

# **RELIABLE ENERGY BALANCE MODEL FOR WIRELESS SENSOR NETWORKS**

A project Report submitted in partial fulfillment of Degree of Bachelor  
Of Technology

*Submitted By*

**Subhash Chandra (13CS50)**

*Under Supervision of*

**Dr. Ravendra Singh**



**Department of Computer Science and Information Technology**

**Faculty of Engineering & Technology**

**MJP Rohilkhand University, Bareilly (U.P.)**

**2017**

## STUDENT'S DECLARATION

I **Subhash Chandra**, S/o Mr. Radhey Shyam Verma, student of B.Tech. (CSIT) 4th year (2016-17) hereby declare that the project entitled **RELIABLE ENERGY BALANCE MODEL FOR WIRELESS SENSOR NETWORKS** is my authentic work done carried out in partial fulfillment of the requirement for the award of degree of **Bachelor of Technology in Computer Science and Information Technology** under the supervision of **Dr. Ravendra Singh**, Project guide, **Department of Computer Science and Information Technology, Institute of Engineering and Technology, M.J.P. Rohilkhand University, Bareilly.**

NAME

SIGNATURE

**Subhash Chandra**

**(Roll No. : 13CS50)**

.....



# INSTITUTE OF ENGINEERING AND TECHNOLOGY

## M. J. P. ROHILKHAND UNIVERSITY, BAREILLY

Ref.:MJPRU/FET/CSIT/Project/2016-17/Gr.

No-

Date: . . . . .

### CERTIFICATE

This is to certify that **Subhash Chandra (Roll No. 13CS50)**, has carried out the project work, presented in this report entitled **“RELIABLE ENERGY BALANCE MODEL FOR WIRELESS SENSOR NETWORKS”** submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology in Computer Science and Information Technology, Institute of Engineering and Technology, M.J.P. Rohilkhand University, Bareilly (U.P)** during the academic year 2016-2017.

The project work carried out by him is satisfactory. The report embodies result of the original work and studies carried out by them and the contents of the report do not form the basis for the award of any other degree to the candidate.

**Dr. Ravendra Singh**

(Project Guide)

## ABSTRACT

The work is focused on enhanced energy balance model for lifetime maximization for systematically/randomly distributed sensor network. The lifetime of a sensor network depends on the rate of energy depletion caused by multiple factors, such as load imbalance, sensor deployment distribution, scheduling, transmission power control, and routing. Therefore, we have developed a mathematical model for analysis of load imbalance under uniform and accumulated data flow. Based on this analysis, we developed a model to rationalize energy distribution among the sensors for enhancing the lifetime of the network. To realize the energy balance model, three algorithm ó annulus formation, slice formation, connectivity ensured routing and coverage preserved scheduling have been introduced. The model has been simulated in Net Sim\_9.2 and results are compared with Energy Balanced Balance Model given by D.K. Lobiyal. Lifetime has been measured in term of the time duration for which the network provides satisfactory level of coverage and data delivery distribution among the sensors is lower than other models.

# ACKNOWLEDGEMENT

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if I did not have a support of many individuals and organizations. Therefore, we would like to extend our sincere gratitude to all of them. First of all, I am thankful to the Faculty Mentor **Dr. Ravendra Singh** for providing necessary guidance concerning projects implementation.

I would like to express my sincere thanks towards volunteer researchers who devoted their time and knowledge in the implementation of this project. Nevertheless, i express my gratitude toward my family and colleagues for their kind co-operation and encouragement which help me in completion of this project.

Subhash Chandra (13CS50)

---

# TABLE OF CONTENTS

Title	Page
<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1. Previous work í	2
<b>2. NetSim Introduction.....</b>	<b>4</b>
2.1. Introduction to modeling and simulation of networks .....	4
2.2. Versions of NetSim-Academic, Standard & Pro .....	5
2.3. Component in Pro and Standard versions .....	7
2.4. Getting Started in NetSim.....	9
2.4.1.1. Installing NetSim in Client.....	9
2.5. Setting up License Server.....	14
2.5.1. Installing NetSim License Server.....	14
2.5.2. Running RLM Dongle (Server).....	17
2.5.3. Running NetSim Software.....	18
<b>3. INTRODUCTION TO MATLAB.....</b>	<b>19</b>
3.1. What is MATLAB? .....	19
3.2. The MATLAB ENVIRONMENT.....	19
3.3. Key Features.....	19
3.4. Numeric Computation.....	20
3.5. Data Analysis and Visualization.....	20
3.6. Acquiring Data.....	21
3.7. Analyzing Data.....	21
3.8. Visualizing Data.....	21
3.9. Programming and Algorithm Development.....	21

---

3.9.1. The MATLAB Language.....	21
3.9.2. Development tools.....	22
3.9.3. Integration with other Language and Applications.....	22
3.10. Performance.....	22
3.11. Application Delopment and Deployment.....	23
3.12. Designing Graphical User Interfaces.....	23
3.13. Deploying Application.....	23
<b>4. Problem Description .....</b>	<b>24</b>
<b>5. Mathematical Analysis of Load on Sensors.....</b>	<b>28</b>
5.1. Load Analysis under uniform Data Flow.....	29
5.2. Theorem 1.....	30
5.3. Theorem 2.....	30
5.4. Theorem 3.....	31
5.5. Load Analysis under Accumulation Data Flow.....	32
5.6. Load Equalization.....	34
<b>6. Algorithms Implemented .....</b>	<b>36</b>
6.1. Annulus Formation Algorithm (AFA) .....	36
6.1.1. Algorithm 1.....	38
6.2. Connectivity Ensured Routing Algorithm.....	39
6.2.1. Algorithm 2.....	39
6.3. Coverage Preserved Scheduling Algorithm.....	40
6.3.1. Algorithm 3.....	41
6.4. Slice Formation Algorithm.....	42
6.4.1. Algorithm 4.....	42
<b>7. Performance Analysis.....</b>	<b>44</b>
7.1. Simulation Environment.....	44
7.2. Annulus Formation.....	44
7.3. Connectivity Measurement.....	45
7.4. Coverage Measurement.....	46
7.5. Distribution of Residual Energy.....	46

---

7.6. Lifetime Measurement.....	48
<b>8. Conclusion.....</b>	<b>49</b>
<b>REFERENCES.....</b>	<b>50</b>

---



# LIST OF FIGURES

Title	Page
Figure 1: Installation Step 1	.09
Figure 2: Installation Step 2	.09
Figure 3: Installation Step 3	.09
Figure 4: Installation Step 4	.10
Figure 5: Installation Step 5	.10
Figure 6: Installation Step 6	.11
Figure 7: Installation Step 7	.11
Figure 8: Installation Step 8	.11
Figure 9: Installation Step 9	.12
Figure 10: Installation Step 10	.12
Figure 11: Installation Step 11	.12
Figure 12: Installation Step 12	.13
Figure 13: Installation Step 13	.13
Figure 14: Installation Step 14	.13
Figure 15: Installation Step 15	.14
Figure 16: Installation Step 16	.15
Figure 17: Installation Step 17	.15
Figure 18: Installation Step 18	.15
Figure 19: Installation Step 19	.16
Figure 20: Installation Step 20	.16
Figure 21: Installation Step 21	.17
Figure 22: Installation Step 22	.18
Figure 23: Installation Step 23	.18
Figure 24: NetSim Home Page	.18
Figure 25: Load on Sensors	.28
Figure 26: Slice of the Region.	.29
Figure 27: Data Flow Ring	.33
Figure 28: Data Flow in Slice	.34

---

Figure 29: Data Accumulation graphí í ..í í í í í í í í í í í í ..í ..í í í 34  
Figure 30: Exponential Data Congestioní í í í í í í í í í í ..í í í í í í í í ..í .35  
Figure 31: Remaining Energy for EBMí í í í í í í í í í ..í í í í í í í í ..í .47  
Figure 32: Remaining Energy for REBMí í í í í í í í í í ..í í í í í í í í ..í .47

# LIST OF TABLES

Title	Page
1: Netsim Versions. í í í í í í í í í í í í í í í í .í í í í í í í í .í í	...05
2: Component in Pro and Standard Versionsí í í í í í í í í .í í í í í í í í	...07
3: Notation Table í .í í í í í í	26
3: Routing Tableí .í í í	..36

---

# 1. INTRODUCTION

Wireless sensor network is a set of small and intelligent entities which are responsible for their organization, and working in order to provide sensing services assigned to them. The services provided by a sensor network belong to broad spectrum of real time applications. Surveillance in battlefields and buildings for stopping possible intrusion and theft, monitoring vehicular traffic and for pest management in agricultural fields etc. are some of the vital application. In spite of its wide applicability, research in this area has its own challenges. One of the main challenges in WSN is to maximize the lifetime of the network by conserving energy and without compromising the coverage and connectivity in the realistic environments.

Energy management is a critical task in wireless sensor networks due to severely power limited sensors. The limitation of power is because of the two operating factors. First, sensors consume lots of energy while operating with full intensity which cannot be supported by a small battery for longer duration. Secondly, battery cannot be replenished in a sensor network due to inaccessibility of physical locations. In the literature, many solutions exist for energy balancing in the wireless sensor network. These solutions balance energy consumption among different sensors in the network while they are at work. However, performance of a sensor network primarily depends on the deployment scheme. But at the same time, these solutions also suffers from the inherent discrepancy in the deployment itself. There are former, sensors are deployed in pre calculated manner in the sensing region. In the latter, sensors are not deployed in so called well calculated manner. However, if situation is not very hostile, some estimation can be carried out before deployment. Different sensors bear different load in randomly deployed networks. Due to this factor sensors near the sink lose their energy earlier and thus network suffers from the problem of connectivity.

In energy balanced model for wireless sensor network, we assume that for a large wireless sensor networks in hostile environment sensors are deployed randomly. However, this random deployment can be realized by following normal distribution from its center to their boundaries. We analyze the energy consumption in transmission and sensing by all the sensors. We have used energy consumption and load interchangeably. Load analysis has been performed for uniform data flow and accumulated data flow. The work is intended to fairly equalize the energy consumption by all the sensor deployed in sensing region using adaptive sensing, adjusting transmission range and density control of sensors. To realize the reliable

energy balance model, supporting algorithms for annulus formation, slice formation, connectivity ensured routing and coverage preserved scheduling are developed. We have carried out extensive simulations to validate the applicability of reliable energy balance model.

The technique has also been validated through simulation for data monitoring and propagating task by applying the probabilistic data propagation algorithm with optimal parameters. Energy transmission policy has been proposed based on controlled transmission power. The analysis has been performed by taking concentric circles around the sink and load is balanced by decreasing the transmission range of different sensors near the sink.

D.K.Lobiyal have proposed a scheme to maximize the lifetime of the network based on the cooperation among different sensors [1]. This scheme considers information importance based communication for energy efficient data processing. It is proved analytically that choosing multiple paths for sensors to balance energy in the network. It is proved analytically that choosing multiple path for sensors forward data conserves energy to maximize the lifetime of a network. A scalable and distributed algorithm is presented for routing of data by Chang and Tassiulas. This routing technique is based on linear programming formulation used for choosing next hop. Additionally, they have also presented a routing scheme to balance the energy among sensors using mobile agents. Clustering algorithms are also made by which clusters lifetime can be increased.

Our motivation for this work was to address the problem of the energy imbalance inadequately. A few of researchers addressed this problem once the distribution of sensors had been assumed. This assumption possess lots of limitations due to uneven load. Therefore in this we have made an attempt to address the problem of energy imbalance by using adaptive sensing and transmission without compromising coverage and connectivity. This approach does not only fairly equalize the energy consumption by different sensors but also improve lifetime of network.

## **1.1 Previous Work**

In the literature, most of the research works on energy balancing models for wireless sensor networks deal with the planned networks. Therefore, researcher have not focused on energy balancing in these networks since such networks either do not require energy balancing necessarily or do not provide much opportunity for energy balancing. But, in randomly

deployed networks, energy balancing is one of the key requirement due to unpredicted topology. D.K.Lobiyal [1] have proposed a spreading technique to balance the energy model among sensors of the same region. Whole sensing region is divided in many annulus.

## 2. NetSim – Introduction

### 2.1 Introduction to modeling and simulation of networks

A network simulator enables users to virtually create a network along with its components such as devices, links, and applications etc. to study the behavior and performance of the Network.

Some examples of applications of network simulators are

- “ Protocol performance analysis
- “ Application modeling and analysis
- “ Network design and planning
- “ Research and development of new networking technologies
- “ Test and verification

The key features essential to any network simulation are -

- “ **Building the model** ó Create a network scenario with devices, links, applications etc
- “ **Running the simulation** - Run the discrete event simulation and log different performance metrics
- “ **Visualizing the simulation**- Use a packet animator to view the flow of packets
- “ **Analyzing the results** - Examine output performance metrics such as throughput, delay, loss etc. at multiple levels - network, sub network, link, queue, application etc.
- “ **Developing your own protocol / algorithm** - Extend existing algorithms by modifying the simulators source C code

## 2.2 Versions of NetSim – Academic, Standard & Pro

NetSim [2] is used by people from different areas such as academics, industry and defense to design, simulate, analyze and verify the performance of different networks.

NetSim comes in three versions- **Academic**, **Standard** and **Pro**. The academic version is used for lab experimentation and teaching. The standard version is used for project work and research while Pro version addresses the needs of defense and industry. The standard and pro versions are available as components in NetSim v9 from which users can choose and assemble. The academic version is available as a single product and includes all the technologies shown below. The main differences between the various versions are tabulated below:

Features	Academic	Standard	Pro
<b>Technology Coverage</b>			
Internetworks	✓	✓	✓
Legacy Networks and MPLS Networks	✓	✓	✗
BGP	✓	✓	✓
Advanced Wireless Networks	✓	✓	✓
Cellular Networks	✓	✓	✓
Wireless Sensor Networks	✓	✓	✓
Internet of Things	✓	✓	✓
Zigbee	✓	✓	✓
Cognitive Radio Networks	✓	✓	✓
LTE Networks	✓	✓	✓
Military Radio: TDMA-Link16	✗	✗	✓
<b>Basics</b>			
Understand networking concepts using more than 400 animations	✓	✗	✗
<b>Performance Reporting</b>			
Performance metrics available for Network and Sub-network	✓	✓	✓
<b>Packet Animator</b>			
Used to animate the packet flow in network	✓	✓	✓
<b>Packet Trace and Event Trace</b>			
Available in tab ordered .txt format for easy post processing	✗	✓	✓
<b>Protocol Library Source Codes with</b>			



<b>Documentation</b> Protocol C source codes and appropriate header files with extensive documentation	✗	✓	✓
<b>Wireshark Interface</b> Capture NetSim simulation packets using wire-shark	✗	✓	✓
<b>Integrated debugging</b> Users can write their own code, link their code to NetSim and debug using Visual Studio	✗	✓	✓
<b>Dynamic Metrics</b> Allows users to plot the value of a parameter over simulation time	✗	✓	✓
<b>Emulator (Add on)</b> Connect to real hardware running live application	✗	✓	✓
<b>Target Users and Pricing</b>	Educational	Educational	Commercial

Table 1:- NetSim Versions (Source : Netsim User Manual)

## 2.3 Components in Pro and Standard versions

In NetSim v9, users can choose and assemble components for Pro and Standard version. The components are as follows:

Component No	Networks / Protocols	International Standards
<b>Component 1 (Base. Required for all components)</b>	<p><b>Internetworks</b>            Ethernet - Fast &amp; Gigabit            Address Resolution Protocol            WLAN - 802.11 a, b, g, n, ac and e            Propagation ó Line of Sight, Log-normal            Shadowing, Rayleigh Fading            IP v4 with VPN Firewalls            Routing - RIP, OSPF            Queuing - Round Robin, FIFO, Priority            TCP, UDP</p> <p><b>Common Modules</b>            Applications (Traffic Generator): CBR,            Voice, Video, FTP, Database, HTTP,            Email, Peer-to-peer and Custom            Application encryption ó AES, DES            Virtual Network Stack            Simulation Kernel            Command Line Interface            Metrics Engine with packet trace, event trace            and dynamic metrics            Packet Animator</p>	<p>IEEE 802.3            RFC 826            802.11 a/b/g/n/ac/e</p> <p>RFC 2453,2328</p> <p>RFC's 793, 2001 and 768</p>
<b>Component 2</b>	<p><b>Legacy Networks</b>            Aloha - Pure &amp; Slotted            CSMA/CD            Token ring            Token bus            ATM            X.25            Frame Relay            Real Time (Frame Capture)            Multi-Protocol Label Switching (MPLS)</p>	<p>IEEE 802.3            IEEE 802.4            IEEE 802.5            ATM Forum            ITU Forum            IETF RFC 3031</p>
<b>Component 3</b>	<p><b>BGP Networks</b>            Border Gateway Protocol (BGP)</p>	<p>IETF RFC s 1771 &amp; 3121</p>

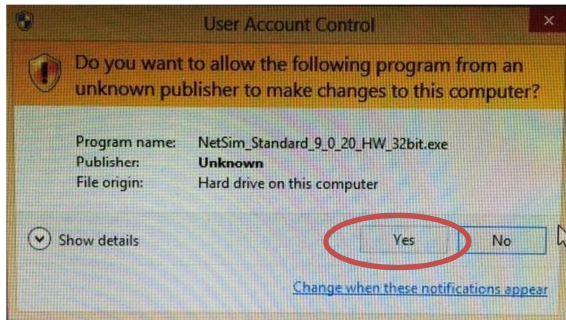
<b>Component 4</b>	<b>Advanced Wireless Networks</b> MANET - DSR, AODV, OLSR, ZRP Wi-Max	IETF RFC 4728, 3561, 3626 IEEE 802.16d
<b>Component 5</b>	<b>Cellular Networks</b> GSM CDMA	3GPP, ETSI, IMT-MC, IS95 A/B, IxRTT, 1x-EV-Do, 3xRTT
<b>Component 6</b> <b>(Component 4 required)</b>	<b>Wireless Sensor Networks, Internet of Things &amp; ZigBee</b> WSN with agent model & battery models ZigBee	IEEE 802.15.4 MAC , MANET in L3
<b>Component 7</b>	<b>Cognitive Radio Networks</b> WRAN	IEEE 802.22
<b>Component 8</b>	<b>Long Term Evolution</b> LTE	3GPP
<b>Component 9</b> <b>(Component 4 required)</b>	<b>Military Radio (Only in PRO Version)</b> TDMA Link 16	----

Table 2 :- Component in Pro and Standard Versions (Source : NetSim User Manual)

## 2.4 Getting Started in NetSim

### 2.4.1 Installing NetSim in Client

**Note:** Based on the NetSim version under installation the version type being displayed in the following windows will change. For example you will see NetSim Standard for a standard version install



Click on **YES** to install the software.

Figure 1: Installation step 1

Setup prepares the installation wizard and software installation begins with a **Welcome Screen**.

Click on **Next** button

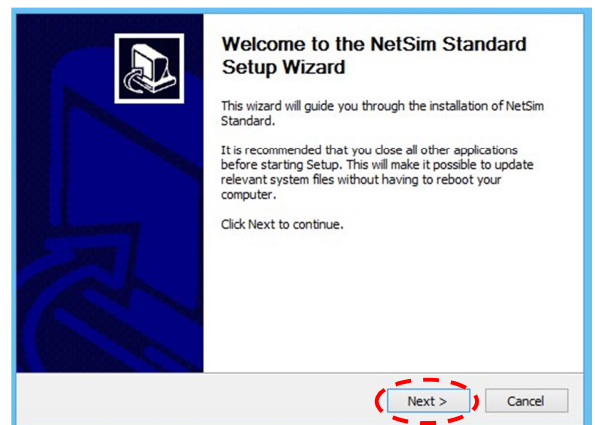
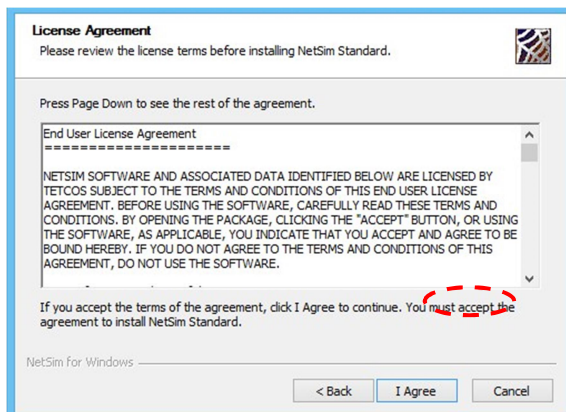


Figure 2: Installation step 2



In the next screen, License agreement will be displayed.

Figure 3: Installation step 3

Read the agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted click on **Agree** button else quit the setup by clicking **Cancel** button.

If you agree with the license agreement, you will be prompted to select the components to be installed. The list of components is available for selection and assembly only in the Standard and Pro version. Other versions of NetSim are available as a single package.

Click on the **Next** button.

**Note:** Select all the supporting applications for complete installation of the software.

In the next screen, you will be requested to enter the installation path

Select the path in which the software needs to be installed and click on **Next** button.

**Note:** In the case of 64 bit machine, ensure that the path is <OS installed drive>:/Program Files(x86)/NetSim Standard

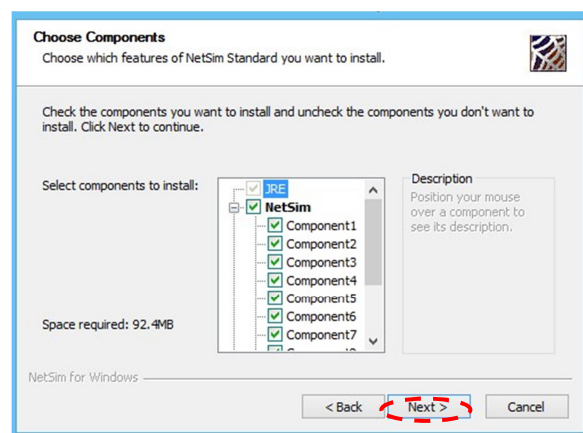


Figure 4: Installation step 4

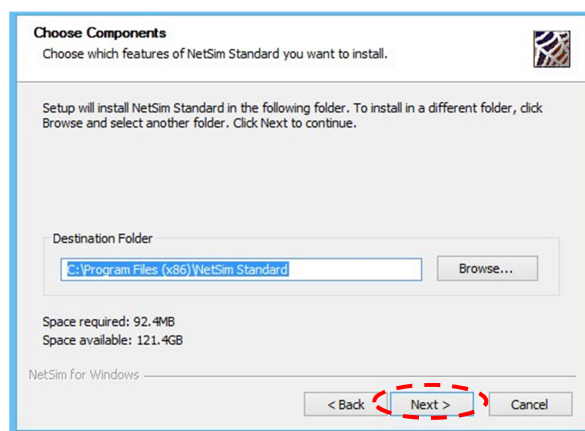


Figure 5: Installation step 5

In the next screen, you will be requested to enter the Start menu folder name.

Click on the **Install** button to start the installation.

The installation process begins. After the installation of required files, the installation of supporting software begins.

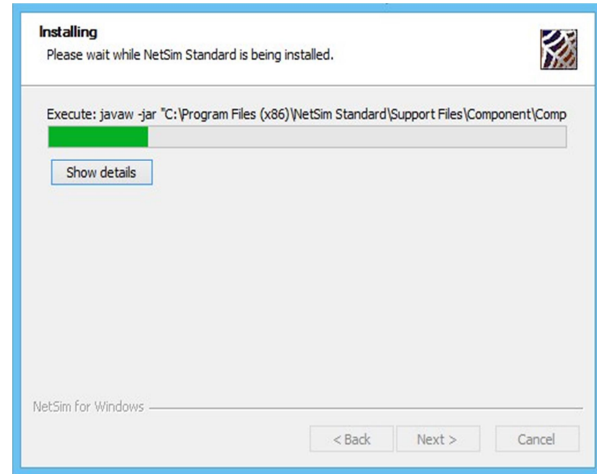


Figure 6: Installation step 6

For **NetSim Academic**, Adobe Flash Player will be installed. For **NetSim Standard Version** and **Pro Version**, gnuplot installation will start by default (if not deselected during 3<sup>rd</sup> party software selection)

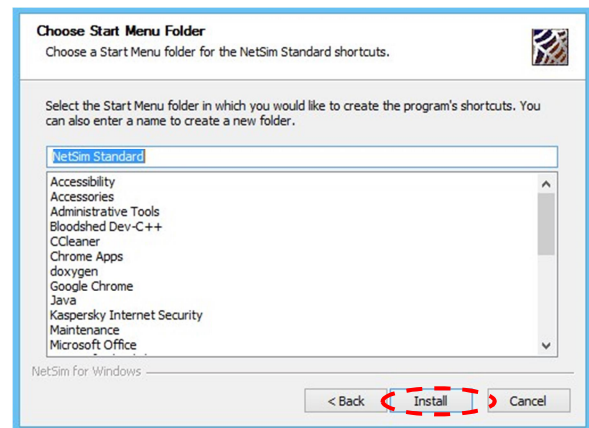


Figure 7: Installation step 7

The below screen will appear to choose the language.

After selecting **English language**, next screen is as follows

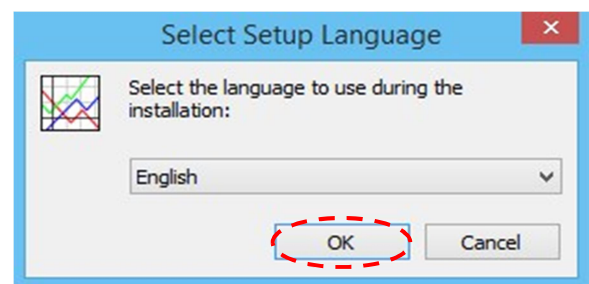


Figure 8: Installation step 8

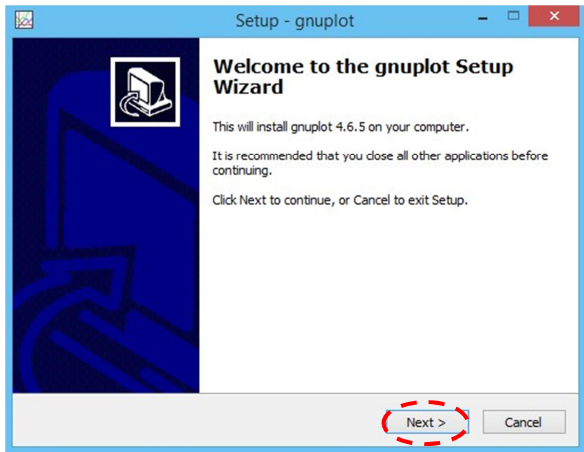


Figure 9: Installation step 9

Click on the **Next** button to install gnuplot (Supporting Software), License Agreement screen will appear as shown:

Click on **I accept the agreement** and then **Next** icon as shown above.

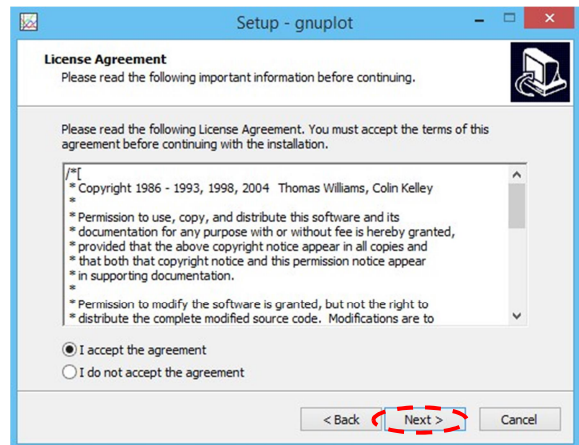


Figure 10: Installation step 10

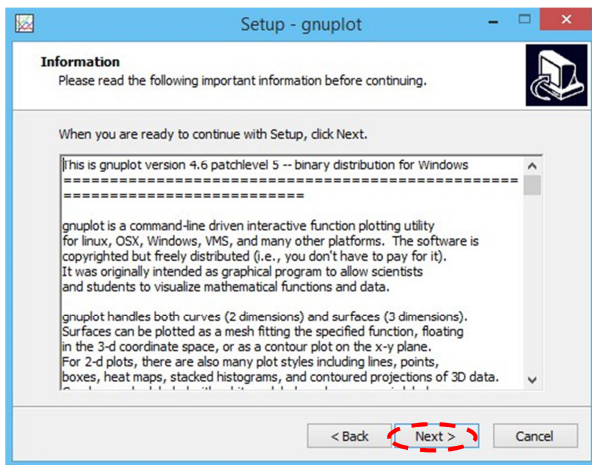


Figure 11: Installation step 11

Click on **Next**.

In the next screen, you will be requested to enter the installation path. Select the installation path and click on **Next** icon:

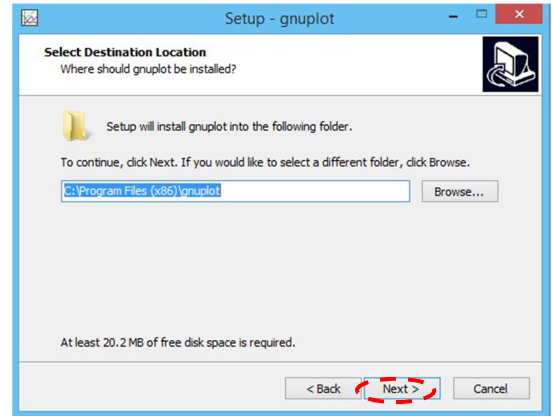


Figure 12: Installation step 12

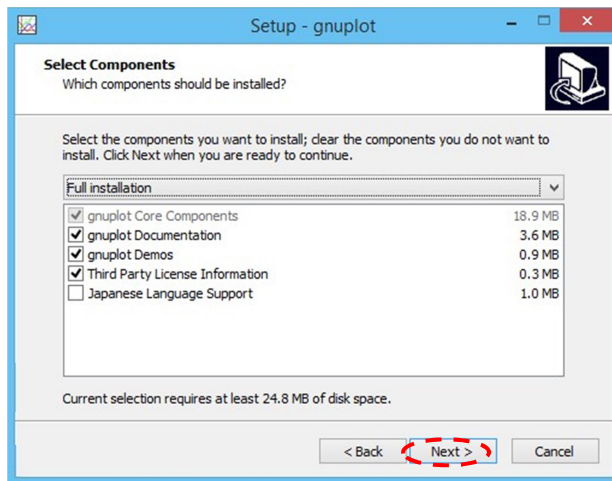


Figure 13: Installation step 13

In this screen, select **Full installation** and components to be installed and select **Next**. All the components are required except Japanese language support:

After the installation of the software, you will be requested to click Finish to complete the installation process.

**Note:** During the installation of NetSim Academic version the supporting software installed are Adobe Flash player and WinPcap.

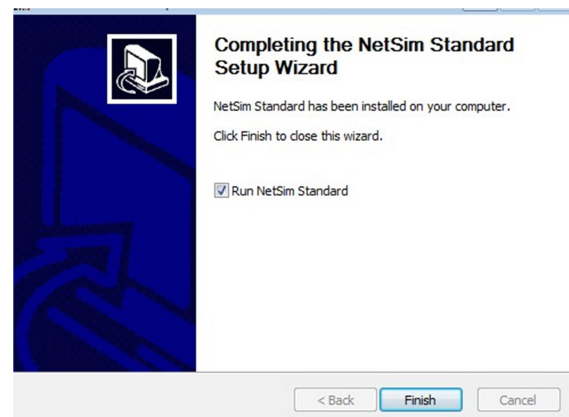


Figure 14: Installation step 14



## 2.5 Setting up License Server

### 2.5.1 Installing NetSim RLM Dongle

This section guides you to install the **RLM Dongle** software from the CD-ROM.

- “ Insert the CD-ROM disc in the CD drive.
- “ Double click on My Computer and access the CD Drive
- “ Double click on **Driver\_Software** folder.
- “ Open 32bit\_Installer folder or 64bit\_Installer folder depending on the system architecture.
- “ Double click on **HASPUserSetup.exe**

Each prompt displayed during the process tells you what it is about to do and prompts to either **continue** or **Exit**.

Setup prepares the installation wizard and the software installation begins with a **Welcome Screen**.

Click on the **Next** button

**Note:** Any other program running during the installation of the Dongle will affect the proper installation of the software.

In the next screen, the License agreement is displayed.

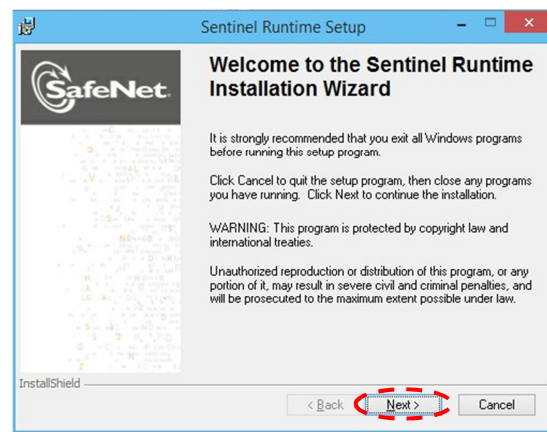
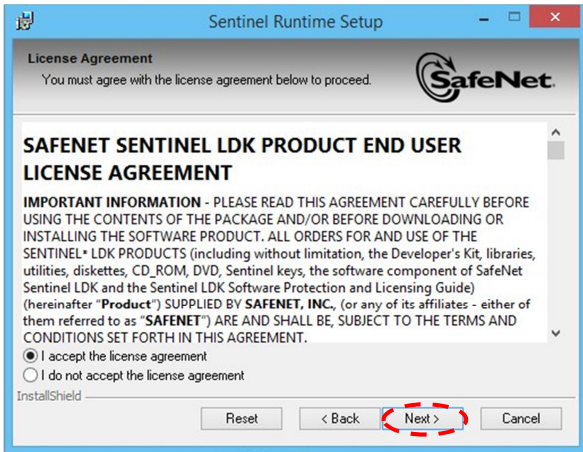


Figure 15: Installation step 15

Read the license agreement carefully, scroll down to read the complete license agreement. If the requirement of the license agreement is accepted select the **I accept** button else quit the setup by clicking **Cancel** button.



Click on the **Next** button

Figure 16: Installation step 16

Click on the **Next** button. The installation process begins.

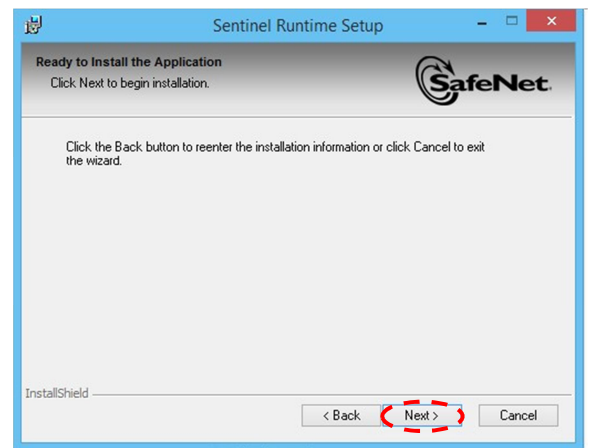


Figure 17: Installation step 17

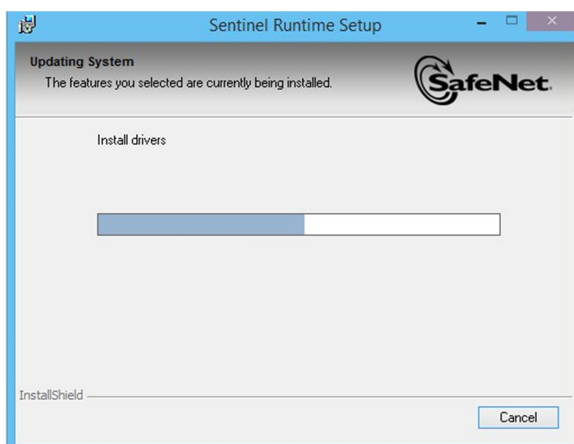


Figure 18: Installation step 18

After the installation of the software, you will be requested to click **Finish** button to complete the installation process.

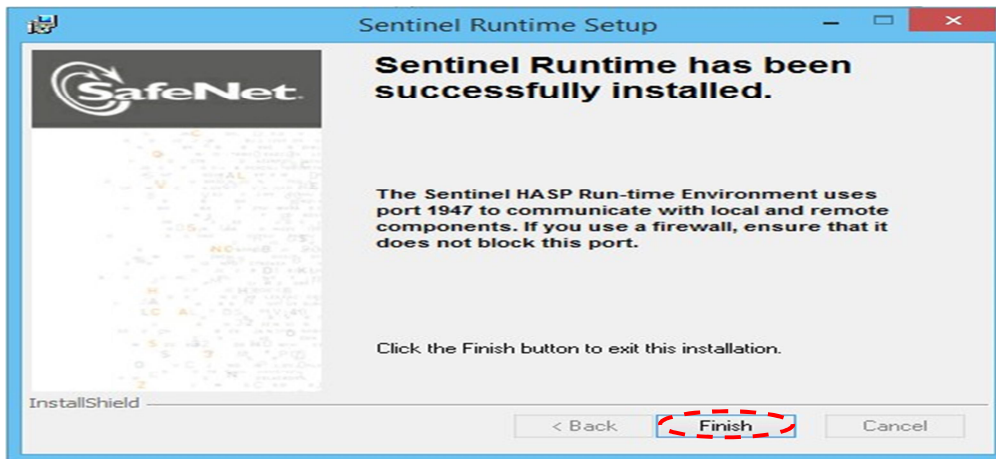


Figure 19: Installation step 19

Now the **RLM driver software** is installed successfully.



Figure 20: Installation step 20

If the driver has been successfully installed then upon connecting the Dongle in the USB port red light would glow (Refer picture below). If the driver is not correctly installed this light will not glow when the dongle is connected to the USB port.

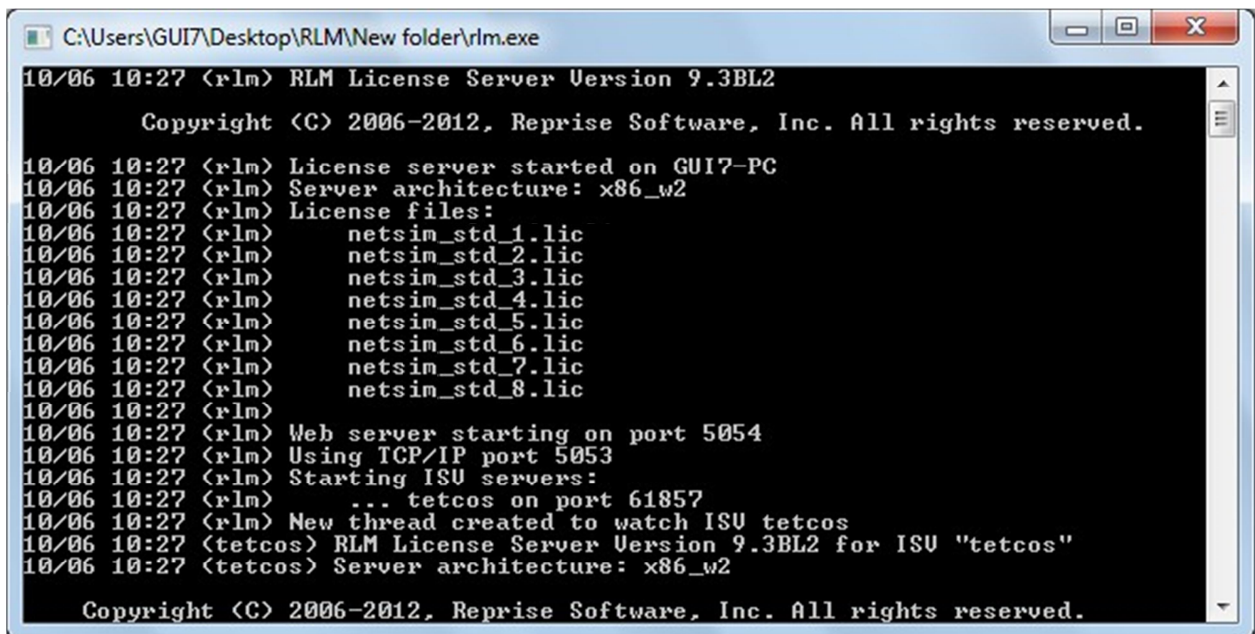
## 2.5.2 Running RLM Dongle (Server)

After the Driver Software installation, connect the **RLM dongle** to the **system USB port**.

- “ Double click on My Computer and access the CD Drive
- “ Copy the **NetSim License Server** folder and paste it on **Desktop**.
- “ Double click on **NetSim License Server** folder from Desktop.
- “ Double click on **rlm.exe**

**Note:** For running **NetSim**, **rlm.exe** must be running in the server (license server) system and the server system IP address must be entered correctly. Without running **rlm.exe**, **NetSim** won't run.

When you run **rlm.exe**, the screen will appear as shown below.



```
C:\Users\GUI7\Desktop\RLM\New folder\rlm.exe
10/06 10:27 <rlm> RLM License Server Version 9.3BL2
      Copyright (C) 2006-2012, Reprise Software, Inc. All rights reserved.
10/06 10:27 <rlm> License server started on GUI7-PC
10/06 10:27 <rlm> Server architecture: x86_w2
10/06 10:27 <rlm> License files:
10/06 10:27 <rlm>     netsim_std_1.lic
10/06 10:27 <rlm>     netsim_std_2.lic
10/06 10:27 <rlm>     netsim_std_3.lic
10/06 10:27 <rlm>     netsim_std_4.lic
10/06 10:27 <rlm>     netsim_std_5.lic
10/06 10:27 <rlm>     netsim_std_6.lic
10/06 10:27 <rlm>     netsim_std_7.lic
10/06 10:27 <rlm>     netsim_std_8.lic
10/06 10:27 <rlm> Web server starting on port 5054
10/06 10:27 <rlm> Using TCP/IP port 5053
10/06 10:27 <rlm> Starting ISU servers:
10/06 10:27 <rlm>     ... tetcos on port 61857
10/06 10:27 <rlm> New thread created to watch ISU tetcos
10/06 10:27 <tetcos> RLM License Server Version 9.3BL2 for ISU "tetcos"
10/06 10:27 <tetcos> Server architecture: x86_w2
      Copyright (C) 2006-2012, Reprise Software, Inc. All rights reserved.
```

Figure 21: Installation step 21

## 2.5.3 Running NetSim Software

After running **rlm.exe**, click the NetSim icon in the Desktop.

The screen given below will be obtained.

Enter the **Server IP address** where the **rlm.exe** is running, then click ok button.

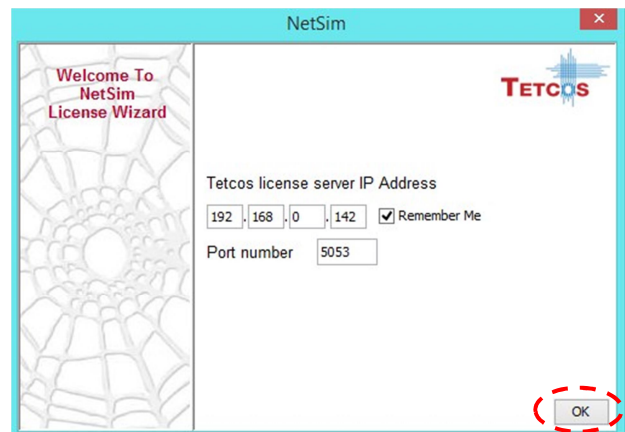


Figure 22: Installation step 22

You have now reached to the main menu of NetSim.



Figure 23: Installation step 23

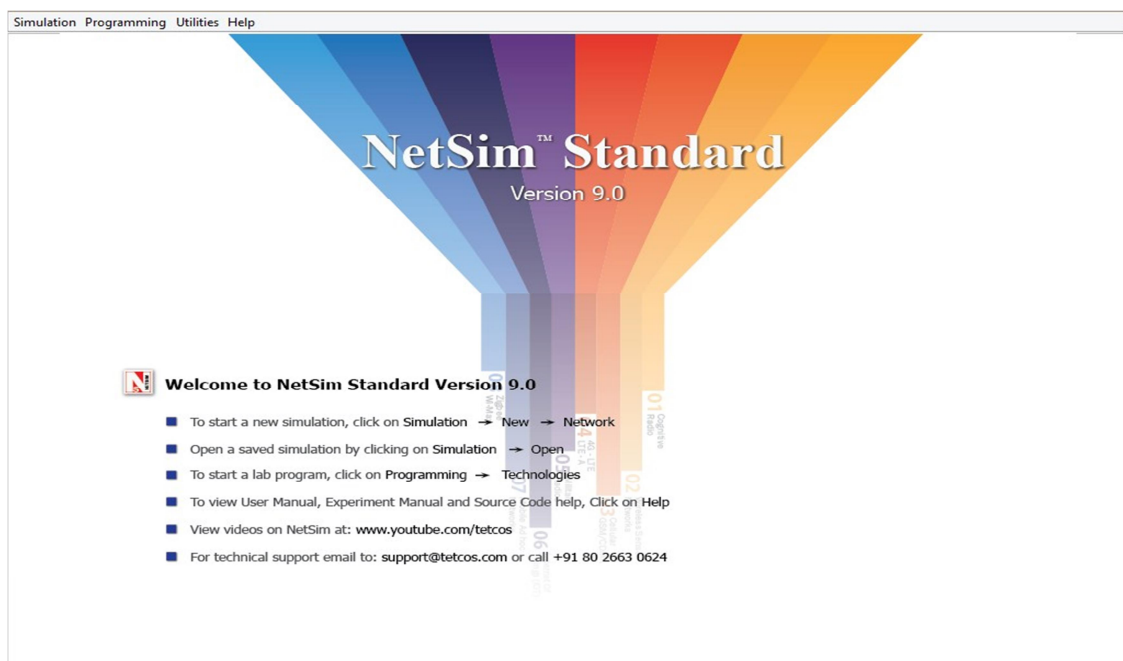


Figure 24: NetSim Home Page

## 3. INTRODUCTION TO MATLAB

### 3.1 What is MATLAB?

MATLAB [5] is widely used in all areas of applied mathematics, in education and research at universities, and in the industry. MATLAB stands for Matrix Laboratory and the software is built up around vectors and matrices. This makes the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. It is also a programming language, and is one of the easiest programming languages for writing mathematical programs. MATLAB also has some tool boxes useful for signal processing, image processing, optimization, etc.

### 3.2 The MATLAB ENVIRONMENT

The MATLAB environment (on most computer systems) consists of menus, buttons and a writing area similar to an ordinary word processor. There are plenty of help functions that you are encouraged to use. The writing area that you will see when you start MATLAB, is called the command window. In this window you give the commands to MATLAB. For example, when you want to run a program you have written for MATLAB you start the program in the command window by typing its name at the prompt. The command window is also useful if you just want to use MATLAB [5] as a scientific calculator or as a graphing tool. If you write longer programs, you will find it more convenient to write the program code in a separate window, and then run it in the command window.

In the command window you will see a prompt that looks like `>>`. You type your commands immediately after this prompt. Once you have typed the command you wish MATLAB to perform, press `<enter>`. If you want to interrupt a command that MATLAB is running, type `<ctrl> + <c>`.

The commands you type in the command window are stored by MATLAB and can be viewed in the Command History window. To repeat a command, you have already used, you can simply double-click on the command in the history window, or use the `<up arrow>` at the command prompt to iterate through the commands you have used until you reach the command you desire to repeat.

### 3.3 Key Features

- É High-level language for numerical computation, visualization, and application development
- É Interactive environment for iterative exploration, design, and problem solving
- É Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- É Built-in graphics for visualizing data and tools for creating custom plots

- É Development tools for improving code quality and maintainability and maximizing performance
- É Tools for building applications with custom graphical interfaces
- É Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft® Excel®

### **3.4 Numeric Computation**

MATLAB provides a range of numerical computation methods for analyzing data, developing algorithms, and creating models. The MATLAB language includes mathematical functions that support common engineering and science operations. Core math functions use processor-optimized libraries to provide fast execution of vector and matrix calculations.

Available methods include:

- É Interpolation and regression
- É Differentiation and integration
- É Linear systems of equations
- É Fourier analysis
- É Eigenvalues and singular values
- É Ordinary differential equations (ODEs)
- É Sparse matrices

MATLAB add-on products provide functions in specialized areas such as statistics, optimization, signal analysis, and machine learning.

### **3.5 Data Analysis and Visualization**

MATLAB provides tools to acquire, analyze, and visualize data, enabling you to gain insight into your data in a fraction of the time it would take using spreadsheets or traditional programming languages. You can also document and share your results through plots and reports or as published MATLAB code.

## **3.6 Acquiring Data**

MATLAB lets you access data from files, other applications, databases, and external devices. You can read data from popular file formats such as Microsoft Excel; text or binary files; image, sound, and video files; and scientific files such as netCDF and HDF. File I/O functions let you work with data files in any format.

Using MATLAB with add-on products, you can acquire data from hardware devices, such as your computer's serial port or sound card, as well as stream live, measured data directly into MATLAB for analysis and visualization. You can also communicate with instruments such as oscilloscopes, function generators, and signal analyzers.

## **3.7 Analyzing Data**

MATLAB lets you manage, filter, and pre-process your data. You can perform exploratory data analysis to uncover trends, test assumptions, and build descriptive models. MATLAB provides functions for filtering and smoothing, interpolation, convolution, and fast Fourier transforms (FFTs). Add-on products provide capabilities for curve and surface fitting, multivariate statistics, spectral analysis, image analysis, system identification, and other analysis tasks.

## **3.8 Visualizing Data**

MATLAB provides mathematical functions, as well as volume visualization functions. You can use these functions to visualize and understand data and communicate results. Plots can be customized programmatically. The MATLAB plot gallery provides examples of many ways to display data graphically in MATLAB. For each example, you can view and download source code to use in your MATLAB application.

## **3.9 Programming and Algorithm Development**

MATLAB provides a high-level language and development tools that let you quickly develop and analyses algorithms and applications.

### **3.9.1 The MATLAB Language**

The MATLAB language provides native support for the vector and matrix operations that are fundamental to solving engineering and scientific problems, enabling fast development and execution.

With the MATLAB language, you can write programs and develop algorithms faster than with traditional languages because you do not need to perform low-level administrative tasks such as declaring variables, specifying data types, and allocating memory. In many cases, the



support for vector and matrix operations eliminates the need for for-loops. As a result, one line of MATLAB code can often replace several lines of C or C++ code.

MATLAB provides features of traditional programming languages, including flow control, error handling, and object-oriented programming (OOP). You can use fundamental data types or advanced data structures, or you can define custom data types.

You can produce immediate results by interactively executing commands one at a time. This approach lets you quickly explore multiple options and iterate to an optimal solution. You can capture interactive steps as scripts and functions to reuse and automate your work.

MATLAB add-on products provide built-in algorithms for signal processing and communications, image and video processing, control systems, and many other domains. By combining these algorithms with your own, you can build complex programs and applications

### **3.9.2 Development Tools**

MATLAB includes a variety of tools for efficient algorithm development, including:

É Command Window - Lets you interactively enter data, execute commands and programs, and display results.

É MATLAB Editor - Provides editing and debugging features, such as setting breakpoints and stepping through individual lines of code.

É Code Analyzer - Automatically checks code for problems and recommends modifications to maximize performance and maintainability.

É MATLAB Profiler - Measures performance of MATLAB programs and Identifies areas of code to modify for improvement.

Additional tools compare code and data files, and provide reports showing file Dependencies, annotated reminders, and code coverage.

### **3.9.3 Integration with Other Languages and Applications**

You can integrate MATLAB applications with those written in other languages.

From MATLAB, you can directly call code written in C, C++, Java, and .NET.

Using the MATLAB engine library, you can call MATLAB code from C, C++, or FORTRAN applications.

## **3.10 Performance**

MATLAB uses processor-optimized libraries for fast execution of matrix and vector computations. For general-purpose scalar computations, MATLAB uses its just-in-time (JIT)

compilation technology to provide execution speeds that rival those of traditional programming languages.

To take advantage of multicore and multiprocessor computers, MATLAB provides many multithreaded linear algebra and numerical functions. These functions automatically execute on multiple computational threads in a single MATLAB session, enabling them to execute faster on multicore computers.

You can take further advantage of multicore desktop and other high-performance computing resources such as GPUs and clusters with add-on parallel computing products. These products provide high-level constructs that let you parallelize applications with only minor changes to MATLAB code.

### **3.11 Application Development and Deployment**

MATLAB tools and add-on products provide a range of options to develop and deploy applications. You can share individual algorithms and applications with other MATLAB users or deploy them royalty-free to others who do not have MATLAB.

### **3.12 Designing Graphical User Interfaces**

Using GUIDE (Graphical User Interface Development Environment), you can lay out, design, and edit custom graphical user interfaces. You can include common controls such as list boxes, pull-down menus, and push buttons, as well as MATLAB plots. Graphical user interfaces can also be created programmatically using MATLAB functions.

### **3.13 Deploying Applications**

To distribute an application directly to other MATLAB users, you can package it as a MATLAB app, which provides a single file for distribution. Apps automatically install in the MATLAB apps gallery for easy access.

To share applications with others who do not have MATLAB, you can use application deployment products. These add-on products automatically generate standalone applications, shared libraries, and software components for integration in C, C++, Java, .NET, and Excel environments. The executables and components can be distributed royalty-free.

## 4. Problem Description

Problem of coverage and connectivity are critical for randomly distributed wireless sensor networks. Connectivity is measured in terms of successful packet transmission to the sink. However this could not be the only metric measurement of connectivity; how efficiently connection is established also matters. At the same time optimizes coverage for the region is required for quality of service. While addressing the problem of coverage and connectivity, issue of energy consumption and balancing is mostly ignored. Some efforts have been made by researchers to address the problem of energy balancing independent of coverage and connectivity. Therefore, to increase the lifetime of the network, one needs to balance the energy consumption among sensors with maintaining satisfactory coverage and connectivity. The problem is formulated as follow:

1. Sensors in a wireless sensor network primarily perform two task, sensing and transmission. For sensing, we assume that each sensor takes  $E$  units of energy per unit time. According to free space path loss model, the received power  $P$  at a distance  $d$  from a sensor is given by following equation

$$P_r = P_t \left( \frac{\lambda}{4\pi d} \right)^2 G_t G_r \quad (1)$$

Where  $P_r$  represents the transmission power used by a sensor,  $\lambda$  is the wavelength of signal, and  $G_t$  and  $G_r$  are antenna gains for transmitting and receiving sensors respectively. We note that power consumed in sensing is directly proportional to the square of distance.

We have

$$E_{s1} \propto r_1^2 \quad (2)$$

$$E_{s2} \propto r_2^2 \quad (3)$$

Where,  $E_{s1}$  and  $E_{s2}$  are energy consumed by sensors  $S_1$  and  $S_2$  with sensing range  $r_1$  and  $r_2$  respectively.

From equation (1) and (2) we get:-

$$\frac{E_{s1}}{E_{s2}} = \frac{r_1}{r_2} \quad (4)$$

Another task performed by a sensor is transmission of accumulate data. We assume that  $E$  is the energy consumed per unit time in transmission between two sensors separated by distance  $t$ . The path loss formula given in (1) is also applicable for this case, therefore, we have:-

$$\frac{E_{t1}}{E_{t2}} = \frac{t_1}{t_2} \quad (5)$$

We assume that sensing region is circular in shape having radius  $R$ . Further, we also assume that data generated by each sensor in a sensing region follows binomial distribution and routed to the sink which is situated in the center of the region. Our objective, in this work is to fairly equalize the total energy consumption of each sensor over a specified period of time. We consider that the energy consumed during time  $T$  in sensing and transmission by  $i$ th sensor are  $E_i^s$  and  $E_i^t$  respectively. Therefore, the total energy consumed during time  $T$  by  $i$ th sensor can be expressed as

$$E_i^s + E_i^t = E_i^T \quad \forall i = 1 \dots N \quad (6)$$

Where,  $N$  is total number of sensors deployed. Our objective is to keep  $E_i^T$  fairly equal for all sensors, i.e.  $E_1^T = E_2^T = \dots E_{N-1}^T = E_N^T$ . Further, energy  $E_i^T$  is also referred to as load  $i$ th sensor. While achieving this objective we are required to maintain a satisfactory coverage and connectivity in the sensing region. The notations used are given in table.

## Notation Table

$E_s$	Energy required per unit time in sensing
$P_r$	Received power
$d$	Distance between two sensors
$P_t$	Transmitted power
$\delta$	Wavelength of the signal
$G_t$	Antenna gain of transmitter
$G_r$	Antenna gain of receiver
$r_i$	Sensing range of $i$ th sensor
$E_t$	Transmission energy required per unit time for distance $t$
$t_i$	Transmission distance
$E_i^S$	Energy consumed in sensing by $i$ th sensor in time $T$
$E_i^t$	Energy consumed in transmission for $i$ th sensor in time $T$
$E_i^T$	Total energy consumed in both sensing and transmission during time $T$
$R$	Radius of circular sensing region
$N$	Total number of sensor deployed
$L$	Total load on sensors in all annulus
$L_i$	Load on all sensors in $i$ th annulus
$\delta$	Spacing parameter between each concentric circles
$X(n)$	Probability of data generated in $n$ th annulus
$Y$	Poisson random variable
$f_y(n)$	Probability density function of $Y$ representing load on sensors in $n$ th annulus
$f_{GU}(x)$	Probability density function of load on sensors under uniform data flow

$f_B(r)$	Binomial distribution probability mass function
$f_{GA}(x)$	Probability density function of load on sensors under accumulated data flow
$W_i$	Exponential distributed waiting time for $i$ th sensor
$\varepsilon_i$	Rate parameter for $W_i$
$h_i$	Minimum hop-counts at $i$ th sensor
$\eta_i$	Density of sensors achievable in $i$ th hop count
$R_i$	Transmission range of sensor in $i$ th annulus
$A_i$	$i$ th Annulus
$A$	Total number of annulus in the region

Table 3: Notation Table [1]

## 5. Mathematical Analysis of Load on Sensors

We present a mathematical model for energy balancing based on uniform data flow and accumulated data flow. Load analysis has been performed for both uniform and accumulated data flow. In the following section we analyze the distribution pattern of sensors and its impact on the load.

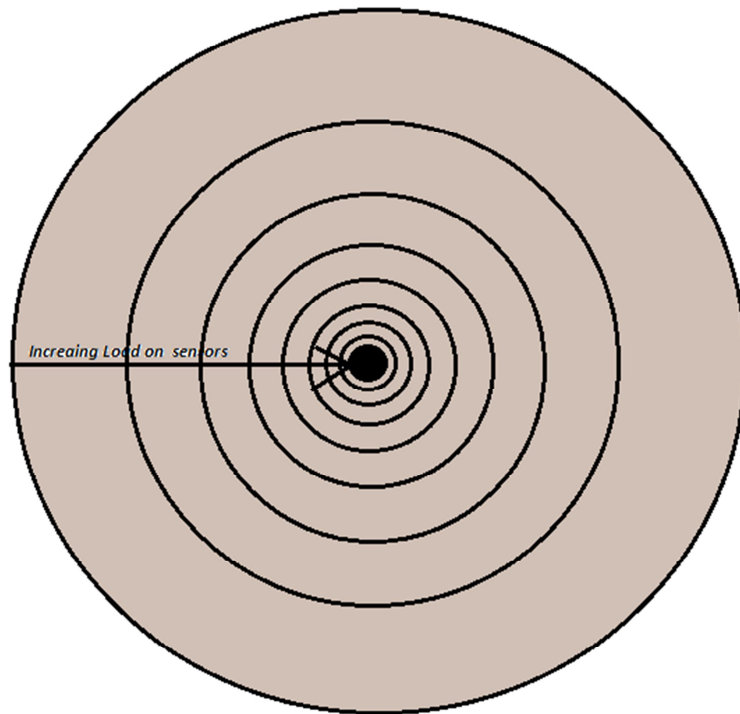


Figure 25: Load on Sensors

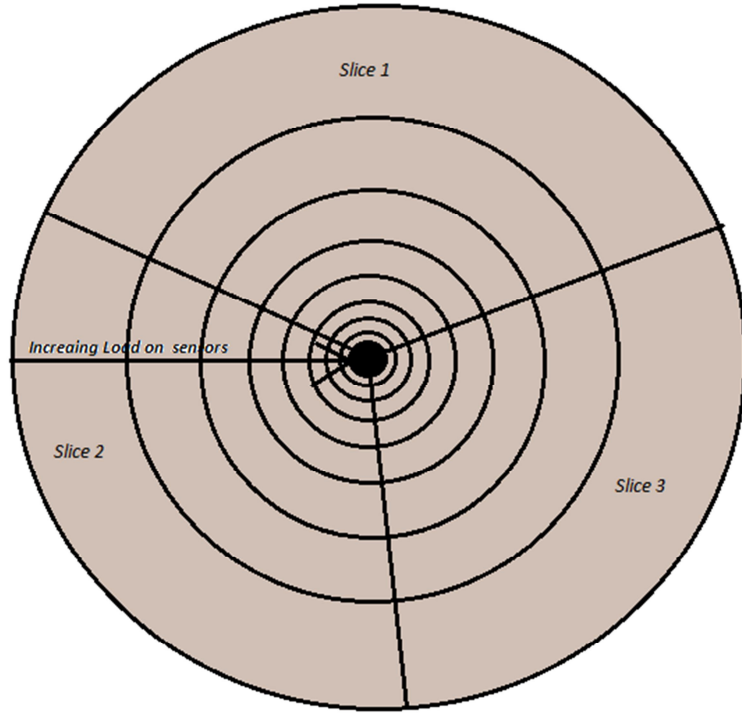


Figure 26: Slice of the region

## 5.1 Load Analysis under Uniform Data Flow

Without losing the generality of problem, we assume that sensing region is circular in shape with radius  $R$ . It is also assumed that sink is situated at the center of sensing region. In view of the assumption, it is clear that data sensed in the entire region ultimately reach the center of circle. In this approach, sensors near the center deplete their energy faster. Therefore, for fair distribution of sensor or load, we have divided sensing region in concentric circle with approximately equal spacing. Using this model, we analyze how the rate of flow of data exert transmission load on sensors in each concentric circle towards the sink.

Let total amount of data to be carried through each annulus from outer most to the sink is  $L$ . Let spacing between each concentric circle is decreasing by a very small factor  $\delta$  from outside to inside. Radii of circle from outside to inside can be given by  $R$ ,  $\delta R$ ,  $\delta^2 R$  and so on. Area of outer most annulus is given by  $\pi(R^2 - \delta^2 R^2)$ . In the same way we can calculate the area of outer annulus as follows:  $\pi(\delta^2 R^2 - \delta^4 R^2)$ ,  $\pi(\delta^4 R^2 - \delta^6 R^2)$  and so on. We assume that sensors are distributed uniformly in each annulus and the load in an



annulus is equally distributed among sensors in that annulus. Load in each annulus is given by the following formula.

$$Load = \frac{\text{Amount of data in the annulus}}{\text{area of the annulus}} \quad (7)$$

Let  $X$  is a random variable such that  $X(n)$  represent the probability that data is generated in  $n$ th annulus ( from outside to inside ) by following uniform distribution. The Probability  $X(n)$  can be calculated as follows.

$$X(n) = \frac{\pi(\delta^{2n} R - \delta^{2n+2} R^2)}{\pi R^2}, \quad 0 < \delta < 1 \text{ and } n \geq 0 \quad (8)$$

## 5.2 Theorem 1

**$X$  is geometrically distributed.**

**Proof** Probability of data generated in  $n$ th annulus is given by  $X(n) = \frac{\pi(\delta^{2n} R - \delta^{2n+2} R^2)}{\pi R^2}$

. By solving the expression on right hand side of the Eq. (8), we get following series  $(1 - \delta^2), \delta^2(1 - \delta^2), \delta^4(1 - \delta^2), \dots$  for different possible values on  $n$ . This is a geometrically distributed series with parameter  $(1 - \delta^2)$  hence the result.

## 5.3 Theorem 2

**Geometric distribution  $X(n)$  in limiting case can be expressed by exponential distribution.**

**Proof**

We assume that  $q = 1 - \delta^2$  is very small and  $mq = \mu$  where  $m$  is supposed to be large and  $x = \frac{n}{m}$  From Eq. (8), we have

$$\sum_{n=0}^{\infty} q(1-q)^n = 1$$

$$\sum_{n=0}^{\infty} \mu \left(1 - \frac{\mu}{m}\right)^{\frac{n}{m}} \frac{1}{m} = 1$$

As  $m \rightarrow \infty$ , and applying  $(1 - z) \approx e^{-zm}$ , we have

$$\int_0^{\infty} \mu e^{-\mu x} dx = 1 \quad (9)$$

Where  $\mu = m(1 - \delta^2)$  From Eq. (9), it is clear that data generated in all annuli is exponentially distributed from outside to inside.

### 5.4 Theorem 3

**The probability density function of load on sensors given in Eq. (9) yields to Gaussian distribution in limiting case from inner to outer annulus.**

#### Proof

We assume that Y is a random variable where  $Y = 1/X$ , and Y follow Poisson distribution.

The probability density function  $f_y(n)$  for Y can be given as:

$$f_y(n) = \frac{(m(1 - \delta^2))^n e^{-m(1 - \delta^2)}}{n!} \quad (10)$$

Let  $m(1 - \delta^2) = \mu$  and  $x = n = \mu(1 + \varepsilon)$ ,  $\mu \gg 1$  and  $\varepsilon \ll 1$

Since coverage to  $\mu$  and  $x \approx \sqrt{2\pi x} \left(\frac{e}{x}\right)^n$  as  $x \rightarrow \infty$ , we have

$$\begin{aligned} f_y(n) &= \frac{(\mu)^{\mu(1+\varepsilon)} e^{-\mu}}{\sqrt{2\pi e^{-\mu(1+\varepsilon)} [\mu(1+\varepsilon)]^{\mu(1+\varepsilon)+\frac{1}{2}}}} \\ &= \frac{e^{\mu\varepsilon} [(1+\varepsilon)]^{-\mu(1+\varepsilon)-\frac{1}{2}}}{\sqrt{2\pi\mu}} \\ &= \frac{e^{-\mu\varepsilon^2/2}}{\sqrt{2\pi\mu}} \end{aligned}$$

By substituting the value of  $\mathcal{E}$  the probability density function  $f_{GU}(x)$  of the load on sensors under uniform data flow can be expressed as:

$$f_{GU}(x) = \frac{e^{-(x-\mu)^2/2\mu}}{\sqrt{2\pi\mu}}$$

$$f_{GU}(x) = \frac{e^{-(x-m(1-\delta^2))^2/2m(1-\delta^2)}}{\sqrt{2\pi m(1-\delta^2)}} \quad (11)$$

The distribution of load in the various annuli has been analyzed by simulating. For  $m=100000$  and  $p=.99$  the distribution of load in various annuli has been show in Fig. 2

The result in Fig. 2 shows that the distribution of load on sensors from inner to outer annulus follows folded Gaussian distributed in limiting case. Sensors are deployed as per Eq. (11) with appropriate mean and variance to equalize the load in network.

## 5.5 Load Analysis under Accumulated Data flow

In this section, we analyze the load on each sensor under accumulated data flow. Sensors are distributed according to Gaussian distribution as described in previous section by Eq.(11) to compensate for excessive load on sensors in the inner annulus under uniform data flow model. It is assumed that sensors are generating data following binomial distribution with parameter  $t$ . Let the area of each annulus is sufficiently large to accommodate a large number of sensors to it. Probability mass function of data generated by  $r$  out of  $n$  sensors is given by:

$$f_B(r) = \frac{n!}{r!(n-r)!} t^r (1-t)^{n-r} \quad (12)$$

Let  $t = \frac{\mu}{s}$  where  $s$  is assumed to be large and  $t$  is small. According to the law of rare events,

We obtain:

$$f_P(r) = \frac{1}{r!} \mu^r e^{-\mu} \quad (13)$$

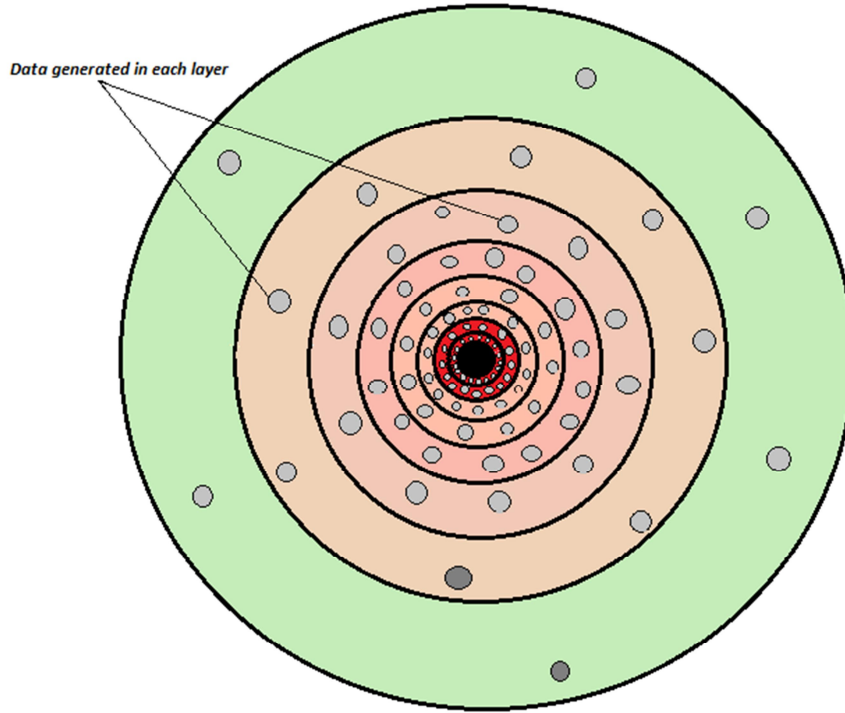


Figure 27: Data Flow

Expression (13) represents a probability mass function for Poisson distribution. Data generated from each annulus is assumed to follow Poisson Process. Under accumulated data flow model, we have to analyze the load on each sensor in each annulus. It is apparent from the Fig. 3 that inner annuluses have to carry not only their own data but also data generated from outer annuluses. We have assumed that for a large number of annulus this data distribution follow Gaussian distribution as proved in theorem 3. Therefore, probability density function of the load on sensors can be given as:-

$$f_{GA}(x) = \frac{e^{-(x-\mu)^2 / 2\mu}}{\sqrt{2\pi \mu}} \quad (14)$$

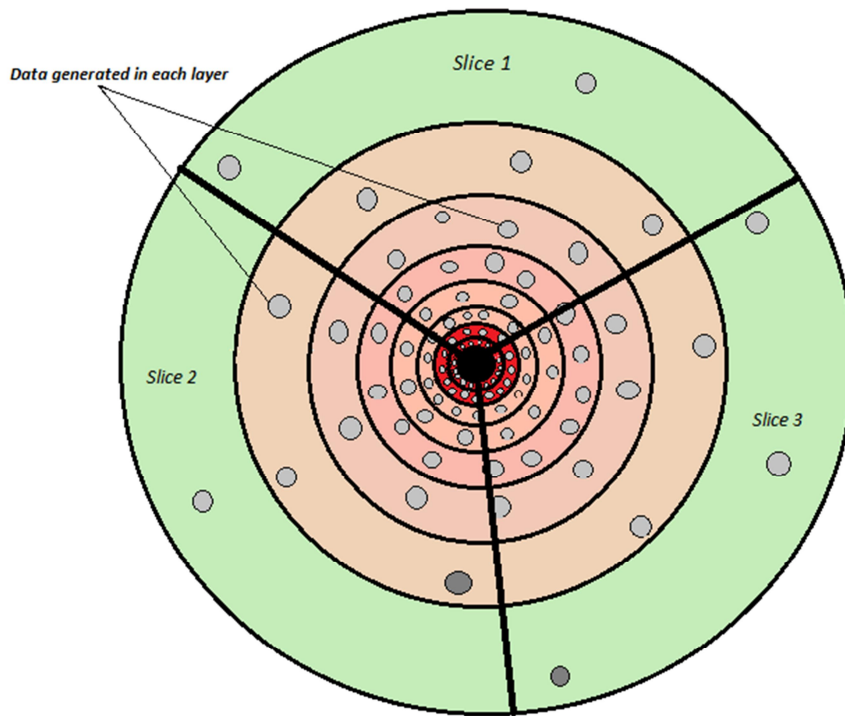


Figure 28: Data Flow in Slice

## 5.6 Load Equalization

Before equalization of load, we need to simulate Eq. (7) to show the distribution of load in the various annuli. Result obtained through is depicted in Fig.4 for  $t=0.1$  and  $s=100,000$  using Eq. (12) for generation of data by sensors. It shows that due to accumulated data flow from outer annulus to the sink, load on sensors in inner annulus increases rapidly.

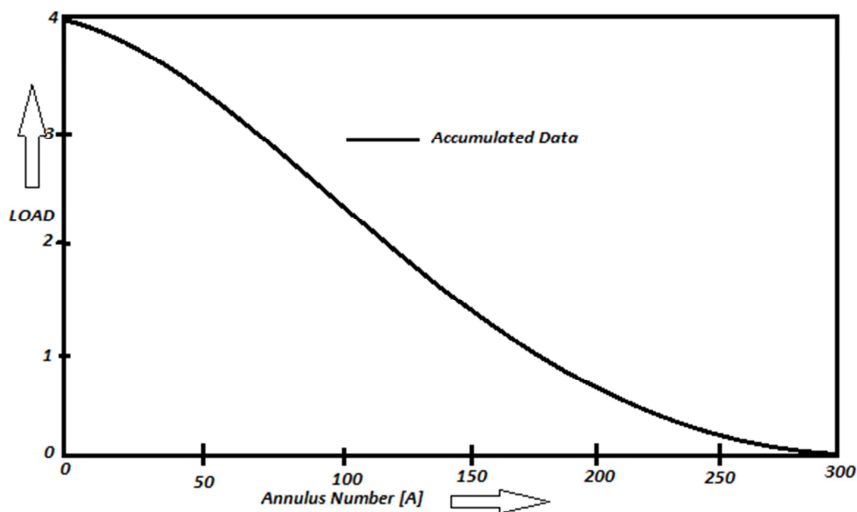


Figure 29: Data Accumulation graph

We present a scheme to equalize the load on sensor in different annulus. In the inner annulus, sensors transmit more packets in comparison to those in outer annulus. We have analyzed the impact of increasing data load on sensors. It was observed that the load follows folded Gaussian distribution.

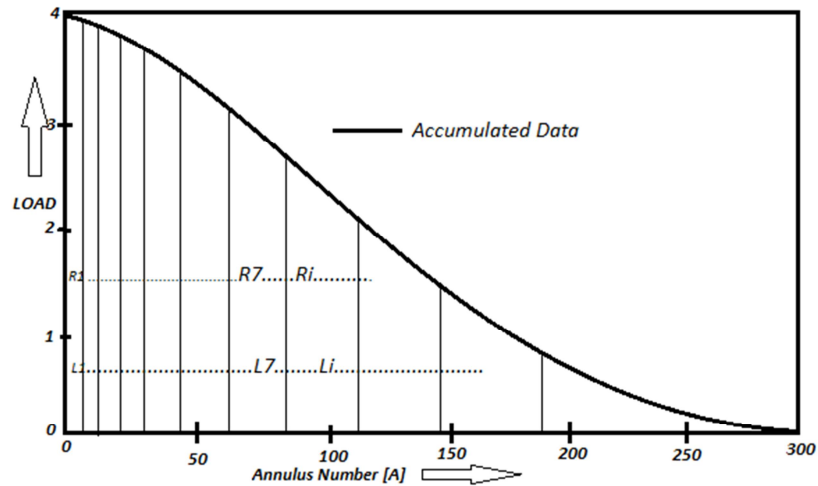


Figure 30: Exponential data Congestion

We have presented the load curve for accumulated data in Fig. 5. From the figure, the area of an annulus is calculated by multiplying height of load curve over annulus with the width of annulus. For equalizing the load, we take rectangles of equal area throughout the curve. These areas can be kept equal if we increase the width with increasing distance from the origin as shown in Fig. 5. However, in this discussion annulus number represents the number of sensors in that annulus. Therefore, by changing the width of annulus we are varying the sensing range or transmission range of the sensors in the annulus. The area of sensors can be controlled by either controlling the sensing or transmission range. Sensing range of sensors can be controlled by scheduling the sensors. Transmission range of a sensor can be controlled by designing appropriate routing protocols. We have used adaptive sensing and transmission method for equalizing excessive load on sensors. To accomplish this objective we present scheduling and routing algorithms in the next section.

## 6. Algorithms Implemented

In this section, supporting algorithms for energy balancing are presented. These algorithms include creation of annulus in the sensing region, formation of slices, connectivity ensures routing and coverage preserved scheduling.

### 6.1 Annulus Forming Algorithm (AFA)

We assumed that  $R_i$  is the transmission range of a sensor in  $i$ th annulus. The width of an annulus is assumed to be twice of the transmission range of sensor in that annulus. Therefore, it is important to determine the values determine the value of  $R_i$ . We derive transmission range  $R_i$  for sensors according.

Node Id.	Recived Power	Hop Count	Ring, Slice No.
----------	---------------	-----------	-----------------

Table 4: Routing Table

To their distance from the sink by using  $L_i R_i = L_{i-1} R_{i-1} \forall \geq 1$  or  $R_i = \frac{L_{i-1} R_{i-1}}{L_i}$  value of  $L_i$  is computed from (14) as:

$$L_i = \frac{e^{-(A_i - \mu)^2 / 2\mu}}{\sqrt{2\pi\mu}} \quad (15)$$

In the Eq. (15)  $A_i$  is the annulus number determined using the following function.

$$A_i = \begin{cases} h_i / 2 \rightarrow & \text{if } h_i \text{ is even} \\ h_{i-2} / 2 \rightarrow & \text{if } h_i \text{ is odd} \end{cases} \quad (16)$$

Where  $h_i$  is the minimum number of hop counts from sink to  $i$ th sensor. To find  $h_i$ ; sink node sends a Hello packet to its neighboring sensors with hop count 0. All the sensors

nodes, which receive the  $\delta$ Hello packet, increase the hop count by 1 and broadcast the packet further. A sensor node sends  $\delta$ Hello packet if it has lower hop counts than the previously received packet and remembers the packet that has lowest hop counts, i.e.  $h_i$ . Every sensor node finalizes its annulus number after elapsed of a period sufficient to deliver a  $\delta$ Hello packet from sink to the farthest sensor node. A sensor stores the identity of all the sensors which have sent it the hello packet. This information is used in routing of the data packets. In Fig.7 we have depicted the fields of a node of Hello list.

Broadcasting of the  $\delta$ Hello packet by sensors may result in collision of packets. To avoid the collisions a sensor has to wait for a random period before sending the  $\delta$ Hello packet. This waiting time is determined by using a parameter  $\varepsilon_i$  for  $i=1, 2, 3, \dots$  that depends on the hop count. We assumed that waiting times for the sensors are exponential distributed. Therefore, waiting time is given by  $W_i = \varepsilon_i e^{-\varepsilon_i t}$ , where  $t$  is a uniform random variable with interval  $[0,1]$ . We assumed that waiting time of sensor is inversely proportional to hop count, i.e. a node having  $\delta$ Hello packet with minimum hop count value should wait for minimum period. The parameter  $\varepsilon_i$  depends on the density of sensors in nearby region. If density is high, sensors must wait for longer period for avoiding possible collision and therefore,  $\varepsilon_i \propto \frac{1}{\eta_i}$  where  $\eta_i$  is the density of sensors in  $i$ th annulus. Density of sensors decreases from the inner to outer annulus and can be expressed as follows, if the sensors are deployed using Eq. (11)

$$\eta_i \propto \frac{1}{\frac{e^{-(h_i - m(1-\delta^2))^2 / 2m(1-\delta^2)}}{\sqrt{2\pi m(1-\delta^2)}}} \quad (17)$$

$$\varepsilon_i = C \frac{e^{-(h_i - m(1-\delta))^2 / 2m(1-\delta^2)}}{h_i \sqrt{2\pi m(1-\delta^2)}} \quad (18)$$

Where,  $C$  is a normalizing constant that depends on the number of sensors in the region. Therefore, the waiting time for a sensor can be given as follows

$$W_i = C \frac{e^{-(h_i - m(1-\delta^2))^2 / 2m(1-\delta^2)}}{h_i \sqrt{2\pi m(1-\delta^2)}}$$



$$W_i = C \frac{e^{-(h_i - m(1-\delta))^2 / 2m(1-\delta^2)}}{h_i \sqrt{2\pi m(1-\delta^2)}} e^{-C \frac{e^{(h_i - m(1-\delta))^2 / 2m(1-\delta^2)}}{h_i \sqrt{2\pi m(1-\delta^2)}} t} \quad (19)$$

The algorithm for annulus formation is presented below:

### 6.1.1 Algorithm 1

1.  $W = 0$  / variable to store previous waiting time \*/
2.  $W_i =$  time for which sensors has to wait before sending a hello packet
3.  $h_i =$  current hop count of a sensor
4. begin
5.  $i$ th sensor receives a Hello Packet
6.  $i$ th sensor receives the hop count value  $h$
7. if ( $h < h_i$ )
8.      $h_i = h$   
        Annulus number is derived using equation
        
$$A_i = \begin{cases} \frac{h_i}{2} \Rightarrow \Rightarrow \text{if } h_i \text{ is even} \\ \frac{h_i - 1}{2} \Rightarrow \Rightarrow \text{if } h_i \text{ is odd} \end{cases}$$
9.      $W_i = \varepsilon e^{-\varepsilon t}$
10.      $W_i = W_i - W$
11.      $R_i = \frac{L_{i-1} R_{i-1}}{L_i}$
12.     Wait( $W_i$ )  
        broadcast the Hello Packet
13.     else
14.         discard the packet /\* delayed packet \*/
15.     end if
16.     end begin

## 6.2 Connectivity Ensured Routing Algorithm (CERA)

We present a routing algorithm to realize Balancing Model proposed to rationalize the energy consumption by the sensors. The algorithm is designed to deliver a packet from a sensor to the sink using minimum number of hops. To forward packet a sensor selects a next hop sensor from its Hello list. While selecting next hop sensor from the Hello list, sensors from lower level annulus are preferred over sensors from same annulus. For rationalization of energy among sensors, from an annulus a sensor with the highest residual energy is selected.

### 6.2.1 Algorithm 2

1.  $p_{id}$  : packet identity assigned to the packet
2.  $s_{id}$  : sender identity
3.  $r_{id}$  : received identity
4. begin
5. A packet is generated or received at  $i$ th sensor
6. If (data is generated by itself)
7. /\* select a next hop sensor to forward the packet \*/
8. if (sensor is lower annulus is available in the Hello List)
9. choose one with highest residual energy.
10. else
11. choose a sensor from its own annulus with highest residual energy
12. end if
13.  $s_{id}$  = identity of a sensor which is generating packet
14.  $r_{id}$  = identity of a next hop sensor
15. transmit packet with transmission range  $R_i$
16. end if
17. end begin.

### 6.3 Coverage Preserved Scheduling Algorithm (CPSA)

In this section, we present coverage preserved scheduling algorithm for Reliable Energy Model to further rationalize energy. The algorithm for routing and connectivity rationalize energy consumed in transmission only. However, energy consumed in sensing also needs to be rationalized. A large number of sensors are deployed in the sensing region to achieve satisfactory coverage that results in redundancy. We proposed coverage preserved scheduling algorithm which sets off redundant sensors for a specific period. Identification of redundant sensors while preserving coverage is critical for scheduling. Therefore, in the proposed algorithm, we have developed a relative coordinate system that helps in identifying redundant sensors. In the relative coordinate of two sensors. In this system, we designate any sensor to be located at origin (0,0). We choose another sensor which is assumed to be located on x-axis at distance d from sensor at origin. Therefore, coordinate of this sensor is (d, 0). By using the geometry, now we can determine the relative coordinate of any other sensor by using the coordinate, i.e. (0, 0) and (d, 0) of the two known sensors.

In Fig. 8, we have designated a sensor O at (0, 0) and chosen another A with coordinate (d, 0). With help of this, the relative coordinate of sensor B is calculated as

$$ON = a = OB \cos(BON) \quad (20)$$

$$BN = b = OB \sin(BON) \quad (21)$$

Where the angle BON is given by  $\cos(BON) = \frac{BO^2 + ON^2 - BN^2}{2 \cdot BO \cdot ON}$ . Similarly, we can calculate the coordinates of other sensors in the network. Once, the relative coordinates of all sensors are known, we can determine redundant sensors by using concept of geometry as follows.

We consider a sensor with coordinate (m, n) and sensing range  $r_1$ . Now choose its neighboring sensor within sensing range having coordinate (p, q) and sensing range  $r_2$ . The coverage of these sensors can be expressed by the equation of circle as:

$$(x - m)^2 + (y - n)^2 = r_1^2 \quad (22)$$

$$(x - p)^2 + (y - q)^2 = r_2^2 \quad (23)$$

Now we can determine intersection points of these two circles by solving the above equations. Similarly, we can determine intersection points of sensor at  $(m, n)$  with its all neighboring sensors. Now middle points of all pairs of consecutive intersection points are determined. If every middle points are covered by at least one neighboring sensor, the sensor  $(m, n)$  is fully covered and therefore, considered to be redundant.

### 6.3.1 Algorithm 3

1. *Begin*
2. *Let the coordinate of sink are  $(0, 0)$*
3. *Let a sensor  $S_A$  at distance  $d$  from sink has coordinate  $(d, 0)$*
4. *for (each sensor  $S_B$ )*
5. *Find relative coordinate of  $S_B$  using coordinates of  $S_s$  and  $S_A$ ; // sensors are localized*
6. *One of the sensor  $S_i$  is ready to go in sleep mode // scheduling starts here.*
7. *for (each neighboring sensor  $S_i$  of  $S_j$ )*
8. *Find intersecting points of coverage area of  $S_i$  and  $S_j$  ;*
9. *for (each pair of consecutive intersection points)*
10. *find middle point;*
11. *if (every middle points is covered by at least one neighboring sensor )*
12. */\* Sensors  $S_i$  is fully covered by neighboring sensors and is redundant \*/;*
13.  *$S_i$  goes to sleep mode;*
14. *else*
15.  *$S_i$  cannot go to sleep mode;*
16. *end if*
17. *end begin.*

The algorithm for annulus formation, slice formation, connectivity ensured routing and coverage preserved scheduling are backbone of the REBM (Reliable Energy Model) Hello packets are used in annulus formation in AFA algorithm. Hello packets add to overheads in terms of energy consumption in WSNs. But at the same time, information gathered by exchange of Hello Packets is used in routing. In other word no additional packet are required for routing for route discovery. Second, formation of annulus is also used for transmission power control by adjusting the transmission range in each annulus. Therefore, the energy consumed in transmission of Hello packets in annulus formation is a gain in energy saving in terms of route discovery and transmission power control.

## 6.4 Slice Formation

Each ring are divided in multiple slice by radial line originating form center. In this work we divided each concentric rings in four slices. For the dividing concentric rings in slices we developed following algorithm.

```
CLUSTER (0,0) [ ]
```

```
CLUSTER (0,1) [ ]
```

```
CLUSTER (0,2) [ ]
```

```
CLUSTER (0,3) [ ]
```

```
CLUSTER (1,0) [ 37 ]
```

```
CLUSTER (1,1) [ ]
```

```
CLUSTER (1,2) [ ]
```

```
CLUSTER (1,3) [ 38 ]
```

```
CLUSTER (2,0) [ 36 44 45 ]
```

```
CLUSTER (2,1) [ 20 21 28 29 ]
```

```
CLUSTER (2,2) [ 22 23 30 31 ]
```

```
CLUSTER (2,3) [ 39 46 47 ]
```

### 6.4.1 Algorithm 4

```
for (SensorID = 1; SensorID <=NUMBEROFSENSOR;SensorID++){
lb:
    Sensor_X = DEVICE_POSITION(SensorID)->X - SinkNode_X;
    // Finding relative x-axis position from SINK Centre
    Sensor_Y = SinkNode_Y - DEVICE_POSITION(SensorID)->Y;
    // Finding relative y-axis position from SINK Centre

    Sensor_Sinknode_distance = fn_NetSim_Uilities_CalculateDistance
        (DEVICE_POSITION(NUMBEROFSENSOR + 1), DEVICE_POSITION(SensorID));
    for (i=0;i<NUMBEROFCONCENTRICCIRCLE+1;i++){
        if (Sensor_Sinknode_distance <= CONCENTRIC_DISTANCE[i]){
            Sensor_Angle = (180 * atan2(Sensor_Y,Sensor_X)) / PI;
            Arc = Sensor_Angle/angle;

            if (Arc >= 0){
                Arc = floor(Arc);
            }
            else{
```



## 7. Performance Analysis

In this section, we evaluate the performance of REBM via simulation in terms of distribution of energy consumption among the sensors, lifetime measurement in terms of data delivery and coverage, and number of Hello packets received by different sensors.

### 7.1 Simulation Environment

REBM model is simulated using network simulator Net\_Sim9.2. We have assumed that all the sensors have same transmission range and same sensing range. The sensing range varies from 5 to 35 m and the transmission range varies from  $r_t = 5m$  to 15m. Data packet length is assumed to be 30byte. Transmission power consumed in each packet transmission is  $10 \mu\text{J/bit}$ . Power consumed in receiving of a data packet is  $0.9 \mu\text{J/bit}$  and power consumed by a sensor in idle mode  $0.05 \mu\text{J/bit}$ . Energy consumed in sensing is  $1 \text{mJ/s}$ . Energy consumed in sleep mode of sensors is infinitesimally small and its value is  $1.5 \text{W/s}$ . Initial energy of each sensor is assumed to be  $1.5 \text{J}$  and transmission bit rate is 40kbps. Sensing field is assumed to be a circular region of radius  $R = 500\text{m}$ . We assume that width of last annulus is 50m. Sink is situated at the center of sensing field. The number of sensors deployed in sensing field is taken to be  $N = 1,500$ .

### 7.2 Annulus Formation

In the analysis of load in case of uniform data flow, we assume a large number of annuli in the field. Since transmission range of sensors is taken in the range from 25 to 35 m, we cannot take a large number of annulus in this case. However, radio transmission range of sensor changes with width of annulus. It is assumed that distribution of sensor in the sensing field follows Gaussian distribution. The main objective is to rationalize energy but at the same time to maintain full coverage. Therefore, there should be sufficient number of sensors in the outermost annulus to provide complete coverage. Let assume that number of sensor required for this purpose is  $k$ . Therefore, by following Gaussian distribution, it can be expressed as under.

$$\int_{R_{15}}^R \left( \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/2\sigma^2} \right) dx = k/N \quad (24)$$

Where,  $R_{15}$  and  $R$  are inner and outer radii of outermost annulus respectively. AFA is run with above simulation parameters. Result of this simulation is shown in Fig. 9 and 10.

In Fig. 9, the number of Hello packets received by different sensors is shown. About 35% of sensors received 4 Hello packet. Analytically the average value ought to be 3.78 for all sensors. About 30% of sensors received 3 Hello packets. As the sensors are deployed randomly, some of sensors are deployed randomly, some of sensors receive no Hello packet.

Figure 10 depicts the number of sensors in each annulus. By using Eq. (24) with for  $\mu = 0$  analytically we found that the number of sensors in outermost annulus should be 100 when 1500 sensors are considered. The result in Fig. 10 shows that there are 98 sensors in the outermost annulus. There is a marginal variation in the two results.

### 7.3 Connectivity Measurement

This section analyses the performance of CERA through simulation. We assumed that every sensor in each annulus generates data by following Poisson distribution with rate parameter 20 packets/s. The connectivity measurement is measured in term of fractions of packet transmitted successfully to sink. We call this fraction as data delivery ratio and it is define as:

$$Data\ Delivery\ Ratio = \frac{Number\ of\ packets\ fully\ received\ at\ sink}{Number\ of\ Packets\ Generated}$$

The result of simulations for data delivery ration are shown in Fig. 11. We observed that the delivery ratio increases exponentially with the increasing number of sensors. Data delivery ratio starts saturation after 0.8 for  $r=25$  m and after 0.9 for  $r=35$  m. With increasing transmission range, data delivery ratio increases rapidly and less number of sensors are required to achieve satisfactory ratio.

Figure 12 shows coverage number of hop counts that a packet to reach the sink. It is observed that with increasing number of sensor, the hop counts decreases rapidly. It is due to the fact that CERA first finds a sensor in lower annulus to forward the packet to the sink. If such a sensor is not found, it looks for a forwarding sensor in its own annulus. This increases the number of hops for a packet to reach the sink



## 7.4 Coverage Measurement

In this section the performance of CPSA is measured in terms of percentage of coverage obtained for varying number of sensors and different sensing ranges. Results in Fig. 13 show that higher level of coverage is obtained when sensors are not scheduled since sensors use maximum sensing range. However, with scheduling, the same level of coverage is obtained.

In Fig.14, shows the percentage of average number of sleeping sensor after scheduling. It is apparent from the figure that with increasing sensing range, percentage of sleeping sensors also increases. For the larger number of sensor, the percentage of sleeping sensors increases rapidly.

## 7.5 Distribution of Residual Energy

In this section, performance of REBM in terms of distribution of residual energy among sensors at the end of different simulation period has been measured. We have run simulation for total 200s. Distribution of residual energy is measured in four intervals of 50s each. It is assumed that sensors generate data by Poisson distribution with rate parameter of 20 packets per second. The simulation results are shown in Fig. 15. The results obtained from this simulation are compared with the model of energy balancing mode from the results, it is observed that the distribution of sensors for residual energy follows Gaussian curve. Average is taken to count the member of sensor with a specific energy value. At  $t = 50s$ , there are about 800 sensors with residual energy of 0.8000 Joules which is mean value of residual energy in REBM. In case of EBM, the spread of sensors around mean is faster than REBM. From this, we concluded that in EBM rationalization of energy regardless of simulation time is better than energy balance model.

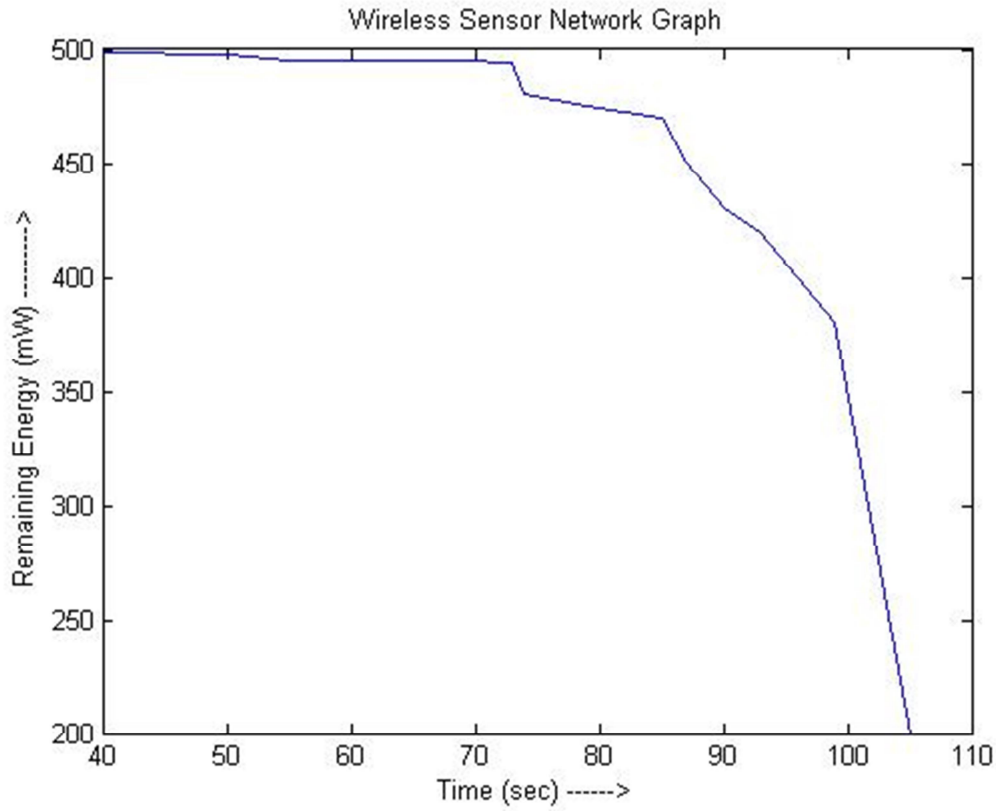


Figure 31: Remaining Energy for EBM

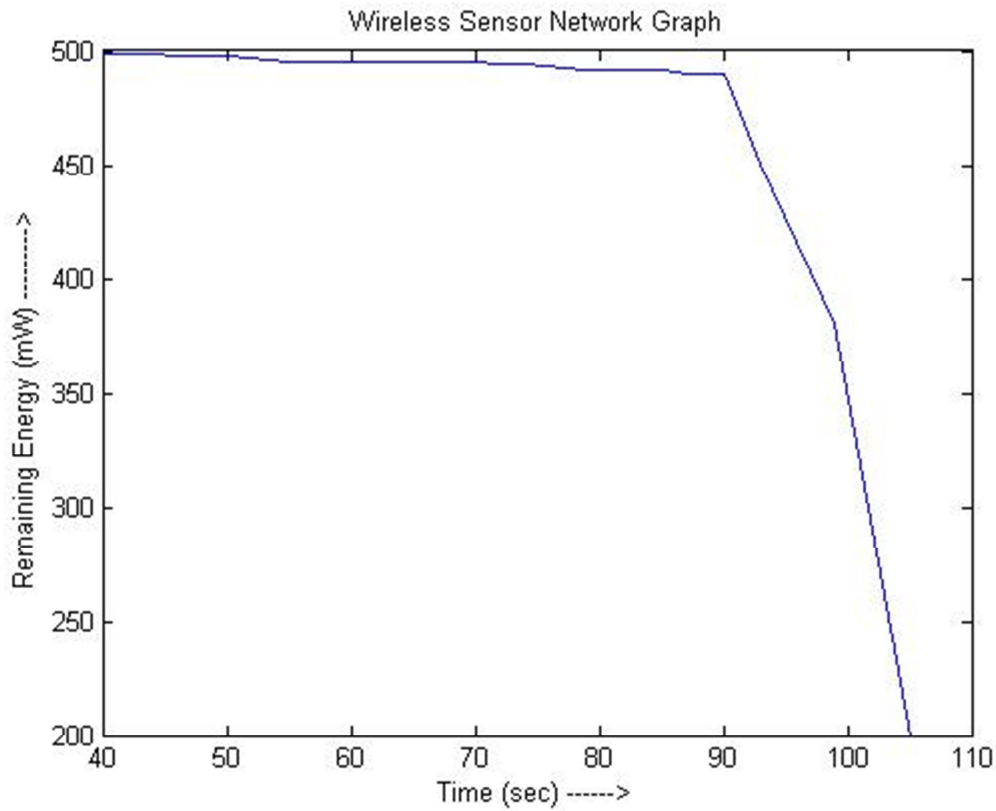


Figure 32: Remaining Energy for REBM

## 7.6 Lifetime Measurement

A sensor network assume to be alive for a duration to which it provide satisfactory service. In this work, lifetime of sensor network in terms of time duration for which satisfactory coverage and data delivery ratio achieved.

In Fig.16 shows simulation results for comparison of data delivery ratio among REBM and energy balance model. In simulation we have considered data delivery ratio above 80% as satisfactory level. It is quite evident from the figure that the lifetime of REBM is better than with other model. This can be attributed to the fact that in REBM, sensors near the sink

Remain alive for longer duration since redundant sensor are set off for a specific period by CPSA, and sensor with higher residual energy is selected as next hop and transmission range is controlled by CERA. In case of balance energy model, sensors near the sink exhaust their energy early since they bear excessive load of forwarding, and absence of energy efficient next hop selection and scheduling. For satisfactory data delivery ratio, the lifetime obtained using REBM is almost 25 percent more than energy balance model.

In Fig.17 simulation results of lifetime in terms of coverage provided for 280s is shown. In our simulation we have considered coverage above 90% as satisfactory level. REBM provides better satisfactory coverage for long duration in comparison to energy balance model. The reason behind this improvement is that CPSA sets off redundant sensors in sensors in the sensing field for a specific period. For satisfactory coverage REBM provides about 25 to 40 percent longer network lifetime as compared to that of energy balance model.

## 8. Conclusion

The reliable energy balance model result shows the reliable energy model is far better than the other models. Since region is divided in concentric circles and further each concentric circle in slices due to which number of hop count decreases so consumption of energy reduces. So due to less hop count data reaches to destination in less time with more reliability.

The Reliable energy model for wireless for sensor network presented in this project, shows the significance of energy rationalization for providing satisfactory coverage and enhancing lifetime of the network. Energy loss in transmission of "Hello" packets for formation of ring slice will be compensated by energy gain in route discovery is a proved as a novel contribution of the model through simulation count. As packets are only accepted by inner levels of ring slice so therefore reducing the no. of hop counts . So as hop count is reduced therefore less sensors will try to broadcast packet again and again. Therefore less energy is consumed in routing a packet and overall network lifetime is increased.

## Reference

- [1] *Energy balance model for lifetime maximization in randomly distributed wireless sensor networks.* by Upasana Dohare, D.K.Lobiyal, Sushil Kumar.
- [2] NetSim Standard ( <http://www.tetcos.com/> )
- [3] *Two-tiered wireless sensor networks – base station optimal positioning case study.* by R.K. Tripathi Y.N. Singh N.K. Verma
- [4] *A Decentralized Fuzzy C-Means-Based Energy-Efficient Routing* by Osama Mohød Alia
- [5] Matlab 2016 ( <https://www.mathworks.com/> )
- [6] *Balanced Unequal Clustering Algorithm under Noisy\_Environment.* by Ravendra Singh

