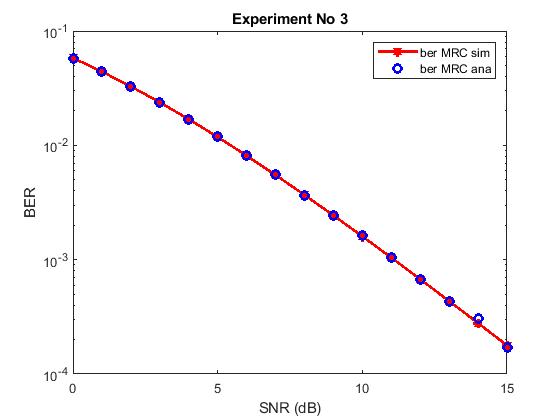
hold on;

semilogy(snr\_dB,ber\_MRC\_sim,'ob','LineWidth',2)

legend('ber MRC sim', 'ber MRC ana');

xlabel('SNR (dB)');

ylabel('BER');



**Experiment 4**

**Aim**

Bit error rate analysis of digital communication receivers with Equal Gain Combining (EGC) receive diversity in frequency-flat and slowly varying fading channel.

clc;

close all;

clear all;

% Number of information bits

m= 10^3;

%Range of SNR values

snr\_dB = [0:1:20];

for j=1:1:length(snr\_dB)

n\_err = 0;

n\_bits = 0;

while n\_err < 100

inf\_bits=round(rand(1,m));

% BPSK modulator

x=-2\*(inf\_bits-0.5);

% Noise variance

N0=1/10^(snr\_dB(j)/10);

n1 = sqrt(N0/2)\*abs((randn(1,length(x)) + i\*randn(1,length(x)))); %noise for the first

n2 = sqrt(N0/2)\*abs((randn(1,length(x)) + i\*randn(1,length(x)))); %noise for the first

h1 = sqrt(0.5)\*abs((randn(1,length(x)) + i\*randn(1,length(x)))); %rayleigh amplitude 1

h2 = sqrt(0.5)\*abs((randn(1,length(x)) + i\*randn(1,length(x)))); %rayleigh amplitude 1

%Equal Gain combining

y1 = h1.\*x+n1; % Signal 1

y2 = h2.\*x+n2; % Signal 2

y\_equal = 0.5\*(y1+y2);

% dec\_metric=(norm(y\_equal- h1\*x-h2\*x))^2;

% Decision making at the Receiver

est\_bits=y\_equal < 0;

% Calculate Bit Errors

diff=inf\_bits-est\_bits;

n\_err=n\_err+sum(abs(diff));

n\_bits=n\_bits+length(inf\_bits);

end

% Calculate Bit Error Rate

BER(j)=n\_err/n\_bits;

end

semilogy(snr\_dB,BER,'or-','LineWidth',2);

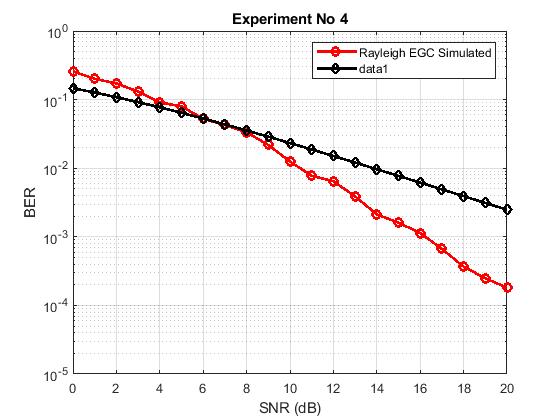
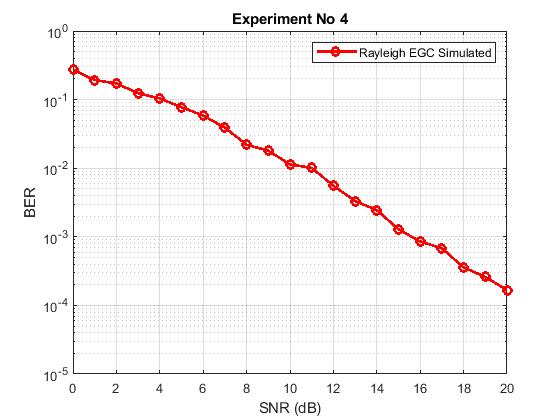
legend('Rayleigh EGC Simulated', 'Rayleigh Theoretical');

axis([0 20 10^-5 1]);

xlabel('SNR (dB)');

ylabel('BER');

grid on;



**Experiment 5**

**Aim**

To study the performance of the BPSK direct sequence spread spectrum system in AWGN channel.

clc;

close all;

clear all;

% Number of information bits

m= 10;

f\_data=1; % DATA FREQUENCY

f\_chip=7; % LENGTH OF CHIPSEQUENCE

fc=210; % RELATIVE CARRIER FREQUENCY

fs=fc\*3; % SAMPLING FREQUENCY

N=fs/f\_chip;% CODING RATE

%Range of SNR values

snr\_dB = [0:2:26];

for jj=1:1:length(snr\_dB) n\_err = 0;

n\_bits = 0;

while n\_err < 100

% Generate sequence of binary bits inf\_bits=round(rand(1,m));

PN\_sequence=round(rand(11,1)); % GENERATION OF PN SEQUENCE

%Spread the information bits with PN sequence

j=1;

for i = 1:m

for k = j:j+f\_chip-1 msg\_spread(k)=inf\_bits(i);

end;

msg\_spread(j:(j+f\_chip-1)) = xor(msg\_spread(j:(j+f\_chip-1))',PN\_sequence(1:f\_chip));

j = f\_chip\*i+1;

end; len\_msg\_spr=length(msg\_spread);

% MODULATING THE SPREAD MESSAGE

% BPSK Constellation symbols

x=-2\*(msg\_spread-0.5);

% Noise variance N0=1/10^(snr\_dB(jj)/10);

% Send over Gaussian Link to the receiver

y=x + sqrt(N0/2)\*(randn(1,length(x))+i\*randn(1,length(x)));

% Decision making at the Receiver msg\_demod=y < 0;

% CORRELATING WITH THE PN SEQUENCE

j=1;

for i = 1:m

msg\_demod(j:(j+f\_chip-1)) = xor(msg\_demod(j:(j+f\_chip-1))',PN\_sequence(1:f\_chip)); j = f\_chip\*i+1;

end;

% DESPREADING THE RECEIVED SIGNAL

j=1;

for i = 1:m s1=0;

for k = j:j+f\_chip1 s1=s1+msg\_demod(k);

end;

if (s1>=6) msg\_demod\_rec(i)=1;

else

msg\_demod\_rec(i)=0;

end;

j=f\_chip\*i+1;

end;

% Calculate Bit Errors

diff=inf\_bits-msg\_demod\_rec; n\_err=n\_err+sum(abs(diff)); n\_bits=n\_bits+length(inf\_bits);

end

% Calculate Bit Error Rate BER(jj)=n\_err/n\_bits

end

semilogy(snr\_dB,BER,'or-','LineWidth',2); legend('AWGN Simulated');

axis([0 26 10^-4 1]);

xlabel('SNR (dB)');

ylabel('BER'); grid on;

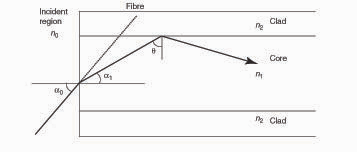
**Experiment 6**

Determination of Numerical aperture, Attenuation (propagation) loss, Bending loss and coupling loss and of a given fiber using OFT. Measurement of bit rate using digital link and study of TDM.

**Fiber-optic communication** is a method of transmitting information from one place to another by sending pulses of light through an optical fibre. The light forms an electromagnetic carrier wave that is modulated to carry information. First developed in the 1970s, fibre-optic communication systems have revolutionized the telecommunications industry and have played a major role in the advent of the Information Age.

The process of communicating using fibre-optics involves the following basic steps: Creating the optical signal involving the use a transmitter, relaying the signal along the fibre, ensuring that the signal does not become too distorted or weak, receiving the optical signal, and converting it into an electrical signal. Light pulses move easily down the fibre-optic line because of a principle known as total internal reflection, which states that *when the angle of incidence exceeds a critical value, light cannot get out of the fibre; instead, the light bounces back in*.

The **index of refraction** is a way of measuring the speed of light in a material. Light travels fastest in vacuum, such as outer space. The actual speed of light in vacuum is 299,792 kilometers per second. Index of refraction is calculated by dividing the speed of light in vacuum by the speed of light in some other medium. The index of refraction of a vacuum is therefore 1, by definition. The typical value for the cladding of an optical fibre is 1.46. The core value is typically 1.48. The larger the index of refraction the more slowly light travels in that medium. When light traveling in a dense medium hits a boundary at a steep angle (larger than the *critical angle* for the boundary), the light will be completely reflected. This effect is used in optical fibres to confine light in the core. Light travels along the fibre bouncing back and forth off of the boundary. Because the light must strike the boundary with an angle less than the critical angle, only light that enters the fibre within a certain range of angles can travel down the fibre without leaking out. This range of angles is called the acceptance cone of the fibre. The size of this acceptance cone is a function of the refractive index difference between the fibre’s core and cladding.



**Figure 6.1 : *Internal reflection in fiber***

Figure 6.1 illustrates, total internal reflection at core-cladding boundary of fibre. The numerical aperture is defined as *n*0 sin *a*0, where *n*0 is the refractive index of the region *(n*0 = 1*)* and *a*0 is the maximum angle at which the ray can enter a fibre and propagate. Only if a ray enters the fibre at an angle less then *a*0, it undergoes total internal reflection at the core cladding boundary as shown in the figure. At this incident region, angle *θ* is the critical angle.

An **Optical Fiber kit** (OFT) helps to study the principles of Fiber Optic Communication in a laboratory. It generally consists of the following elements:

1. Optical Transmitter, including Electrical to Optical Convertor

2. Optical Receiver, including Optical to Electrical Convertor

3. Optical Fiber

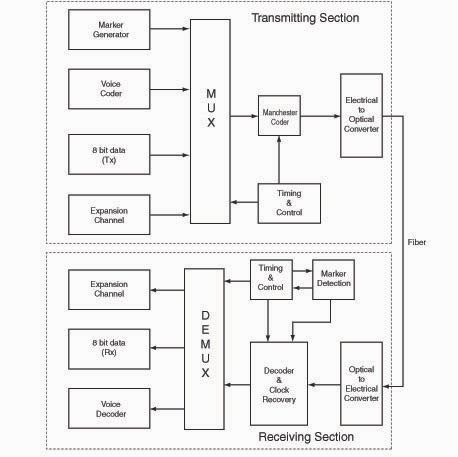
4. Timing Recovery Unit

5. Line Coder

6. Line Decoder

7. Voice Coders and Multiplexer

8. Demultiplexer and Voice Decoders



**Figure 6.2: *Block diagram of an OFT***

A typical block diagram of an OFT illustrating different elements is shown in Figure 9.2. The fibre cable used in trainer kits generally is of plastic due to its easiness of termination and the distance of application is very less. The wave lengths of the light used in OFT are typically 850 nm and 650 nm. An LED at the transmitter end converts electrical signal in to light and a photo transistor is used to convert back the light in to electrical signal at the receiver end. The fibre carries the signal in terms of light from transmitter to receiver.

blockdiagram-losses

**Figure 6.3: *Block diagram – Attenuation and bending loss***

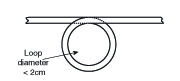
Block diagram to conduct the experiment is shown in Figure 9.3. An analog signal is applied to the specified input point that is converted as light by means of an LED. A fibre cable is used to carry the light from transmitter point to the receiver. At the receiver end, the light is converted back to an electrical signal using a photo detector. The received signal can be observed on an oscilloscope and compared with the transmitted signal. Gain control knob provided in the kit can be adjusted for the required gain.

**Procedure**

1. **Bending loss:**

* The OFT connections are made as per the instruction.
* Switch on the DC power supply.
* Apply a sinusoidal signal of 1V, 1 kHz as the message signal at the transmitting end.
* Use a One meter fiber between transmitting and receiving end.
* Check the output signal at the receiving end. Make sure that there is no bend in the fiber cable. Adjust the gain control knob to get the required output.
* Make a loop (diameter more than 1 cm) with the cable as shown in Figure 9.4 and observe the change in the output.

|  |  |
| --- | --- |
| **Tabular column for bending loss** | |
| ***Diameter of loop*** | ***Output*** |
| 2 cm  2.5 cm  3 cm  3.5 cm  4.0 cm |  |



**Figure 6.4: *Bending loss***

* Observe the output for different loop diameters and tabulate the magnitude of the output.
* Now you can have a feeling of the bending loss by observing the output versus bending diameter.
* A graph may be plotted with the output against the diameter of loop.

1. **Propagation loss:**

* Now tabulate the output *(V*1*)*with 1meter *(l*1*)*fibre without any bend.
* Replace the 1meter fibre with a 3 meter fibre *(l*3*)* and tabulate the output *(V*3*)*, without changing the input signal.
* Make sure that gain control knob is not disturbed.
* Calculate the propagation loss using the relation,



1. **Coupling loss:**

* Connect one end of the 1m fiber to LED (transmitter) and other end to the detector PD1 (receiver).
* Measure the detector output and designate it as V1.
* Disconnect the fiber from the detector. (keep the other end at the transmitter itself)
* Connect one end of the fiber to the detector.
* Bring the free ends of both the cables (1m and 3m) as close as possible and align them using the fiber alignment unit as shown in figure 9.5.
* Measure the detector output and designate it as V4.
* Compute the coupling loss using the formula



|  |
| --- |
| coupling loss 001 |
| **Fig. 6.5 Fiber allignment using the Alignment Unit** |

1. **Numerical Apperture**

* Insert one end of the fiber into the Numerical Aperture Measurement Unit as shown in fig 6.6.
* Connect the other end of the fibre to LED2. Let d be the distance between the fiber tip and the screen.
* Measure the diameter of the circular patch of red light in tow perpendicular directions (BC and DE in fig 9.7). The mean radius of the circular patch is given by



* Carefully measure the distance d between the tip of the fibre and the illuminated screen (OA in fig. 6.6).
* The numerical aperture is given by



* Repeat the steps for different values of d and find the average value of numerical aperture.

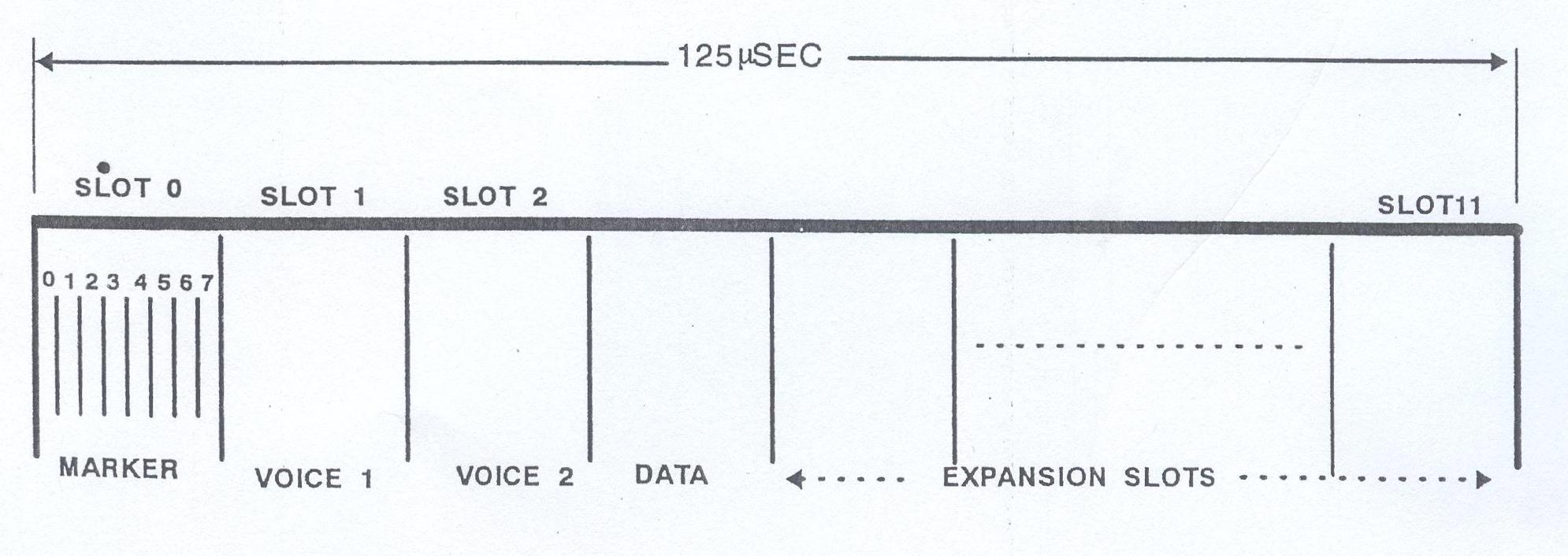
|  |  |
| --- | --- |
| numerical app 001 | numerical app 002 |
| **Fig: 6.6 Numerical aperture measurement setup** | **Fig 6.7 Measurment of radius** |

1. **Maximum bit rate:**

* The OFT connections are made as per the instruction.
* Switch on the DC power supply.
* Apply a TTL signal (0-5V), 1 kHz as the message signal at the transmitting end.
* Use a One meter fiber between transmitting and receiving end.
* check the output signal at the receiving end
* Vary the frequency and observe the output.
* Find the maximum frequency the digital link can transmit
* Calculate the maximum bit rate by multiplying the maximum transmitted frequency by 2.

1. **TDM:**

* The OFT connections are made as per the instruction.
* Switch on the DC power supply.
* Check all the jumper connections are intact.
* Connect a One meter fiber cable between transmitting and receiving end.
* Check the Marker data are in same format at the transmitter and receiver end.
* Use the data switches to set an 8-bit digital data pattern.
* Adjust the gain control knob (if any) to observe the received digital data in the LED block provided at the receiver end.
* Compare the received digital data pattern with the transmitted pattern.
* Use the voice channel to feed an analog signal of 1V at 1kHz.
* Observe the retrieved audio signal at the receiver end and compare with the original audio signal
* Observe the simultaneous transmission of digital and analog signal by changing the data pattern and audio signal simultaneously at the transmitter end.
* To observe the time switching of voice, use the voice channels with handsets instead of audio signals.
* Use mouth piece of one hand set to send the voice and use the ear piece of other to hear the voice and check the performance of TDM.
* Check whether the voice transmitted through one of the hand set can be heard on the other. If not, the system is in loop back mode.
* Make the cross connections at the appropriate block and check the performance again.
* Observe the frame time, slot time, marker data of TDM wave form.



***Result and Conclusion:***

Question Bank WDC

1. a ) Write a MATLAB code to analyze the performance of BPSK scheme in AWGN channel and compare with theoretical results.

b) Write a program for error detection using CRC-CCITT (16 bits) using C.

1. a) Simulate a three node point-to-point network between n1 to n2 and n2 to n3. Set the queue size, vary the bandwidth and find the number of packets dropped and plot the graph using NS3.

b) Determine the Numerical aperture for a given fiber and measure the bit rate using digital link through TDM using OFT kit

1. a) Simulate a four node point-to-point network and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP agent between n1-n3. Apply relevant applications over TCP and UDP agents by changing the parameters and determine the number of packets sent by TCP/UDP using NS3.

b) Simulate the performance of QPSK scheme in AWGN channel and compare with theoretical results and tabulate the same using Simulink.

1. a) Simulate the performance of 8-PSK scheme in AWGN channel and compare with theoretical results and tabulate the same using Simulink.

b) Write a program for a HLDC frame to perform bit stuffing and de-stuffing in a single frame.

1. a) Write a code to compute Bit Error Rate (BER) for BPSK modulation scheme in Rayleigh fading channel.

b) Write a program for a HLDC frame to perform character stuffing and de-stuffing in a single frame.

1. a) Write a MATLAB code to analyze the Bit error rate of digital communication receivers with Maximal Ratio Combining (MRC) diversity in frequency-flat and slowly varying fading channel.

b) Write a program for encryption and decryption of text using ceaser cipher technique.

1. a) Write a MATLAB code to analyze the Bit error rate of digital communication receivers with Equal Gain Combining (EGC) diversity in frequency-flat and slowly varying fading channel.

b) Determine the Attenuation (propagation) loss, bending loss and coupling loss for a given fiber using OFT kit.

1. a) Write a MATLAB code to analyze the performance of the BPSK direct sequence spread spectrum system in AWGN channel.

b) Determine the Attenuation (propagation) loss, bending loss and coupling loss for a given fiber using OFT kit.

1. a) Determine the Numerical aperture for a given fiber and measure the bit rate using digital link through TDM using OFT kit.

b) Write a MATLAB code to analyze the performance of BPSK scheme in AWGN channel, and compare the results with theoretical results.

1. a) Write a MATLAB code to analyze the Bit error rate of digital communication receivers with Maximal Ratio Combining (MRC) diversity in frequency-flat and slowly varying fading channel.

b) Simulate a three node point-to-point network between n1 to n2 and n2 to n3. Set the queue size, vary the bandwidth and find the number of packets dropped and plot the graph using NS3.

1. a) Write a program for error detection using CRC-CCITT (16 bits) using C.

b) Simulate the performance of 8-PSK scheme in AWGN channel and compare with theoretical results and tabulate the same using Simulink.

1. a) Write a program for encryption and decryption of text using ceaser cipher technique with a shift of ‘F’.

b) Simulate the performance of BPSK scheme in AWGN channel and compare with theoretical results and tabulate the same using Simulink.