CHAPTER 3

ENERGY MANAGEMENT IN WIRELESS SENSOR NETWORKS USING MULTI ROBOTS SCHEME FOR NODE DEPLOYMENT

3.1 INTRODUCTION

Energy Consumption is the primary issue in the case of wireless sensor networks. Several systems have been proposed to reduce the energy consumption. In such cases, the time to deploy the sensors is not taken into account. The random deployment of stationary nodes may result in an incompetent WSN, wherein some areas have a high density of nodes while others have a low density. The areas with high density increase hardware costs, computation time, and communication overheads, whereas areas with low density may raise the problems of coverage holes.

Other schemes discussed for node deployment using mobile sensors have been pointed out by Wang et al (2004). Mobile sensors first cooperatively compute their target locations based on their information in holes after an early phase of random deployment of stationary sensors and then move to target locations. Though the hardware costs cannot be reduced for areas that have a high density of stationary sensors deployed, another deployment choice is to use the robot to deploy static sensors.

The robot explores the environment and deploys a stationary sensor to the target location from time to time (Wang et al 2005). A single robot deployment algorithm achieves the purposes of energy management and full coverage with the deployment of fewer sensors Ching-Yung Chang et al
(2009). Node deployment using single robot has a few drawbacks. All the nodes are active until the deployment process is complete. It leads to more energy consumption. This problem is addressed in this chapter by proposing an energy efficient scheme in WSN using multi robots for node deployment.

The chapter is planned as follows. Section 3.1 consists of an introduction to the chapter, existing problem and advantages of multi-robots. Section 3.2 provides the details of multi robot scheme for node deployment. In section 3.3, simulation results of the proposed method are illustrated. Section 3.4 shows the comparison between the node deployment using single and multi-robots. Finally, the summary is given in section 3.5.

3.2 NODE DEPLOYMENT USING MULTI ROBOTS SCHEME

3.2.1 Need of attention for Node deployment

The major challenge in designing wireless sensor networks (WSNs) is the support of the functional requirements, such as data latency and the non-functional, such as data integrity, while coping up with the computation, energy and communication constraints. Careful node placement can be a very effective means of optimization for achieving the desired design goals. Categorizing the placement strategies into static and dynamic, depends on whether the optimization is performed at the time of deployment or while the network is operational, respectively.

Recent years have witnessed an increased interest in the use of wireless sensor networks (WSNs) in numerous applications such as forest monitoring, disaster management, space exploration, factory automation, secure installation, border protection, and battlefield surveillance (Akyildiz et al 2002). In these applications, miniaturized sensor nodes are deployed to operate autonomously in unattended environments. In addition to the ability to probe its surroundings,
each sensor has an onboard radio to be used for sending the collected data to a base-station either directly or over a multi-hop path.

For many set ups, it is envisioned that WSNs will consist of hundreds of nodes that operate on small batteries. A sensor stops working when it runs out of energy and thus a WSN may be structurally damaged if many sensors exhaust their onboard energy supply. Therefore, WSNs should be carefully managed. The base-station is deployed in the vicinity of the sensors to interface the network to remote command centers.

The bulk of research on WSNs has focused on the conservation of available energy in order to prolong the life of the network. Popular optimization techniques at the medium access control layer, collision avoidance, output power control, and minimizing idle listening time of radio receivers are a few samples of the proposed schemes (Naik et al 2005). One of the design optimization strategies is to deterministically place the sensor nodes in order to meet the desired performance goals. In such a case, the coverage of the monitored region can be ensured through careful planning of node densities and fields of view. Thus the network topology can be established at setup time.

However, in many WSNs applications, sensor deployment is random and a little control can be exerted in order to ensure coverage and yield uniform node density while achieving strongly connected network topology. Therefore, controlled placement is often pursued only for a selected subset of the employed nodes. This is done with the goal of structuring the network topology in a way that achieves the desired application requirements. In addition to coverage, the nodes' positions affect numerous network performance metrics such as energy consumption, delay and throughput. For example, large distances between nodes weaken the communication links, lower the throughput and increase energy consumption.
3.2.2 Strategies for Node Deployment

The following mobility patterns are suitable for Wireless Sensor Networks. The patterns are random walk, random waypoint, discrete Brownian motion, and extended Levy walk.

1) Random Walk: In this mobility model, a mobile node (MN) moves from its current location to a new location by randomly choosing a direction and speed from pre-defined ranges, [speed_{min}, speed_{max}] and [0, 2\pi] respectively. Each movement in this model occurs in a constant time interval T. At the end of each movement, a new direction and speed are calculated. Assume the speed is uniformly selected from the pre-defined range.

2) Random Waypoint: In this mobility model, a MN moves from its current location to a new location, which is randomly selected in an area. The MN then travels toward the newly chosen destination at the speed selected from a pre-defined range. Let the new location uniformly distributed in the circular region centered at the current location with radius r \leq R_{max}.

3) Discrete Brownian motion: In this mobility model, times are slotted with constant slot interval T. At the beginning of each slot i, a MN selects the destination (X_{i+1}, Y_{i+1}) based on current location, (X_i, Y_i), i.e., X_{i+1} = X_i + \eta W_1, Y_{i+1} = Y_i + \eta W_2, where W_1 and W_2 follows normal distribution. \eta is the variance of Brownian motion.

4) Extended Levy Walk: Levy walk is a variant of random walk in which the traveled distance of each movement is distributed according to a heavy-tailed distribution, which follows a power law form F(d) = 1 - (d \alpha). In this model, the upper bound of the traveled distance can be only set as infinity.
In this research, random walk mobility patterns used for both Mobile robots and Mobile relay. Because, In this mobility model, a mobile node (MN) moves from its current location to a new location by randomly choosing a direction and speed from pre-defined ranges, $[\text{speed}_{\text{min}}, \text{speed}_{\text{max}}]$ and $[0, 2\pi]$ respectively. Each movement in this model occurs in a constant time interval $T$. At the end of each movement, a new direction and speed are calculated. Assume the speed is uniformly selected from the pre-defined range. The traveled distance during each time interval follows

$$\frac{(d - d_{\text{min}})}{(d_{\text{max}} - d_{\text{min}})}$$

where $d_{\text{max}} = \text{speed}_{\text{max}} \times T$ and $d_{\text{min}} = \text{speed}_{\text{min}} \times T$

Speeds of the mobile robots are 10m/s. This part of the work concentrates on categorizing the various strategies for positioning nodes in WSNs. Categorizing the placement strategies into static and dynamic depends on whether the optimization is performed at the time of deployment or while the network is operational, respectively.

Analyze the performance efficiency of deployment algorithm in a non-obstacle environment in terms of the number of deployed sensors. Assume the sensing range $r_s$. For simplicity of analysis, the maximal hexagon cell covered by the sensor is used to represent the sensing range of the sensor. The optimal deployment of an area is the one that is deployed with the minimal number of sensors but achieves full coverage purpose. The number of sensors deployed in an optimal deployment equals to the number of hexagon partitions in the area.

The area of each hexagon can be derived by expression (1). This hexagon can be split into 6 triangles which are equilateral. Since side of each triangle is the same so area is also the same.
Area of Hexagon = 6 x area of 1 triangle

\[
\text{Area} = 6 \times (\sqrt{3}/4) r_s^2 = (3\sqrt{3})/2 r_s^2 \quad \text{-------------------------- (1)}
\]

Let L and W denote the length and width of the monitoring region, respectively. The ideal number of sensor nodes deployed in the monitoring region can be derived by (2).

\[
N_{\text{ideal}}(W, L) = \left[ \frac{W \times L}{\text{Area}} \right] = \left[ \frac{W \times L}{(3\sqrt{3})/2 r_s^2} \right] = \left[ \frac{2\sqrt{3}}{9} r_s^2 \right] \times WL \quad \text{------ (2)}
\]

In the real scenario, the robot may require to deploy one more sensor as it encounters the boundary of the monitoring region. Therefore, the robot requires deploying more sensors than that in the ideal case. Expression (3) evaluates the number of sensors deployed by applying the deploying mechanism in the worst case.

\[
N_{\text{worst}}(W, L) = \left[ \frac{W \times L}{\text{Area}} \right] + 2 \times \left[ \frac{(W/2) + L}{WL} \right] = \left[ \frac{2\sqrt{3}}{9} r_s^2 \right] \times WL + 2 \times [(W/2) + L] \quad \text{----- (3)}
\]

In considering the average case, rather than the worst case, the number of sensors deployed nearby the boundary highly depends on the distance between the latest deployed sensor and the boundary.

Let H = [L/height of a hexagon cell] and P = [W/ height of a hexagon cell]

Where H = Average Height P = Average Width

Since the probabilities that each row and each column require to deploy one more sensor are \( \frac{1}{2} \) and \( \frac{1}{3} \), respectively, expression (4) reflects the average number of deployed sensors by applying deployment algorithm.

\[
N_{\text{approximate}}(W, L) = \left[ \frac{W \times L}{\text{Area}} \right] + 2 \times [(1/2)*H] + [(1/3)*P] = \left[ \frac{2\sqrt{3}}{9} r_s^2 \right] \times WL + [H] + [2/3]*p \quad \text{-- (4)}
\]
Based on the above mathematical model, the proposed methods are verified. The above model is focused for node deployment using single robot. It can be applied for multi robot also.

Compared with random deployment, step-wise deployment of static sensors in a specific region can give full sensing coverage with fewer sensors. Previously the research proposed by Batalin et al (2003) assumed that the robot is equipped with a compass and is able to detect obstacles. Each sensor had a communication range \( r_c \) and a sensing range \( r_s \). To guide the robot’s movement, each deployed sensor maintained time duration for each direction, whether south, east, north, or west, that the robot did not visit. The longer the time length, the higher the priority of the direction is.

When the robot intends to make a movement decision it communicates with the closest deployed sensor, queries the time length of each direction, and then selects one direction with the highest priority to be the direction of its next movement. Although the robot-deployment algorithm developed likely achieves the purpose of full coverage and network connectivity, the next movement of the robot is guided by only one sensor, resulting in its taking a long time to achieve full coverage. It requires more sensors due to a big overlapping area.
As shown in Figure 3.1(a), assume that sensors $s_k$ and $s_j$ were previously deployed in the monitor.
ing area and sensor $s_i$ is the most recently deployed sensor. According to the guiding of sensor $s_p$, the robot will move east and deploy sensor $s_{i+1}$, resulting in a hole this time even though sensor $s_k$ is in the communication range of the robot. A sensor will be deployed to the hole after a long period. Figure 3.1(b) shows another example. Assume that the distance between $s_i$ and $s_k$ is smaller than $r_s$. The robot only guided by $s_i$ may deploy a sensor $s_{i+1}$ in the location that overlaps a lot of sensing regions of $s_k$ and $s_j$, requiring more sensors to be deployed on the whole target region. This work develops an efficient robot-deployment algorithm.

The robot will be guided by $s_i$ and $s_k$ at the same time to achieve full coverage and network connectivity purposes using fewer deployed sensors. The improved deployment in this case is shown in Figure 3.1(c). The overlapping area among the sensing ranges of $s_i$, $s_{i+1}$, and $s_k$ is minimal, achieving full coverage by using fewer sensors. In addition, the algorithm does not consider energy conservation. Since all the sensors have no knowledge of when the robot will revisit them, all the sensors should be active and therefore consume significant energy. Energy conservation should be taken into consideration in developing the robot-deployment algorithm wherein most deployed sensors may stay in sleep mode during the robot-deployment process.

This work aims at developing a robot-deployment algorithm that has the following characteristics. The robot deploys sensors in a way that fewer sensors are deployed but are likely to achieve full coverage. As for power conservation, most deployed sensors can stay in the sleep mode to conserve energy. In addition, the developed deployment algorithm can resist obstacles so
that fewer sensors need to be deployed to achieve full sensing coverage even if there are obstacles in the monitoring area.

### 3.2.3 Snake movement policy

Snake Movement policy for the node deployment using Robots has been discussed by Ching-Yung Chang et al (2007).

#### 3.2.3.1 Network Environment

This chapter considers a collection of robots that carry limited static sensor nodes and embedded with a compass through which the robot is aware of its moving direction. Initially, the robot is assumed to be located at the left corner of the monitoring region. Let $r_c$ and $r_s$ denote the communication and sensing ranges, respectively. Herein, assume $r_c$ is larger than $3r_s$.

#### 3.2.3.2 Movement Policy

An optimal robot deployment refers to the deployment that the robot deploys minimal number of sensors but achieves full-coverage purpose. To achieve the optimal deployment, the overlapped sensing region of neighboring sensor nodes should be strictly controlled. Figure 3.2 illustrates the basic requirement for optimal deployment. Let nodes A, B, and C be the three neighboring sensors.

The optimal deployment can be reached if the three sensor nodes intersect with each other at one point. In this situation, the distance of combination of A, B, and C exactly equals to $\sqrt{3}r_s$. Based on this, the deployment policy informs the robot to deploy a sensor at every $\sqrt{3}r_s$. 

In addition to the deployment policy, a snake-like movement policy is employed for the robot's movement. Figure 3.3 depicts the snake-like movement where the robot deploys a sensor every $\sqrt{3}r_s$ distance.

**Figure 3.2 Optimal deployments of the sensors**

**Figure 3.3 Snake like movement deployment**
The robot will deploy a sensor node after each movement of distance $\sqrt{3} r_s$. To achieve the optimal deployment, the movement of robot should be one of the six legal patterns as shown in Figure 3.4.

$$d_1=\sqrt{3} r_s \quad d_2= (3/2) r_s \quad d_3= \sqrt{3}/2 \ r_s$$

**Figure 3.4 Legal patterns for basic movement.**

The six types of basic movement are referred to as the legal patterns for basic movement. Types 1 and 2 are used when the robot moves towards east and west directions, respectively. As shown in Figure 3.2, the sensor nodes deployed on the $i^{th}$ row are located on the perpendicular bisector of two neighboring sensors deployed on the $(i-1)^{th}$ row. As the robot encounters a boundary or obstacle, it should deploy the sensor on the next row. That is, the robot should move towards the south. Type 3 will be used when the robot moves in the west direction but encounters left boundary or obstacle. In this case, the robot moves towards the south for a distance of $(3/2) r_s$, then moves toward east for a distance of $(\sqrt{3}/2) r_s$.

Similarly, type 4 is used when the robot moves toward east but encounters right boundary or obstacle. To overcome the unpredicted obstacles, sometimes the robot would move towards the north direction. Types 5 and 6 are used when the robot tries to overcome obstacles and moves towards north direction.
3.2.4 Robot Deployment

In robot deployment, the robot stays in one of the two possible states: east and west. This is called snake like deployment. The east and west states denote that the robot is currently moving toward east and west directions, respectively. Each state has two movement direction options with different priorities to guide the robot to the promising direction of the next movement as shown in Table 3.1. In both east and west states, the preferred direction 1 has a higher priority which enables the robot to move along east and west directions, respectively. The preferred direction 2 has a lower priority and it will be applied in case if the movement in the preferred direction 1 is a failure. The preferred direction 2 will guide the robot to move towards south. As soon as the prefer direction 2 has been applied, the robot’s state should be changed.

<table>
<thead>
<tr>
<th>States</th>
<th>Prefer Direction 1</th>
<th>Prefer direction 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>(Type 1)</td>
<td>(Type 1)</td>
</tr>
<tr>
<td>West</td>
<td>(Type 2)</td>
<td>(Type 3)</td>
</tr>
</tbody>
</table>

3.2.4.1 Vertical Movement Policy

In this method, the entire region is divided into small regions. For each small region a robot is assigned to deploy the sensors. Here, the robot is made to deploy the sensors by moving vertically (from top to bottom or bottom to top), so that the deployment process is done in all the regions simultaneously. This reduces the time of deploying the sensors and also reduces the power consumption in the previous deployment methods.

3.2.4.2 Horizontal movement policy
In this method also, the entire region is divided into small regions. For each small region a robot is assigned to deploy the sensors. The only difference in this method is that the robot is made to deploy the sensors by moving horizontally.

3.2.5 Obstacle-free robot deployment

3.2.5.1 Impact of obstacles

This subsection considers additionally the existence of obstacles and develops an obstacle-free snake-like deployment mechanism.

3.2.5.2 Obstacle handling rules

Assume that the robot stays in the east state. The robot repeatedly applies type 1 movement and hence moves in the east direction. As the robot encounters the right boundary, it checks prefer direction 2 according to the simple snake-like movement mechanism and applies type 4 movement. Therefore, the robot moves in south direction for a distance \((3/2)r_s\) firstly. Then it moves in the west direction for a distance \((\sqrt{3}/2)r_s\). However, moving in this way will result that there is no opportunity for the robot to visit the north and west directions to redeploy sensors in the hole that existed. In order to overcome the obstacles, the robot should check whether or not there exists any sensing hole in the north or west directions. Therefore, the movement towards the north and west directions should be prior to the current prefer direction1.

Table 3.2 lists the check directions for the robot to further check if the check direction contains any sensing hole. Prior moving in prefer direction 1, the robot will check the check direction first and tries to move in the check direction. In case there exist sensors deployed in the checked direction, the
robot will then apply the simple snake-like movement scheme and utilize Table 3.1 to decide the next movement direction.

There are two check directions for each state. In case if the robot stays in the east state, it first checks the check direction 1. In case if there is no sensor deployed in the west direction, the next movement direction will be west. Otherwise, the robot will check the check direction 2. If the north direction did not deploy any sensor, the robot will move in the north direction in the next movement. If, fortunately, the hole does not exist in the two check directions, the robot will further apply the simple snake-like movement scheme which utilizes Table 3.1 to determine the next movement direction of the robot. It stays in West state which is similar to that in East state.

As the robot stays in the East state and visits location a, it checks check direction 1 (West). Since there is a deployed sensor in the West direction, the movement in check direction 1 is a failure. The robot further checks the check direction 2 (North) and finds that there is a hole in the north direction. Then the robot moves in north direction and uses type 5 movement which moves in north direction first and then west direction.

As the robot arrives at location b, it checks check direction 1 and moves in west direction. Following the rules described above, the robot will continue to move towards the west direction until it arrives at location d and again the robot encounters the obstacle. Since the movement in check direction 1 (West) is a failure, the robot then checks check direction 2 and moves in north direction accordingly. During the movements from locations d to c, the movement in check direction 1 will be a failure and therefore the robot moves according to check direction 2.
By checking check direction 2 as shown in Table 3.2, the robot uses types 5 and 6 movement patterns alternatively and moves from locations d to c. The movements after location c will be based on simple snake like movement mechanism since both check direction 1 and check direction 2 are failures. Finally, the robot overcomes the impact of obstacles and achieves the goal of full coverage deployment.

Table 3.2 Check directions for overcoming the obstacle.

<table>
<thead>
<tr>
<th>States</th>
<th>Check Direction 1</th>
<th>Check Direction 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td>(Type 2)</td>
<td>(Type 5)</td>
</tr>
<tr>
<td>West</td>
<td>(Type 1)</td>
<td>(Type 5)</td>
</tr>
</tbody>
</table>

Table 3.3 Obstacle-Free Snake-Like Movement Rule.

<table>
<thead>
<tr>
<th>States</th>
<th>Check Direction 1</th>
<th>Check Direction 2</th>
<th>Prefer Direction 1</th>
<th>Prefer Direction 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>East</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>West</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Tables 3.3 summarize the check directions and prefer directions in a priority order. As shown in Table 3.3, the robot should select one of the six types of movement as the movement pattern is according to the priority of each movement type listed in Table 3.3.

In general, if the robot stays in the East state, the six movement types have the following priorities:

Type 2 > Type 5 > Type 6 > Type 1 > Type 3 > Type 4
On the other hand, if the robot stays in the West state, the six movement types have the following priorities:

Type 1 > Type 6 > Type 5 > Type 2 > Type 4 > Type 3

Note that if the robot fails to move in a certain direction with higher priority, the robot will try the next higher priority until there is a successful movement. The following four rules summarize the above mentioned obstacle handling algorithm. The movement direction is said to be failure if there exists a deployed sensor, obstacle, or boundary of the monitoring region in that direction.

The robot moves and deploys sensor nodes according to the following four rules in order. Rules 1 and 2 are mainly designed for handling obstacles whereas Rules 3 and 4 are designed for snake-like movement in the environment without obstacle. The robot checks the rules from rule 1 to rule 4 in order and executes one of the four rules to select a direction for the next movement. Then the robot determines the next moving direction by checking the following four rules again.

/* Rules 1 and 2 are designed for overcoming obstacle */

Rule 1: The robot checks check direction 1 (which is the opposite direction to prefer direction 1) for possible movement. If ‘the try’ in this direction is a failure, the robot executes the next rule. Otherwise, the robot will move towards the check direction 1 for $\sqrt{3}r_s$ distance.

Rule 2: The robot tries to move in check direction 2. If ‘the try’ in this direction is a failure, the robot checks Rule 3 subsequently.
Otherwise, the robot moves toward the check direction 2 for \((3/2)r_s\) distance.

/* Rules 3 and 4 are designed for snake-like movement */

Rule 3: The robot checks prefer direction 1 for possible movement. If ‘the try’ in this direction is a failure, the robot executes Rule 4. Otherwise, the robot moves toward the prefer direction 1 for \(\sqrt{3}r_s\) distance.

Rule 4: The robot checks prefer direction 2 for possible movement. If the try in this direction is a failure then the deployment is not terminated. The robot will go back to the location of previous deployed sensor and checks the four rules again in order. Otherwise, the robot moves toward the check direction 2 for \((3/2)r_s\) distance.

A region without any deployed sensor can be treated as a coverage hole without sensing capability. Assume that the robot stays in East state. Applying the simple snake-like movement (Rules 3 and 4), the robot will move from west to East until it encounters the right boundary and then from north to south direction. Therefore, the most difficult for robot deployment is that there exists obstacle so that a hole appeared in the west or north directions.

However, if there is a hole in the west or north directions, the robot may apply Rules 1 and 2; it examines the check directions including west and north directions and then moves back toward the west and north directions to deploy the sensors in the whole region. Therefore, the four rules presented herein can overcome the existence of unpredicted obstacle and achieve the purpose of full coverage deployment. As the obstacle-free snake-like deployment algorithm involves the consideration of two check directions,
the constraint that the robot should initially start its movement at left up corner could be released.

Even though the robot initially starts its movement at the central location of the monitoring region, the robot may apply Rules 1 and 2 to move toward west and north directions to deploy the sensors and then moves East and South to achieve the purpose of full coverage deployment. The state of robot depends on the initial location. If the robot starts its movement in east (or west) boundary of the monitoring region, the robot will stay in east (or west) state.

### 3.3 SIMULATION RESULTS

Single and Multi Robot deployment schemes adopted for deployment of sensor nodes have been analyzed. A Network simulator (NS-2) is used to evaluate the performance of the networks. Simulation setups are the sensor model consists of:

- Link layer
- MAC layer with the 802.11 type
- Interface queue with the DropTail/PriQueue
- Interface queue length with maximum 50 packets in the interface queue
- Antenna with the Omni-directed Antenna
- Propagation model with the TwoRayGround
- Network interface with the WirelessPhy
- Channel with the SENSOR and the WIRELESS channels
- Routing protocol with DSDV

The performance of the algorithm is evaluated in different simulation scenarios. The performance of the algorithm is also compared to an existing single robot scheme. A base simulation setting was created and
several scenarios were obtained from it to run experiments. In the setting, 100 mobile nodes are randomly placed in a rectangular area of 1500m X 300m to form a wireless sensor network. The nodes are made to move with a maximum velocity of 10m/s. In this model, each node at the start of the simulation remains at a particular location for a certain period of time (i.e pause time seconds). The node travels towards the new destination and upon arrival it pauses for the specified time period and then proceeds as previously described. It repeats this process for the duration of the simulation. Each simulation runs for 900 seconds totally. The radio propagation range of the nodes is 250 meters and the data rate is 2Mbits/s. At the Mac layer we use the 802.11 protocol and at the physical layer, the free space signal propagation model is used.

Simulation results are shown in Tables 3.4 (vertical snake movement policy) and 3.5 (horizontal snake movement policy) and their comparison is shown in Figures 3.5 and 3.7. For the simulation, 100 nodes are used and the energy consumption measured is in terms of Joule. The single robot scheme has been compared with the multi robot scheme and the result has been tabulated.

Figures 3.6 and 3.8 shows the energy consumption by the single robot scheme and multi robot scheme. They are compared for various node densities. The comparison of energy consumption shows that the proposed multi robot deployment of sensor node has less energy consumption than the single robot deployment of sensor node (both vertical and horizontal snake movement policy). Multi robot deployment of sensor node has 4% less energy consumption when compared to single robot deployment of sensor node.

**Table 3.4 Simulation results (Vertical snake movement policy)**
<table>
<thead>
<tr>
<th>Sl. No</th>
<th>No. of Nodes</th>
<th>Energy consumption in Joule</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Existing single robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vertical snake movement</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed multi robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>vertical snake movement</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>68.10</td>
<td></td>
<td>66.60</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>69.80</td>
<td></td>
<td>68.80</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>74.60</td>
<td></td>
<td>70.60</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>78.20</td>
<td></td>
<td>75.60</td>
</tr>
<tr>
<td>5</td>
<td>50</td>
<td>87.80</td>
<td></td>
<td>78.60</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>89.90</td>
<td></td>
<td>85.90</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>93.80</td>
<td></td>
<td>88.80</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>96.60</td>
<td></td>
<td>93.80</td>
</tr>
<tr>
<td>9</td>
<td>90</td>
<td>97.90</td>
<td></td>
<td>95.80</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>98.60</td>
<td></td>
<td>97.80</td>
</tr>
</tbody>
</table>

Table 3.5 Simulation results (horizontal snake movement policy)

<table>
<thead>
<tr>
<th>Sl. No</th>
<th>No. of Nodes</th>
<th>Energy Consumption in Joule</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Existing single robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>horizontal snake</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>movement policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proposed multi robot</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>horizontal snake</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>movement policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>74.20</td>
<td></td>
<td>68.20</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>81.00</td>
<td></td>
<td>73.80</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>86.20</td>
<td></td>
<td>80.60</td>
</tr>
<tr>
<td>4</td>
<td>40</td>
<td>89.20</td>
<td></td>
<td>85.60</td>
</tr>
</tbody>
</table>
3.3.1 Comparison of Node Deployment using Single Robot and Node Deployment using Multi Robot

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>50</td>
<td>93.00</td>
<td>89.80</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>93.80</td>
<td>92.00</td>
</tr>
<tr>
<td>7</td>
<td>70</td>
<td>95.80</td>
<td>93.80</td>
</tr>
<tr>
<td>8</td>
<td>80</td>
<td>96.10</td>
<td>95.40</td>
</tr>
<tr>
<td>9</td>
<td>90</td>
<td>98.60</td>
<td>96.40</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>99.60</td>
<td>97.80</td>
</tr>
</tbody>
</table>

Figure 3.5 Comparison between number of nodes and energy consumption (vertical snake movement policy)
Figure 3.6 Simulation Results (Vertical Snake Movement Policy - (Number of Nodes and Energy Consumption))

Figure 3.7 Comparision between number of nodes and energy consumption (horizontal snake movement policy)
3.4 SUMMARY

It has been shown so far that the deployment and use of many robots instead of a single robot and the placement policy used in case of multi robots are more effective than that of the previous systems. The data loss is minimized. At the same time, the number of sensors used is also reduced. In addition, the overlapping of sensing regions is reduced which leads to the reduction in the number of sensors used in the system. It leads to less energy consumption than that was consumed in the previous systems. The deployment of sensors is faster, which eventually leads to the longevity of sensors. The deployment cost incurred for using many robots instead of a single robot is negligible when compared to the energy conservation benefit of this multi robot deployment algorithm. However, a limitation of this scheme is that the deployment cost incurred for using many robots is high.