Appendix A Sample data set collected for BERs and optimum packet size with a header length of 40 bits.

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0001	612.1234	0.0011	169.5892	0.0021	116.4861	0.0031	91.73385	0.0041	76.63188
0.0002	426.7439	0.0012	161.4216	0.0022	113.2767	0.0032	89.91482	0.0042	75.42259
0.0003	344.573	0.0013	154.2118	0.0023	110.2775	0.0033	88.17818	0.0043	74.25512
0.0004	295.5632	0.0014	147.7855	0.0024	107.4661	0.0034	86.51785	0.0044	73.12706
0.0005	262.0996	0.0015	142.0101	0.0025	104.8238	0.0035	84.92834	0.0045	72.03624
0.0006	237.3847	0.0016	136.7821	0.0026	102.3342	0.0036	83.40472	0.0046	70.98061
0.0007	218.1662	0.0017	132.02	0.0027	99.98285	0.0037	81.94251	0.0047	69.95831
0.0008	202.6664	0.0018	127.6582	0.0028	97.75748	0.0038	80.53765	0.0048	68.96759
0.0009	189.8209	0.0019	123.6433	0.0029	95.6471	0.0039	79.18646	0.0049	68.00684
0.001	178.9482	0.002	119.9314	0.003	93.64208	0.004	77.88556	0.005	67.07455

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0051	66.16933	0.0061	58.3568	0.0071	52.22609	0.0081	47.24187	0.0091	43.08086
0.0052	65.28988	0.0062	57.67923	0.0072	51.68263	0.0082	46.79286	0.0092	42.70132
0.0053	64.43498	0.0063	57.0175	0.0073	51.15009	0.0083	46.35174	0.0093	42.32772
0.0054	63.60348	0.0064	56.371	0.0074	50.6281	0.0084	45.91829	0.0094	41.95988
0.0055	62.79433	0.0065	55.73912	0.0075	50.1163	0.0085	45.49227	0.0095	41.59768
0.0056	62.00653	0.0066	55.12133	0.0076	49.61436	0.0086	45.07347	0.0096	41.24094
0.0057	61.23914	0.0067	54.51709	0.0077	49.12196	0.0087	44.66169	0.0097	40.88955
0.0058	60.49127	0.0068	53.92591	0.0078	48.6388	0.0088	44.25673	0.0098	40.54335
0.0059	59.7621	0.0069	53.34731	0.0079	48.16458	0.0089	43.85839	0.0099	40.20223
0.006	59.05086	0.007	52.78084	0.008	47.69903	0.009	43.46649	0.01	39.86605

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0101	39.53468	0.0111	36.46218	0.0121	33.76384	0.0131	31.36722	0.0141	29.21815
0.0102	39.20803	0.0112	36.17688	0.0122	33.51162	0.0132	31.14194	0.0142	29.01517
0.0103	38.88596	0.0113	35.89522	0.0123	33.26234	0.0133	30.91909	0.0143	28.81422
0.0104	38.56837	0.0114	35.61712	0.0124	33.01593	0.0134	30.6986	0.0144	28.61524
0.0105	38.25516	0.0115	35.3425	0.0125	32.77236	0.0135	30.48045	0.0145	28.41822
0.0106	37.94621	0.0116	35.07128	0.0126	32.53155	0.0136	30.26458	0.0146	28.22311
0.0107	37.64144	0.0117	34.8034	0.0127	32.29346	0.0137	30.05096	0.0147	28.02988
0.0108	37.34074	0.0118	34.53878	0.0128	32.05804	0.0138	29.83955	0.0148	27.8385
0.0109	37.04402	0.0119	34.27736	0.0129	31.82522	0.0139	29.6303	0.0149	27.64895
0.011	36.7512	0.012	34.01907	0.013	31.59497	0.014	29.42318	0.015	27.46119

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0151	27.27519	0.0161	25.50602	0.0171	23.88504	0.0181	22.39161	0.0191	21.00894
0.0152	27.09092	0.0162	25.33763	0.0172	23.73026	0.0182	22.24861	0.0192	20.8762
0.0153	26.90836	0.0163	25.1707	0.0173	23.57673	0.0183	22.1067	0.0193	20.74443
0.0154	26.72747	0.0164	25.00519	0.0174	23.42445	0.0184	21.96586	0.0194	20.61359
0.0155	26.54823	0.0165	24.8411	0.0175	23.27338	0.0185	21.82608	0.0195	20.48369
0.0156	26.37062	0.0166	24.6784	0.0176	23.12351	0.0186	21.68736	0.0196	20.3547
0.0157	26.1946	0.0167	24.51707	0.0177	22.97482	0.0187	21.54966	0.0197	20.22663
0.0158	26.02016	0.0168	24.35708	0.0178	22.82731	0.0188	21.41298	0.0198	20.09944
0.0159	25.84726	0.0169	24.19843	0.0179	22.68094	0.0189	21.27731	0.0199	19.97315
0.016	25.67589	0.017	24.04109	0.018	22.53572	0.019	21.14263	0.02	19.84772

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0201	19.72316	0.0211	18.52274	0.0221	17.39797	0.0231	16.34063	0.0241	15.3437
0.0202	19.59945	0.0212	18.40701	0.0222	17.28933	0.0232	16.23833	0.0242	15.24709
0.0203	19.47658	0.0213	18.29202	0.0223	17.18135	0.0233	16.13662	0.0243	15.15101
0.0204	19.35454	0.0214	18.17777	0.0224	17.07403	0.0234	16.0355	0.0244	15.05547
0.0205	19.23333	0.0215	18.06425	0.0225	16.96736	0.0235	15.93497	0.0245	14.96045
0.0206	19.11292	0.0216	17.95145	0.0226	16.86134	0.0236	15.83501	0.0246	14.86596
0.0207	18.99332	0.0217	17.83936	0.0227	16.75595	0.0237	15.73563	0.0247	14.77198
0.0208	18.87451	0.0218	17.72797	0.0228	16.6512	0.0238	15.63681	0.0248	14.67851
0.0209	18.75648	0.0219	17.61729	0.0229	16.54706	0.0239	15.53856	0.0249	14.58555
0.021	18.63923	0.022	17.50729	0.023	16.44354	0.024	15.44085	0.025	14.49308

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0251	14.40112	0.0261	13.50766	0.0271	12.65878	0.0281	11.85047	0.0291	11.07924
0.0252	14.30964	0.0262	13.42084	0.0272	12.57618	0.0282	11.77174	0.0292	11.00403
0.0253	14.21865	0.0263	13.33445	0.0273	12.49399	0.0283	11.69337	0.0293	10.92916
0.0254	14.12814	0.0264	13.24851	0.0274	12.41219	0.0284	11.61536	0.0294	10.85463
0.0255	14.03811	0.0265	13.16299	0.0275	12.33079	0.0285	11.53771	0.0295	10.78042
0.0256	13.94855	0.0266	13.07791	0.0276	12.24978	0.0286	11.46043	0.0296	10.70654
0.0257	13.85945	0.0267	12.99325	0.0277	12.16916	0.0287	11.38349	0.0297	10.63299
0.0258	13.77082	0.0268	12.90901	0.0278	12.08892	0.0288	11.30691	0.0298	10.55976
0.0259	13.68265	0.0269	12.82519	0.0279	12.00906	0.0289	11.23068	0.0299	10.48685
0.026	13.59493	0.027	12.74178	0.028	11.92958	0.029	11.15479	0.03	10.41426

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0301	10.34198	0.0311	9.635933	0.0321	8.958646	0.0331	8.307922	0.0341	7.681786
0.0302	10.27001	0.0312	9.566948	0.0322	8.892413	0.0332	8.244233	0.0342	7.620455
0.0303	10.19836	0.0313	9.498249	0.0323	8.826443	0.0333	8.180787	0.0343	7.55935
0.0304	10.127	0.0314	9.429832	0.0324	8.760734	0.0334	8.117584	0.0344	7.49847
0.0305	10.05596	0.0315	9.361696	0.0325	8.695284	0.0335	8.054621	0.0345	7.437812
0.0306	9.985214	0.0316	9.293838	0.0326	8.630092	0.0336	7.991896	0.0346	7.377376
0.0307	9.914769	0.0317	9.226256	0.0327	8.565155	0.0337	7.929408	0.0347	7.31716
0.0308	9.844621	0.0318	9.158948	0.0328	8.500472	0.0338	7.867155	0.0348	7.257162
0.0309	9.774767	0.0319	9.091912	0.0329	8.43604	0.0339	7.805134	0.0349	7.19738
0.031	9.705205	0.032	9.025145	0.033	8.371857	0.034	7.743346	0.035	7.137813

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0351	7.07846	0.0361	6.496335	0.0371	5.933952	0.0381	5.389985	0.0391	4.863223
0.0352	7.019319	0.0362	6.439231	0.0372	5.878748	0.0382	5.336554	0.0392	4.81145
0.0353	6.960388	0.0363	6.382323	0.0373	5.823726	0.0383	5.283293	0.0393	4.759836
0.0354	6.901666	0.0364	6.32561	0.0374	5.768886	0.0384	5.230202	0.0394	4.708381
0.0355	6.843151	0.0365	6.26909	0.0375	5.714226	0.0385	5.17728	0.0395	4.657084
0.0356	6.784842	0.0366	6.212762	0.0376	5.659745	0.0386	5.124525	0.0396	4.605944
0.0357	6.726737	0.0367	6.156624	0.0377	5.605442	0.0387	5.071936	0.0397	4.55496
0.0358	6.668836	0.0368	6.100675	0.0378	5.551316	0.0388	5.019513	0.0398	4.50413
0.0359	6.611136	0.0369	6.044915	0.0379	5.497365	0.0389	4.967253	0.0399	4.453454
0.036	6.553636	0.037	5.989341	0.038	5.443588	0.039	4.915157	0.04	4.402931

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0401	4.352559	0.0411	3.856979	0.0421	3.375549	0.0431	2.907411	0.0441	2.45177
0.0402	4.302338	0.0412	3.808214	0.0422	3.328151	0.0432	2.861297	0.0442	2.406864
0.0403	4.252267	0.0413	3.75959	0.0423	3.280885	0.0433	2.815307	0.0443	2.362076
0.0404	4.202345	0.0414	3.711105	0.0424	3.23375	0.0434	2.769441	0.0444	2.317403
0.0405	4.152571	0.0415	3.662759	0.0425	3.186745	0.0435	2.723697	0.0445	2.272846
0.0406	4.102944	0.0416	3.614551	0.0426	3.13987	0.0436	2.678075	0.0446	2.228404
0.0407	4.053463	0.0417	3.56648	0.0427	3.093124	0.0437	2.632575	0.0447	2.184076
0.0408	4.004126	0.0418	3.518545	0.0428	3.046505	0.0438	2.587195	0.0448	2.139861
0.0409	3.954934	0.0419	3.470746	0.0429	3.000014	0.0439	2.541934	0.0449	2.09576
0.041	3.905885	0.042	3.423081	0.043	2.95365	0.044	2.496793	0.045	2.05177

BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt	BER	Nopt
0.0451	2.007892	0.0461	1.575095	0.0471	1.152746	0.0481	0.740255	0.0491	0.337071
0.0452	1.964124	0.0462	1.5324	0.0472	1.111063	0.0482	0.699527	0.0492	0.297245
0.0453	1.920467	0.0463	1.489809	0.0473	1.069477	0.0483	0.658891	0.0493	0.257505
0.0454	1.876919	0.0464	1.447321	0.0474	1.027989	0.0484	0.618347	0.0494	0.217853
0.0455	1.83348	0.0465	1.404935	0.0475	0.986598	0.0485	0.577894	0.0495	0.178287
0.0456	1.790149	0.0466	1.362652	0.0476	0.945303	0.0486	0.537533	0.0496	0.138807
0.0457	1.746925	0.0467	1.32047	0.0477	0.904104	0.0487	0.497262	0.0497	0.099413
0.0458	1.703809	0.0468	1.278389	0.0478	0.863001	0.0488	0.45708	0.0498	0.060103
0.0459	1.660799	0.0469	1.236409	0.0479	0.821991	0.0489	0.416989	0.0499	0.020878
0.046	1.617894	0.047	1.194528	0.048	0.781076	0.049	0.376986		

Appendix B 2Q Algorithm Main Code

```
Imports PCComm
Public Class frmMain
    Private comm As New CommManager()
    Private transType As String = String.Empty

    Private Sub cboPort_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles cboPort.SelectedIndexChanged
        comm.PortName = cboPort.Text()
    End Sub

    ''' <summary>
    ''' Method to initialize serial port
    ''' values to standard defaults
    ''' </summary>
    Private Sub SetDefaults()
        'cboPort.SelectedIndex = 0
        cboBaud.SelectedText = "9600"
        cboParity.SelectedIndex = 0
        cboStop.SelectedIndex = 1
        cboData.SelectedIndex = 1
    End Sub

    ''' <summary>
    ''' methos to load our serial
    ''' port option values
    ''' </summary>
    Private Sub LoadValues()
        comm.SetPortNameValues(cboPort)
        comm.SetParityValues(cboParity)
        comm.SetStopBitValues(cboStop)
    End Sub

    ''' <summary>
    ''' method to set the state of controls
    ''' when the form first loads
    ''' </summary>
    Private Sub SetControlState()
        rdoText.Checked = True
        cmdSend.Enabled = False
        cmdClose.Enabled = False
    End Sub

    Private Sub frmMain_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.
    Load
        LoadValues()
        SetDefaults()
        SetControlState()
    End Sub

    Private Sub cmdClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdClose
    .Click
        comm.ClosePort()
        SetControlState()
        SetDefaults()
    End Sub

    Private Sub cmdOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdOpen.
    Click
        comm.Parity = cboParity.Text
        comm.StopBits = cboStop.Text
        comm.DataBits = cboData.Text
        comm.BaudRate = cboBaud.Text
        comm.DisplayWindow = rtbDisplay
        comm.DisplayReceive = rtbReceive
        comm.DisplaySend = rtbSend
        comm._nodeStatus = txtNodeStatus
```

```
    comm._status = txtStatus
    comm.OpenPort()
    tmrGoble.Start()
    cmdOpen.Enabled = False
    cmdClose.Enabled = True
    cmdSend.Enabled = True
End Sub

Private Sub cmdSend_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdSend.
Click
    comm.Message = txtSend.Text
    comm.Type = CommManager.MessageType.Normal

    'rhino part
    'Establish ACK
    'Sending RTC

    sendData = txtSend.Text()
    nodeStatus = "ACK Request"

    'comm.WriteData(txtSend.Text)
    'txtSend.Text = String.Empty
    'txtSend.Focus()
End Sub

Private Sub rdoHex_CheckedChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
rdoHex.CheckedChanged
    If rdoHex.Checked() Then
        comm.CurrentTransmissionType = PCComm.CommManager.TransmissionType.Hex
    Else
        comm.CurrentTransmissionType = PCComm.CommManager.TransmissionType.Text
    End If
End Sub

Private Sub cboBaud_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cboBaud.SelectedIndexChanged
    comm.BaudRate = cboBaud.Text()
End Sub

Private Sub cboParity_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cboParity.SelectedIndexChanged
    comm.Parity = cboParity.Text()
End Sub

Private Sub cboStop_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cboStop.SelectedIndexChanged
    comm.StopBits = cboStop.Text()
End Sub

Private Sub cboData_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cboData.SelectedIndexChanged
    comm.StopBits = cboStop.Text()
End Sub

Private Sub btnSetNode_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnSetNode.Click
    nodeAddress = txtNodeAddress.Text()
    receiverAddress = txtReceiverAddress.Text()
    'Call packACKFormat()
    ' Dim tem As String
    'tem = ACKFormat
    'tem = tem.Replace("%", "")
    ' MsgBox(ACKFormat)
    ' MsgBox(tem)
    timeCounter = 0
    tmrACK.Start()
```

```
End Sub
Sub packACKFormat()
    ACKFormat = "%RTS%" + nodeAddress + "%"
End Sub


Private Sub tmrACK_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrACK.Tick

    timeCounter = timeCounter + 1
    ACKCount = timeCounter
    pgbACK.Value = timeCounter * 20
    'If CTSStatus = True Then
    'tmrACK.Stop()
    'tmrACK.Enabled = False
    'comm.WriteData(txtSend.Text)
    'txtSend.Text = String.Empty
    'txtSend.Focus()
    ' End If

    If nodeStatus <> "CTS Received" Then
        If timeCounter <> ACKNumber Then
            txtNodeStatus.Text = "Resend ACK" + timeCounter.ToString()
            nodeStatus = "Resend ACK"
        Else
            nodeStatus = "Fail ACK"
            txtStatus.Text = "Idle"
            txtNodeStatus.Text = nodeStatus
            tmrACK.Stop()
            tmrACK.Enabled = False
        End If
    Else
        timeCounter = 0
        tmrACK.Stop()
        tmrACK.Enabled = False
    End If



End Sub

Private Sub btnSetACK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnSetACK.Click
    ACKNumber = txtACKNumber.Text
    ACKTimeout = txtACKTimeOut.Text
End Sub

Private Sub tmrGoble_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrGoble.Tick
    If nodeStatus = "RTS Received" Then
        'MsgBox("MAN")
        nodeStatus = "CTS Transmit"
        comm.WriteData("$$")
    End If

    If nodeStatus = "CTS Received" Then
        'MsgBox("MAN")
        timeCounter = 0
        tmrACK.Stop()
        tmrACK.Enabled = False
        'nodeStatus = "CTS Transmit"
        nodeStatus = "Transmit Data"
        comm.WriteData(sendData + "&")
        sendData = ""
    End If
```

```
    If nodeStatus = "Transmiting Data" Then
        tmrTransmit.Interval = 5 * 1000
        tmrTransmit.Enabled = True
        tmrTransmit.Start()
    End If


    If nodeStatus = "ACK Request" Then
        txtStatus.Text = nodeStatus
        tmrACK.Enabled = True
        tmrACK.Start()
        comm.WriteData("%%")
    End If

    If nodeStatus = "Fail ACK" Then
        nodeStatus = "Waiting to S/R"
        txtStatus.Text = "Idle"

    End If
    If nodeStatus = "Resend ACK" Then
        comm.WriteData("%%")
    End If
End Sub

Private Sub tmrTransmit_Tick(ByVal sender As Object, ByVal e As System.EventArgs) Handles tmrTransmit.
Tick
    nodeStatus = "Idle"
    txtNodeStatus.Text = "Waiting to S/R"
    txtStatus.Text = "Idle"
    tmrTransmit.Stop()
    tmrTransmit.Enabled = False
End Sub

Private Sub GroupPanel2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
GroupPanel2.Click

End Sub

Private Sub txtSend_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
txtSend.TextChanged

End Sub

Private Sub AboutToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles AboutToolStripMenuItem1.Click
    AboutBox1.ShowDialog()
End Sub

Private Sub ExitToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ExitToolStripMenuItem.Click
    Me.Hide()
End Sub

Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear
.Click

    receiveData = ""
    nodeAddress = ""
    receiverAddress = ""
    ACKTimeout = 100
    nodeStatus = "Waiting to S/R"
    ACKCount = 0
    dataStatus = ""
    ACKFormat = ""
    timeCounter = ""
    CTSStatus = ""
```

```
    End Sub

    Private Sub btnReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnReset
    .Click

        receiveData = ""
        nodeAddress = ""
        receiverAddress = ""
        ACKTimeout = 100
        nodeStatus = "Waiting to S/R"
        ACKCount = 0
        dataStatus = ""
        ACKFormat = ""
        timeCounter = 0
        CTSStatus = False
    End Sub

    Private Sub OptimizationToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.
    EventArgs) Handles OptimizationToolStripMenuItem.Click
        frmOptimaztion.ShowDialog()
    End Sub
End Class
```

```
Public Class frmOptimaztion

    'Public variable for calcuation
    Dim ber As Double
    Dim distance As Double
    Dim rate As Double
    Dim nopt1 As Double
    Dim nopt2 As Double
    Dim nopt3 As Double
    Dim header As Double = 160
    Dim en_mu As Double
    Dim packetsize As Double
    Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalc. ↙
    Click
        Call getTextboxValue()
        Call calc_packet()
    End Sub
    Sub getTextboxValue()
        ber = CDbl(txtBER.Text)
        distance = CDbl(txtDistance.Text)
        rate = CDbl(txtRate.Text)
    End Sub
    Sub calc_Nopt_BER()
        nopt1 = (-1) * header * Math.Log(1 - ber) - Math.Sqrt(((-4) * header * Math.Log(1 - ber)) + (header ↙
     * header * Math.Log(1 - (ber * ber))))
        nopt1 = nopt1 / (2 * Math.Log(1 - ber))

        'update textbox'

        Dim str As String

        str = "==================================" + vbCrLf + "Nopt1 [BER]                          = " + ↙
     nopt1.ToString + vbCrLf
        txtDetail.AppendText(str)

    End Sub
    Sub calc_Nopt_throughput()
        Dim mu As Double
        mu = header + ((2 / (1 * 1500)) * distance * rate)
        nopt2 = (mu / 2) * (Math.Sqrt(1 + 4 / (mu * ber)) - 1)

        'update textbox'

        Dim str As String

        'str = "Throughput Nopt               = " + nopt2.ToString + vbCrLf
        str = "Nopt2 [Throughput]              = " + nopt2.ToString + vbCrLf
        txtDetail.AppendText(str)

    End Sub
    Sub calc_Nopt_energy()
        nopt3 = (Math.Sqrt((header * header) - (4 * header / Math.Log(1 - ber))) - header) / 2
    End Sub
    'Sub calc_energy_efficient(ByVal lopt As Double)
    '    en_mu = lopt / (lopt + header) * Math.Pow(1 - ber, lopt + header)
    'End Sub
    Function calc_energy_efficient(ByVal lopt As Double) As Double
        en_mu = (lopt / (lopt + header)) * Math.Pow(1 - ber, lopt + header)
        Return en_mu
    End Function
    Sub calc_packet()

        Dim avg As Double
        Dim eng_eff As Double
        Dim eng_eff_opt As Double
        Dim different As Double
```

```
        Dim str As String
        Call calc_Nopt_BER()
        Call calc_Nopt_throughput()
        Call calc_Nopt_energy()

        avg = (nopt1 + nopt2) / 2
        'update textbox
        str = "Nopt3 [Average of Nop1 & Nopt2] = " + avg.ToString + vbCrLf
        txtDetail.AppendText(str)


        eng_eff = calc_energy_efficient(avg)
        eng_eff_opt = calc_energy_efficient(nopt3)

        different = (eng_eff_opt - eng_eff) / eng_eff_opt
        'udate textbox
        str = "The different between Nopt of energy and Nopt3 is                  = " + different.ToString↙
    + vbCrLf
        txtDetail.AppendText(str)



        If different <= 0.05 Then
            packetsize = avg
            'update textbox
            str = "Since it is less then 5%, final packet size is [Nopt3] =" + packetsize.ToString + vbCrLf↙
    + "================================== "
            txtDetail.AppendText(str)
        Else
            avg = (avg + nopt3) / 2
            'update textbox
            str = "Since it is more then 5%, final packet size is [Average (Nopt3,Nopt of energy)] = " +   ↙
    packetsize.ToString + vbCrLf + "================================== "
            txtDetail.AppendText(str)
        End If



    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClose↙
    .Click
        Me.Hide()
    End Sub
End Class
```

Appendix D: Optimal data packet size (N_{opt} in bits) under different BERs and different header length (h in bits)

N_{opt} / h / BER	h = 30	h = 40	h = 50	h = 60	h = 70	h = 80	h = 90	h = 100
$1x10^{-6}$	5462	6304	7045	7715	8331	8904	9441	9949
$1x10^{-5}$	1716	1979	2210	2419	2610	2788	2954	3111
$1x10^{-4}$	532	612	681	744	800	853	902	948
$1x10^{-3}$	157	178	197	213	227	239	251	262
$1x10^{-2}$	37	39	41	41	40	39	38	36
$1x10^{-1}$	20	20	18	16	13	9	5	1

Appendix E: The Table 5.1 format (on page 94) modified from Appendix D

Header = 30		Header = 40		Header = 50		Header = 60	
BER	N_{opt}	BER	N_{opt}	BER	N_{opt}	BER	N_{opt}
10^{-1}	20	10^{-1}	20	10^{-1}	18	10^{-1}	16
10^{-2}	37	10^{-2}	39	10^{-2}	41	10^{-2}	41
10^{-3}	157	10^{-3}	178	10^{-3}	197	10^{-3}	213
10^{-4}	532	10^{-4}	612	10^{-4}	681	10^{-4}	744
10^{-5}	1716	10^{-5}	1979	10^{-5}	2210	10^{-5}	2419
10^{-6}	5462	10^{-6}	6304	10^{-6}	7045	10^{-6}	7715

Header = 70		Header = 80		Header = 90		Header = 100	
BER	N_{opt}	BER	N_{opt}	BER	N_{opt}	BER	N_{opt}
10^{-1}	13	10^{-1}	9	10^{-1}	5	10^{-1}	1
10^{-2}	40	10^{-2}	39	10^{-2}	38	10^{-2}	36
10^{-3}	227	10^{-3}	239	10^{-3}	251	10^{-3}	262
10^{-4}	800	10^{-4}	853	10^{-4}	902	10^{-4}	948
10^{-5}	2610	10^{-5}	2788	10^{-5}	2954	10^{-5}	3111
10^{-6}	8331	10^{-6}	8904	10^{-6}	9441	10^{-6}	9949

Appendix H: Full table entries for Table 5.3 on page 96 showing the optimal packet size (N_{opt}) based on BERs and its energy efficiency (E_η) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries are based on the *range-rate* (*IR*) product of $1x10^5$ mbps

Header = 30					Header = 40				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	20	-	-	-
10^{-2}	37	0.35	0.34	59	10^{-2}	39	0.31	0.30	63
10^{-3}	157	0.71	0.68	265	10^{-3}	178	0.68	0.66	287
10^{-4}	532	0.90	0.88	934	10^{-4}	612	0.88	0.87	1,015
10^{-5}	1716	0.97	0.96	3,056	10^{-5}	1979	0.96	0.96	3,322
10^{-6}	5462	0.99	0.99	9,771	10^{-6}	6304	0.99	0.99	10,622

Header = 50					Header =60				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	18	-	-	-	10^{-1}	16	-	-	-
10^{-2}	41	0.27	0.27	66	10^{-2}	41	0.25	0.25	73
10^{-3}	197	0.65	0.63	307	10^{-3}	213	0.63	0.60	327
10^{-4}	681	0.87	0.86	1,090	10^{-4}	744	0.86	0.82	1,162
10^{-5}	2210	0.96	0.95	3,567	10^{-5}	2419	0.95	0.95	3,810
10^{-6}	7045	0.99	0.99	11,422	10^{-6}	7715	0.98	0.98	12,191

Header = 70					Header = 80				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	13	-	-	-	10^{-1}	9	-	-	-
10^{-2}	40	0.22	0.22	71	10^{-2}	39	0.21	0.21	73
10^{-3}	227	0.60	0.58	346	10^{-3}	239	0.59	0.57	363
10^{-4}	800	0.85	0.84	1,232	10^{-4}	853	0.84	0.83	1,298
10^{-5}	2610	0.95	0.95	4,040	10^{-5}	2788	0.95	0.94	4,264
10^{-6}	8331	0.98	0.98	12,925	10^{-6}	8904	0.98	0.98	13,642

Header = 90					Header =100				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	1	-	-	-
10^{-2}	37	0.19	0.19	74	10^{-2}	36	0.17	0.18	76
10^{-3}	251	0.57	0.56	380	10^{-3}	262	0.55	0.53	397
10^{-4}	902	0.83	0.83	1,179	10^{-4}	948	0.82	0.81	1,428
10^{-5}	2955	0.94	0.94	3,862	10^{-5}	3111	0.94	0.93	4,694
10^{-6}	9442	0.98	0.98	12,351	10^{-6}	9949	0.98	0.98	15,019

Appendix I: Full table entries showing the optimal packet size (N_{opt}) based on different BERs and its energy efficiency (E_η) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries are based on the *range-rate* (*IR*) product of $5x10^5$ mbps

Header = 30					Header = 40				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	20	-	-	-
10^{-2}	37	0.35	0.34	65	10^{-2}	39	0.31	0.30	66
10^{-3}	157	0.71	0.64	365	10^{-3}	178	0.68	0.61	379
10^{-4}	532	0.90	0.85	1,453	10^{-4}	612	0.88	0.84	1,510
10^{-5}	1716	0.97	0.95	4,958	10^{-5}	1979	0.96	0.94	5,149
10^{-6}	5462	0.99	0.97	16,066	10^{-6}	6304	0.99	0.97	16,642

Header = 50					Header =60				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	18	-	-	-	10^{-1}	16	-	-	-
10^{-2}	41	0.27	0.27	68	10^{-2}	41	0.25	0.25	68
10^{-3}	197	0.65	0.60	393	10^{-3}	213	0.63	0.58	405
10^{-4}	681	0.87	0.83	1,562	10^{-4}	744	0.86	0.82	1,608
10^{-5}	2210	0.96	0.94	5,323	10^{-5}	2419	0.95	0.94	5,486
10^{-6}	7045	0.99	0.98	17,207	10^{-6}	7715	0.98	0.98	17,767

Header = 70					Header = 80				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	13	-	-	-	10^{-1}	9	-	-	-
10^{-2}	40	0.22	0.22	68	10^{-2}	39	0.21	0.21	68
10^{-3}	227	0.60	0.56	415	10^{-3}	239	0.59	0.54	426
10^{-4}	800	0.85	0.81	1,653	10^{-4}	853	0.84	0.81	1,699
10^{-5}	2610	0.95	0.93	5,640	10^{-5}	2788	0.95	0.93	5,788
10^{-6}	8331	0.98	0.98	18,265	10^{-6}	8904	0.98	0.98	18,712

Header = 90					Header =100				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	1	-	-	-
10^{-2}	37	0.19	0.19	68	10^{-2}	36	0.17	0.18	69
10^{-3}	251	0.57	0.53	436	10^{-3}	262	0.55	0.52	446
10^{-4}	902	0.83	0.80	1,740	10^{-4}	948	0.82	0.79	1,780
10^{-5}	2955	0.94	0.93	5,930	10^{-5}	3111	0.94	0.93	6,067
10^{-6}	9442	0.98	0.98	19,171	10^{-6}	9949	0.98	0.98	1,9614

Appendix J: Full table entries showing the optimal packet size (N_{opt}) based on different BERs and its energy efficiency (E_η) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries arebasedon the *range-rate* (IR) product of $9x10^5$ mbps.

Header = 30					Header = 40				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	20	-	-	-
10^{-2}	37	0.35	0.32	66	10^{-2}	39	0.31	0.29	67
10^{-3}	157	0.71	0.61	411	10^{-3}	178	0.68	0.60	424
10^{-4}	532	0.90	0.83	1,760	10^{-4}	612	0.88	0.82	1,812
10^{-5}	1716	0.97	0.94	6,173	10^{-5}	1979	0.96	0.93	6,350
10^{-6}	5462	0.99	0.98	20,176	10^{-6}	6304	0.99	0.98	20,742

Header = 50					Header =60				
BER	N_{opt}	E_η	E_{2Q}	N_{2Q}	BER	N_{opt}	E_η	E_{2Q}	N_{2Q}
10^{-1}	18	-	-	-	10^{-1}	16	-	-	-
10^{-2}	41	0.27	0.26	66	10^{-2}	41	0.25	0.24	69
10^{-3}	197	0.65	0.58	436	10^{-3}	213	0.63	0.56	447
10^{-4}	681	0.87	0.81	1,895	10^{-4}	744	0.86	0.80	1,902
10^{-5}	2210	0.96	0.93	6,505	10^{-5}	2419	0.95	0.93	6,654
10^{-6}	7045	0.99	0.98	21,252	10^{-6}	7715	0.98	0.98	21,727

Header = 70					Header = 80				
BER	N_{opt}	E_{η}	E_{2Q}	N_{2Q}	BER	N_{opt}	E_{η}	E_{2Q}	N_{2Q}
10^{-1}	13	-	-	-	10^{-1}	19	-	-	-
10^{-2}	40	0.22	0.22	68	10^{-2}	39	0.21	0.21	68
10^{-3}	227	0.60	0.53	456	10^{-3}	239	0.59	0.53	465
10^{-4}	800	0.85	0.79	1,942	10^{-4}	853	0.84	0.79	1,981
10^{-5}	2610	0.95	0.92	6,790	10^{-5}	2788	0.95	0.92	6,924
10^{-6}	8331	0.98	0.98	22,180	10^{-6}	8904	0.98	0.98	22,607

Header = 90					Header =100				
BER	N_{opt}	E_{η}	E_{2Q}	N_{2Q}	BER	N_{opt}	E_{η}	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	10^{-1}	10	-	-	-
10^{-2}	37	0.19	0.19	68	10^{-2}	36	0.17	0.18	68
10^{-3}	251	0.57	0.52	474	10^{-3}	262	0.55	0.51	482
10^{-4}	902	0.83	0.78	2,018	10^{-4}	948	0.82	0.78	2,053
10^{-5}	2955	0.94	0.92	7,052	10^{-5}	3111	0.94	0.92	7,170
10^{-6}	9442	0.98	0.98	23,016	10^{-6}	9949	0.98	0.98	23,414

Appendix M: Full table entries for Table 5.5 (on page 102) showing the optimal packet size (N_{opt}) based on different lR products and its energy efficiency (E_{lR}) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries are based on the header length of 80 bits.

$p = 10^{-2}$					$p = 10^{-3}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}	lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	106	0.19	0.21	73	1	487	0.52	0.56	363
2	100	0.18	0.21	70	2	520	0.51	0.56	380
3	98	0.18	0.21	68	3	556	0.50	0.55	398
4	98	0.18	0.21	68	4	587	0.48	0.55	413
5	98	0.18	0.21	68	5	614	0.47	0.54	427
6	98	0.18	0.21	68	6	637	0.46	0.54	438
7	98	0.18	0.21	68	7	657	0.46	0.53	448
8	98	0.18	0.21	68	8	676	0.45	0.53	458
9	98	0.18	0.21	68	9	692	0.44	0.52	466

$p = 10^{-4}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	1,744	0.80	0.83	1,298
2	1,958	0.79	0.82	1,405
3	2,174	0.77	0.81	1,513
4	2,369	0.76	0.81	1,611
5	2,545	0.75	0.80	1699
6	2,705	0.74	0.80	1,779
7	2,847	0.73	0.79	1,850
8	2,987	0.72	0.79	1,920
9	3,110	0.71	0.78	1,981

$p = 10^{-5}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	5,739	0.93	0.94	4,263
2	6,547	0.93	0.94	4,667
3	7,360	0.92	0.93	5,074
4	8,105	0.91	0.93	5,446
5	8,788	0.91	0.93	5,788
6	9,419	0.90	0.92	6,103
7	10,010	0.89	0.92	6,399
8	10,560	0.88	0.92	6,674
9	11,080	0.87	0.92	6,934

$p = 10^{-6}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	18,380	0.98	0.98	13,640
2	21,070	0.98	0.98	14,987
3	23,780	0.98	0.98	16,342
4	26,280	0.98	0.98	17,592
5	28,580	0.98	0.98	18,742
6	30,710	0.98	0.98	19,807
7	32,700	0.98	0.98	20,802
8	34,580	0.98	0.98	21,742
9	36,310	0.98	0.98	22,607

Appendix N: Full table entries showing the optimal packet size (N_{opt}) based on different IR products and its energy efficiency (E_{IR}) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries are based on the header length of 30 bits.

.

$p = 10^{-2}$					$p = 10^{-3}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}	lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	81	0.30	0.34	59	1	373	0.63	0.68	266
2	86	0.29	0.34	62	2	445	0.60	0.67	302
3	87	0.29	0.34	62	3	498	0.57	0.65	328
4	91	0.28	0.34	64	4	540	0.55	0.64	349
5	92	0.28	0.34	65	5	573	0.53	0.64	366
6	93	0.28	0.34	65	6	601	0.52	0.63	380
7	94	0.27	0.32	66	7	625	0.51	0.63	392
8	94	0.27	0.32	66	8	646	0.50	0.62	402
9	95	0.27	0.32	66	9	665	0.48	0.61	412

$p = 10^{-4}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	1,336	0.86	0.88	935
2	1,676	0.83	0.87	1,105
3	1,948	0.81	0.86	1,240
4	2,176	0.79	0.85	1,355
5	2,369	0.78	0.85	1,451
6	2,551	0.77	0.84	1,542
7	2,707	0.75	0.83	1,620
8	2,853	0.74	0.83	1,693
9	2,988	0.74	0.82	1,761

$p = 10^{-5}$					$p = 10^{-6}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}	lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	4,396	0.95	0.96	3,057	1	14,080	0.98	0.98	9,771
2	5,605	0.94	0.96	3,661	2	18,040	0.98	0.98	11,751
3	6,595	0.93	0.95	4,156	3	21,310	0.98	0.98	13,386
4	7,446	0.92	0.95	4,582	4	24,140	0.98	0.98	14,801
5	8,201	0.92	0.95	4,959	5	26,670	0.98	0.98	16,066
6	8,866	0.91	0.94	5,292	6	28,910	0.98	0.98	17,186
7	9,496	0.90	0.94	5,607	7	31,040	0.98	0.98	18,251
8	10,100	0.90	0.94	5,909	8	33,020	0.98	0.98	19,241
9	10,630	0.89	0.93	6,174	9	34,940	0.98	0.98	20,201

Appendix O: Full table entries showing the optimal packet size (N_{opt}) based on different lR products and its energy efficiency (E_{lR}) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm. The table entries are based on the header length of 100 bits.

$p = 10^{-2}$					$p = 10^{-3}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}	lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	116	0.16	0.18	76	1	533	0.49	0.53	398
2	106	0.16	0.18	71	2	550	0.48	0.53	406
3	103	0.17	0.18	70	3	579	0.47	0.53	421
4	102	0.17	0.18	69	4	606	0.46	0.52	434
5	101	0.17	0.18	69	5	630	0.46	0.51	446
6	101	0.17	0.18	69	6	652	0.44	0.51	457
7	100	0.17	0.18	68	7	671	0.44	0.51	467
8	100	0.17	0.18	68	8	688	0.43	0.51	475
9	100	0.17	0.18	68	9	703	0.43	0.50	483

$p = 10^{-4}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	1,908	0.78	0.81	1,429
2	2,071	0.77	0.81	1,510
3	2,264	0.76	0.80	1,607
4	2,446	0.75	0.80	1,698
5	2,613	0.74	0.79	1,781
6	2,762	0.73	0.79	1,856
7	2,904	0.72	0.78	1,927
8	3,035	0.71	0.78	1,992
9	3,158	0.70	0.77	2,054

$p = 10^{-5}$					$p = 10^{-6}$				
lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}	lR (x10^5)	N_{opt}	E_{lR}	E_{2Q}	N_{2Q}
1	6,276	0.92	0.93	4,694	1	20,090	0.98	0.98	15,020
2	6,924	0.92	0.93	5,018	2	22,290	0.98	0.98	16,120
3	7,666	0.91	0.93	5,389	3	24,770	0.98	0.98	17,360
4	8,369	0.91	0.93	5,741	4	27,070	0.98	0.98	18,510
5	9,023	0.90	0.92	6,068	5	29,340	0.98	0.98	19,645
6	9,632	0.89	0.92	6,372	6	31,410	0.98	0.98	20,680
7	10,190	0.88	0.92	6,651	7	33,350	0.98	0.98	21,650
8	10,720	0.87	0.91	6,916	8	35,180	0.98	0.98	22,565
9	11,230	0.86	0.91	7,171	9	36,930	0.98	0.98	23,440

Appendix P: Full table entries based on the format of Table 5.6 (on page 107) showing the optimal packet size (N_{2M}) qualified by BERs and Throughput Efficiency (related to IR product) metrics and the related energy efficiency (E_{2M}) as compared to energy efficiency (E_{2Q}) and optimal packet size (N_{2Q}) from the proposed algorithm.

(a) For near field data transmission:

Header (h) = 30 bits						
lR product = $1x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-	-	-
10^{-2}	37	82	60	0.34	0.35	49
10^{-3}	157	373	265	0.68	0.70	211
10^{-4}	532	1,336	934	0.88	0.89	733
10^{-5}	1,716	4,396	3,056	0.96	0.96	2,386
10^{-6}	5,462	14,080	9,771	0.98	0.99	7617

Header (h) = 60 bits						
lR product = $1x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	16	-	-	-	-	-
10^{-2}	41	96	69	0.23	0.25	55
10^{-3}	213	442	328	0.60	0.62	271
10^{-4}	744	1,581	1,163	0.85	0.85	954
10^{-5}	2,419	5,202	3,810	0.95	0.95	3,115
10^{-6}	7,715	16,660	12,188	0.98	0.98	9,952

Header (h) = 100 bits						
lR product = $1x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	1	-	-	-	-	-
10^{-2}	36	116	76	0.18	0.19	56
10^{-3}	262	533	398	0.55	0.55	330
10^{-4}	948	1,908	1,428	0.81	0.82	1,188
10^{-5}	3,111	6,276	4,694	0.93	0.94	3,903
10^{-6}	9,949	20,090	15,020	0.98	0.98	12,485

Note: N_{2M} = Average of N_p and N_{lR}

(b) For Far field data transmission:

Header (h) = 30 bits						
lR product = $9x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	20	-	-	-		
10^{-2}	37	95	66	0.32	0.35	52
10^{-3}	157	665	411	0.61	0.67	284
10^{-4}	532	2,988	1,760	0.82	0.86	1,146
10^{-5}	1,716	10,630	6,173	0.94	0.95	3,945
10^{-6}	5,462	34,890	20,176	0.97	0.97	12,819

Header (h) = 60 bits						
lR product = $9x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	16	-	-	-	-	-
10^{-2}	41	97	69	0.24	0.25	55
10^{-3}	213	681	447	0.55	0.6	330
10^{-4}	744	3,061	1,903	0.80	0.84	1,324
10^{-5}	2,419	10,890	6,655	0.93	0.94	4,537
10^{-6}	7,715	35,740	21,728	0.98	0.98	14,722

Header (h) = 100 bits						
lR product = $9x10^5$ (m-bps)						
BER	N_p	N_{lR}	N_{2M}	E_{2M}	E_{2Q}	N_{2Q}
10^{-1}	1	-	-	-	-	-
10^{-2}	36	100	68	0.19	0.19	52
10^{-3}	262	703	483	0.50	0.54	373
10^{-4}	948	3,160	2,054	0.78	0.81	1,501
10^{-5}	3,111	11,240	7,176	0.92	0.93	5,144
10^{-6}	9,949	36,880	23,415	0.98	0.98	16,682

Note: N_{2M} = Average of N_p and N_{lR}