Chapter 4

Development Support Tools

This chapter describes an overview of Motes of the wireless sensor network (wsn). Behaviour of the Motes analysed using MATLAB True time Toolbox based on power utilization of Motes.

For the past decade, there has been rapid development and advancement in the communication and sensor technologies that results in the growth of a new, attractive and challenging research area – the wireless sensor network (WSN). A WSN, which typically consists of a large number of wireless sensor nodes formed in a network fashion, is deployed in environmental fields to serve various sensing and actuating applications. With the integration of sensing devices on the sensor nodes, the nodes have the abilities to perceive many types of physical parameters such as, light, humidity, vibration, etc. about the ambient conditions. In addition, the capability of wireless communication, small size and low power consumption enable sensor nodes to be deployed in different types of environment including terrestrial, underground and underwater. These properties facilitate the sensor nodes to operate in both stationary and mobile networks deployed for numerous applications, which include environmental remote sensing, medical healthcare monitoring, military surveillance, etc. For each of these application areas, the design and operation of the WSNs are different from conventional networks such as the internet. The network design must take into account of the specific applications. The nature of deployed environment must be considered. The limited of sensor nodes' resources such as memory, computational ability, communication bandwidth and energy source are the challenges in network design (discussed in chapter 2).

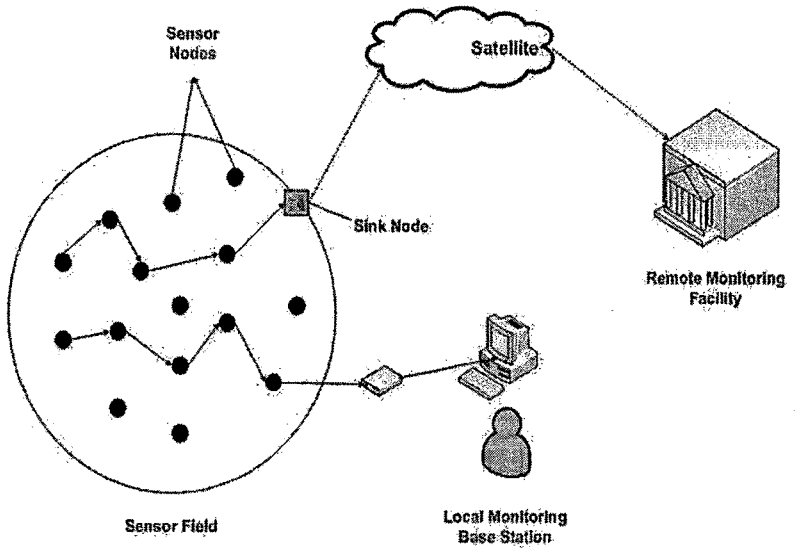

Figure 4.1: WSN [2]

WSN initially consists of small or large nodes called as sensor nodes [1]. These nodes are varying in size and totally depend on the size because different sizes of sensor nodes work efficiently in different fields. Wireless sensor networking have such sensor nodes which are specially designed in such a typical way that they have a microcontroller which controls the monitoring, a radio transceiver for generating radio waves, different type of wireless communicating devices and also equipped with an energy source such as battery. The entire network worked simultaneously by using different dimensions of sensors and worked on the phenomenon of multi routing algorithm which is also termed as wireless ad hoc networking. Each node has one or more sensors integrated on it. In addition to these sensors, a node is also equipped with a transmitter and a receiver. These transmitter and receiver are used for wireless communications with other nodes or directly with the gateway. The gateway is responsible for transmitting sensor data from the sensor patch to the remote base station that provides Wireless Ad-Hoc Network (WANET) connectivity and data logging through a local transit network. Finally, the data is available to researchers through a user interface. The scenario of Wireless Sensor Network is shown in Figure 4.1.

4.1 MOTES

Wireless Sensor Networks, also known as Sensory Networks or Sensitive Networks and abbreviated in English to WSN, consist of a series of small electronic devices that access the outdoor world by means of sensors. The name given to this type of device is "MOTE" [3], from the phrase "mote of dust", with the idea of conveying two main concepts in just one word: their small size and the idea that they can be placed anywhere. This concept gave rise to the formation of Dust Networks and to the nickname Smart Dust. Other ways of denoting a mote include Sensory Node or Sensor Node.

Sensor networks-based monitoring applications range from simple data gathering, to complex Internet-based information systems. Motes run the network embedded programs that mainly sleep, and occasionally acquire, communicate, store and process data. The other parts of a sensor node are the microcontroller and the battery (as the energy source). Battery-powered embedded systems, such as WSN motes, require low energy usage to extend system lifetime. WSN motes must power sensors, a processor, and a radio for wireless communication over long periods of time, and are therefore particularly sensitive to energy use. Recent techniques for reducing WSN energy consumption, such as aggregation, require additional computation to reduce the cost of sending data by minimizing radio data transmissions. Larger demands on the processor

will require more computational energy, but traditional energy reduction approaches, such as multi-core scaling with reduced frequency and voltage may prove heavy handed and ineffective for motes. Battery-powered embedded systems carefully manage energy consumption to maximize system lifetime. WSNs, made up of many "mote" devices, and are often designed to operate for months without intervention.

The most important consideration for a WSN is power consumption. While the concept of WSN looks practical and exciting on paper, if batteries are going to have to be changed constantly, widespread adoption will not occur. Therefore, when the sensor node is designed power consumption must be minimized.

4.2 MATLAB

MATLAB is a high performance language for technical computing. It integrates computation, visualization and programming and easy to use environment where problems and solutions are expressed in familiar mathematical notation. MATLAB is an interactive system while basic data element is an array that does not require dimensioning. This allows us to solve many technical computing problems, especially those with matrix vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C. The name MATLAB is an abbreviation of matrix laboratory [4].

MATLAB features a family of add-on application specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulations and many others. The MATLAB system when started, the MATLAB desktop appears, containing tools for managing files, variables and applications associated with MATLAB [5].

4.3 Simulink

Simulink is a software package for modeling, simulating and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two.

Systems can also be multirate i.e. have different parts that are sampled or updated at different rates [6]. For modeling, Simulink provides a graphical user interface (GUI) for building models as block diagrams, using click and drag mouse operations. It includes a comprehensive block library of sinks, sources, linear and nonlinear components and

connectors. We can also customize and create own blocks. For information on creating own blocks, like separate writing S-Functions. Models are hierarchical, so you can build models using both top down and bottom up approaches. We can view the system at a high level and then double click blocks to go down thorough the levels to see increasing level of model detail. This approach provides insight into how models organized and how its parts interact.

The model can be simulated using a choice of integration methods, either from the Simulink menus or by entering commands in the MATLAB Command window. The menus are particularly convenient of interactive work, while the command line approach is very useful for running a batch of simulations. For example using scopes and other display blocks, we can see the simulation results while simulation is running; the simulation results can be put in the MATLAB workspace for post processing and visualization [4].

4.4 Truetime Toolbox

Truetime is the Matlab/Simulink-based [7] simulator, which facilitates co-simulation of controller task execution in real-time kernels, network transmissions, and continuous plant dynamics. TrueTime [8] is a simulator for networked and embedded control systems that has been developed at Lund University since 1999.

TrueTime supports many network types (Wired: Ethernet, CAN, TDMA, FDMA, Round Robin, and switched Ethernet, and wireless networks: 802.11b WLAN and IEEE 802.15.4) and it is widely used to simulate wireless NCSs [8]. Besides the dynamic system simulation offered by Simulink, network node simulation includes simulation of real-time kernels. The user can write Matlab m-file functions that are scheduled and executed on a simulated CPU.

It is written in C++ MEX, and is an event-based simulation. External interrupts are implemented, and it is possible to write tasks as M-files or C++ functions. Another feature is the possibility to make the devices battery-powered and simulate the power drain. When installed, a separate block library, in addition to the standard Simulink library, becomes available. The TrueTime block library consists of four different blocks [7, 8] (referred in Figure 3.2).

The TrueTime Kernel block simulates the control mechanisms for the nodes in the network, i.e. it acts as the micro controller in the node. The TrueTime Network block simulates medium access and packet transmissions in a local area network. Six simple

models of networks are supported: CSMA/CD, CSMA/AMP, Round Robin, FDMA, TDMA and Switched Ethernet. The TrueTime Wireless Network block is similar to and works in the same way as the wired one. The TrueTime Battery block simulates a battery being drained and has only one parameter, the initial power. It uses simple integrator model, so it can be both charged and recharged. Because TrueTime is based on Simulink in Matlab, knowledge of Matlab programming and Simulink simulation is beneficial. The creation of models can be done by extensive use of graphical Simulink models or by writing Matlab code for different parts of the system. The TrueTime tool comes with a user manual that describes the functions of the models and how to use the built in library. TrueTime was selected as a simulation tool because of its simplicity and, admittedly, due to the familiarity of Matlab and limited knowledge of C++ by the authors. TrueTime is considered experimental software and can be downloaded for free from the TrueTime website [9].

The manual [10] describes the fundamental steps in the creation of a TRUETIME simulation. This include how to write the code that is executed during simulation, how to configure the kernel and network blocks, and what compilation that must be performed to get an executable simulation. The code functions for the tasks and the initialization commands may be written either as C++ functions or as Matlab M-files and both cases are described. [11] has discussed networked control in industrial applications and the possibility to simulate the control systems by TrueTime, the Networked Control Systems, their characteristics and possibilities, the wired and also the wireless systems.[12] uses TrueTime toolbox as a simple and easy way how to realize several network types. Also demonstrates how to setup simple TrueTime network control system with an explanation of its basic settings and parameters. In the last briefly compare simulation results of motor control system which uses different network types. [13] describes, a higher level simulation platform for WSN which is proposed based on the TrueTime toolbox. Relevant features include graphical representation of communication components, wireless communication and battery-driven operation. Special attention was paid to the 3D graphical interface, simulator interactivity and its extendibility. A TrueTime simulation model of the tunnel scenario is developed [14, 15]. The TrueTime simulator allows concurrent simulation of the physical robots and their environment, the software in the nodes, the radio communication, the network routing, and the ultra-sound navigation system.

4.5 Software Requirements

For the Matlab version, pre-compiled files are provided in the archive that is downloaded from the TRUETIME web site. The following compilers are currently supported (it may, of course, also work using other compilers):

- Visual Studio C++ 7.0 (for all supported Matlab versions) for Windows
- gcc, g++ - GNU project C and C++ Compiler for LINUX and UNIX

Before starting Matlab, set the environment variable TTKERNEL to point to the directory with the TRUETIME kernel files, $DIR/kernel. This is typically done in the following manner:

- Unix/Linux: export TTKERNEL=$DIR/kernel
- Windows: use Control Panel / System / Advanced / Environment Variables

Then add the following lines to your Matlab startup script. This will set up all necessary paths to the TRUETIME kernel files.

- addpath([getenv('TTKERNEL')])
- init_truetime;

Starting Matlab and issuing the command

```
>> truetime
```

From the Matlab prompt will now open the TRUETIME block library, as in Figure 3.2. (chapter3).

4.6 Compilation

Since the TRUETIME archive contains pre-compiled files, no compilation is required to run TRUETIME with the M-file API. However, TRUETIME also supports simulations written in C++ code, which then must be compiled. In this case, you first need to configure your C++ compiler in Matlab. This can be done by issuing the command

```
>> mex –setup
```

In the setup, make sure that you change from the Matlab default compiler to a proper C++ compiler. For more detailed instructions on how to compile individual simulations refer manual. [9, 10].

4.7 Simulation Setup [1]

In the simulation model the movements of dynamically moving motes using the built in functionality of Matlab is used. The example consists of three motes, with dynamics in the x and y directions modeled using simple integrators. The motes are sent out on a mission which consists of visiting a number of checkpoints seen as red marks in the animation window in Figure 4.3.

In the window, the transmission range of the motes can be seen as large partly transparent colored circles around the smaller colored motes. The checkpoints should be visited at least once by some node in the group. When the motes are able to communicate, they tell each other where they are heading and share information on which nodes that have been visited by the group. Some of this information is visible as printouts during the execution. When a checkpoint has been visited it changes color from green to red. The motes operate according to the following algorithm [7]:

1. Read new network messages containing information of visited nodes target of the sending node
2. If (someone has the same target as we do && we have the lowest priority), then change target
3. If (heading to a place that has already been visited), then change target
4. If (arrived at target) then paint the target green && change target
5. Send new network messages to other nodes

The simulation results were carried out using TrueTime toolbox of MATLAB [5, 6]. Motes with random topology and 9 nodes were simulated. The performance was evaluated for Signal Reach, Visiting Priority using variation in transmission power of each node [16].

4.7.1 Process to run simulation model

To run the simulation following commands execution require at the command prompt of Matlab after setting the current path in Matlab for kernel

```
>>addpath (getenv ('TTKERNEL'))
>> init_truetime
>> truetime
```

[1] **Published a paper: Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma "A Simulation Study of Behaviour of Wireless Motes With Reference To Parametric Variation" International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 1, Issue 2,pp:91-95 ISSN 2278 – 8875, August 2012.**

It will open the Truetime library, then open the model motes.mdl and init.m file. First execute init.m file then run the model motes.mdl file to visualize the topology animation and the results on the command prompt.

This simulation was carried out with following parameters shown in Table 4.1:

Parameters	Values
Transmit Power	-20 in dbm
Receiver Threshold	-48 in dbm
Path loss	3.5
Error Coding Threshold	0.03
Number Of Motes	3

Table 4.1 Simulation parameters

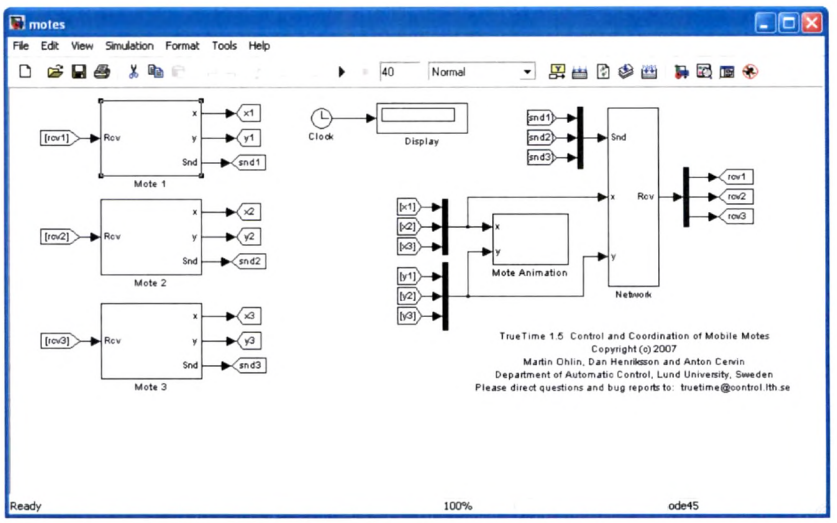

Figure 4.2: Simulink model in truetime

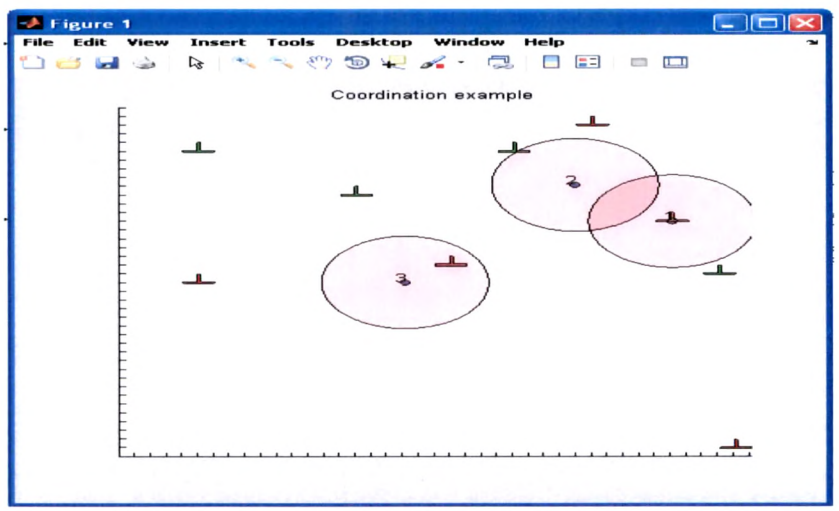

Figure 4.3: Random Network Topology

Figure 4.2 shows the Simulink model of Motes available in MATLAB True time and the random topology for wireless sensor network used for simulation is shown in Figure 4.3.

4.7.2 Simulation Results

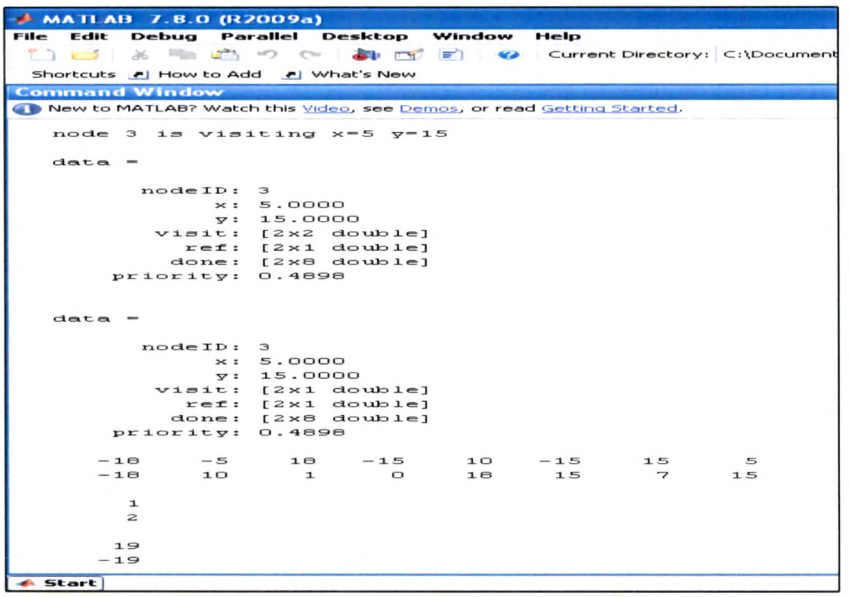

Figure 4.4: Results on Command window for MOTES Simulink model

Figure 4.4 depicts the result of simulation model in matlab command prompt which gives the information about which node visits the visiting point, with some specific priority. Also it shows the list of visited points, remaining visiting points and referred visiting point by the nodes.

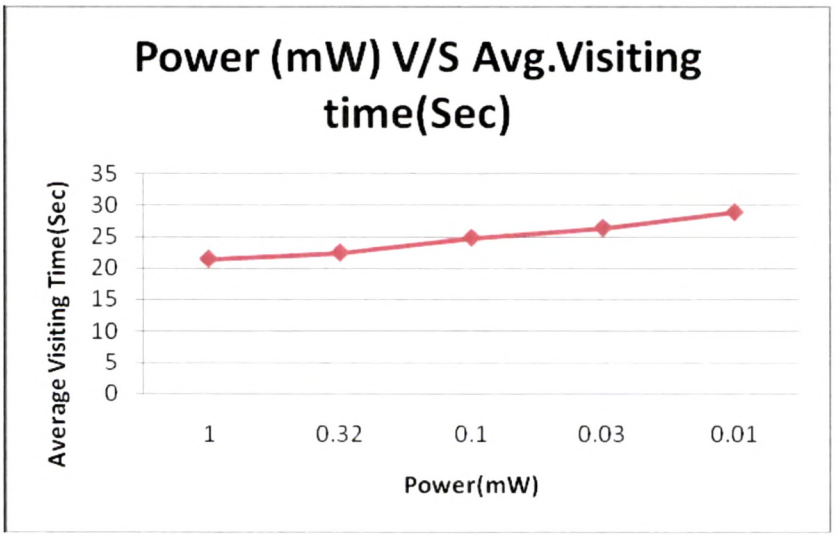

Figure 4.5: power (mW) V/S average visiting time(Sec)

Figure 4.5 shows the average visiting time taken by all the nodes to visit the visiting points in network. From this it can be observed that as the power decreases the average visiting time taken by the nodes increases.

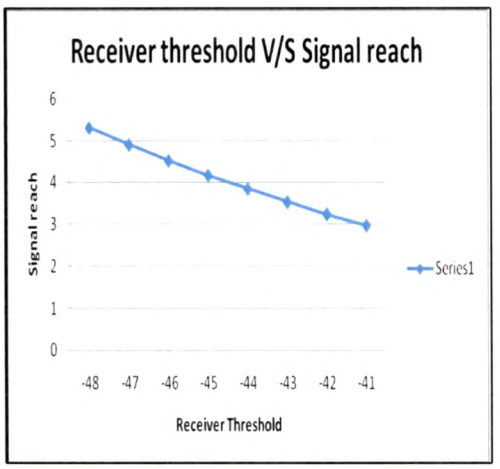

Figure 4.6: Receiver threshold level (dbm) V/S Signal Reach

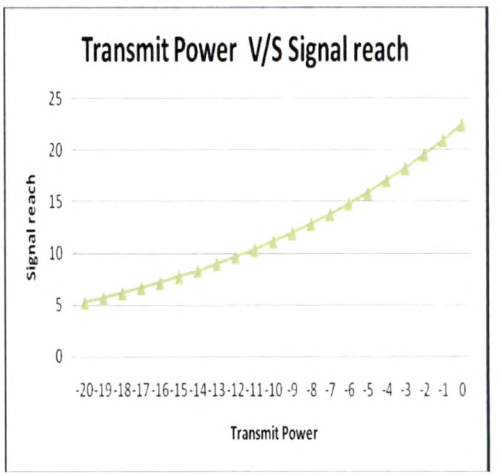

Figure 4.7: Power (dbm) V/S Signal Reach

Effect of changes in Received threshold level and transmission power on Signal Reach explained in Figure 4.6 and 4.7 respectively. As the receiver threshold level increases signal reach of the node decreases. It can be observed from figure 4.7 that as transmission power increases signal reach of the nodes also increases.

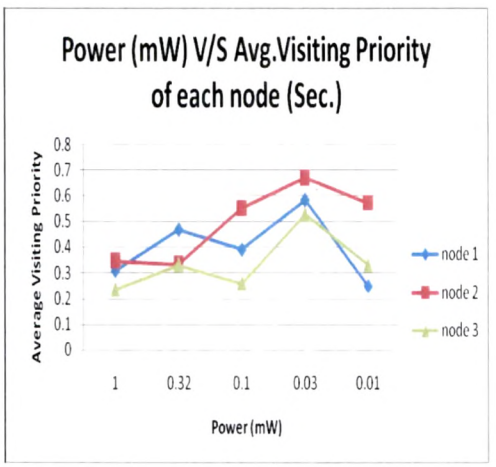

Figure 4.8: Power (mW) V/S Average Visiting Priority for each node

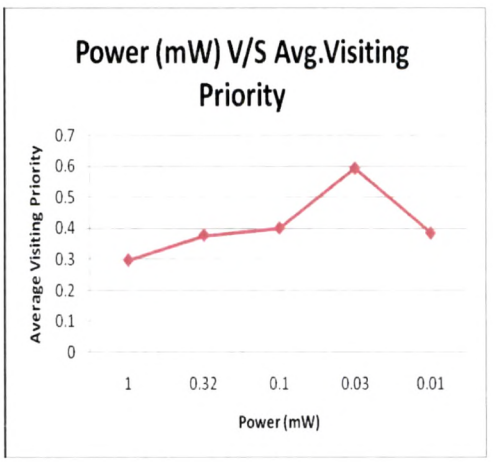

Figure 4.9: Power (mW) V/S Average Visiting Priority

Figure 4.8 depicts the change in average visiting priority of the node 1, 2, 3 with respect to change in transmission power of each node. It can be observed that as power decreases, priority also decreases for each node. Figure 4.9 shows the effect of change in power to the average visiting priority of the all nodes. As power decreases, the average priority for the nodes in network also reduces.

Summary

The power awareness issue is the primary concern within the domain of Wireless Sensor Networks (WSNs). Most power dissipation occurs during communication. From this simulation study it can be concluded that as the signal transmission power increases for each node, performance of WSN improves with signal reach. WSN consist of battery operated nodes and it is not suitable to increase the power to improve the performance of network. Suitable algorithm can be used to determine the optimized value of the transmission power for each node so that in less amount of power, performance can be improved. Also soft computing techniques can be applied to determine the optimized value of the transmission power.