Dynamic Point Coverage in Wireless Sensor Networks: A Learning Automata Approach

M. Esnaashari[1], M. R. Meybodi[2]

[1] Soft Computing Laboratory, Computer Engineering and Information Technology Department
Amirkabir University of Technology, Tehran, Iran
Esnaashari@aut.ac.ir

[2] Soft Computing Laboratory, Computer Engineering and Information Technology Department
Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

Abstract

Point coverage in wireless sensor networks is the problem of detecting some stationary or moving target points in the area of the network using as little sensor nodes as possible. This can be accomplished using different deployment strategies, rearrangement of nodes using a moving strategy, or designing a suitable schedule for making nodes on and off in such a way that in each time slice, only nodes which can sense the target points in that period are on. With dynamic point coverage, we refer to the problem of point coverage, where target points are moving, and hence rearrangement or scheduling should be dynamically adapted to the changing position of targets. In this paper, we propose a novel method for the problem of dynamic point coverage in wireless sensor networks using learning automata. Each node is equipped with a learning automaton which will learn (schedule) the proper on and off times of that node based on the movement nature of a single moving target.

Keywords

Dynamic Point Coverage, Wireless Sensor Network, Learning Automata, Scheduling

1. Introduction

One major problem in the area of sensor networks is the coverage problem. This problem deals with the ability of the network to cover a certain area or some certain events. Coverage problem is classified into three different types [1]:

- Area coverage: The most studied coverage problem in which the main objective is to cover (monitor) an area is called area coverage.
- Point coverage: The objective of point coverage problem is to cover a set of stationary or moving points.
- Barrier coverage: Barrier coverage can be considered as the coverage with the goal of minimizing the probability of undetected penetration through the barrier (sensor network).

In this paper, we focus on the problem of point coverage. As mentioned earlier, point coverage is the problem of detecting some stationary or moving target points in the area of sensor network using as little sensor nodes as possible. This problem can be addressed in many different ways. One approach is to design a deployment strategy which can best address the criterion of minimum number of required nodes [2, 3, 4, 5, 6, 7]. This solution is static in its nature, since it deals only with the deployment of the network, but has nothing to do with changes (which are commonly occurred in sensor networks due to high probability of sensor failures) throughout the lifetime of the network. Another more dynamic solution to the point coverage problem is the rearrangement of nodes assuming movement ability for them [8, 9, 10, 11, 12]. In this approach, nodes are first deployed randomly around the target points. After this initial random positioning, each node tries to find its best position according to the position of target points and other nodes around it. It is straight forward that using this approach, dynamic changes in the topology of the network or position of the targets can be adaptively addressed. One major problem with this solution is the high overhead of controlling packets which are required to control the position of each node according to the position of its neighbors. This can lead to fast energy exhaustion of nodes, and hence shortening the lifetime of the network. Many researchers try to overcome the point coverage problem by designing a suitable scheduling strategy for making nodes on and off in such a way that in each time slice, only nodes which can sense the target points in that period are on [13, 14, 15, 16, 17, 18, 19, 20]. In this method, many nodes are scattered randomly throughout the filed, and the criterion of minimum number of required nodes addressed by the scheduling mechanism which tries to minimize the number of nodes being in on state according to their ability to sense the target points. In terms of dynamicity, this solution can

deal with changes occurred in the topology of the network and position of target points using a dynamic scheduling strategy. In terms of overhead and energy consumption, this method best fit the sensor networks, since in each time slice, only nodes which can sense the target points in that period are on, and rest of the nodes are in off state, reserving their energy.

Point coverage problem with moving target points is hereafter referred to as dynamic point coverage problem.

Centralized scheduling algorithms are suitable only for stationary targets or moving targets with known and static movement paths. For targets with unknown and dynamically changing movement paths, decentralized and more flexible scheduling algorithms are required. To our best knowledge, in all works done in this area, some sort of notification messages are exchanged between nodes currently in on state, monitoring the target points, and their neighbor nodes which are currently in off state, but with high probability should become on, since target points are moving towards them. This has two drawbacks; one is the overhead of these notification messages and the other is that nodes in off state should have the ability to receive these messages, and hence they cannot power off their receiving antenna. This means that these nodes just switch off their processing unit, but their communication units are in idle state. According to [21], energy consumption of a sensor node in receiving and idle states is nearly equal to the energy consumed during transmission state.

In this paper, we propose a novel method for addressing the problem of dynamic point coverage in wireless sensor networks using learning automata. Each node is equipped with a learning automaton which learns (schedules) the proper on and off times of that node based on the movement nature of the target points. This solution is a dynamic scheduling solution to this problem which not only has the advantages of scheduling methods, but also addresses the two shortcomings of these methods mentioned earlier. This is because in this method, each node (or better the automaton of each node) learns its best on and off schedules using only the information of the moving targets passing through its sensing area, and hence no notification messages exchanged between on and off nodes. To simplify the problem, in this paper, we only address one moving target point in the network. Also we assume that only one ‘on period’ and one ‘off period’ are sufficient for scheduling of each node. This means that the target passes the sensing area of the node every T_1 time slices for duration of T_2 time slices and each node has to learn these two periods. T_1 and T_2 can be changed dynamically throughout the lifetime of the network, but their changing rate is not so fast that the learning cannot be occurred at all. The proposed method can be extended to overcome these simplifications, but for best specifying the basis of our work, we assume them in this paper and address these extensions in future works.

The rest of this paper is organized as follows. The problem statement is given in section 2. Learning automata as a basic learning strategy used in the proposed method will be discussed in section 3. In section 4 the proposed method is presented. Simulation results are given in section 5. Section 6 is the conclusion.

2. Problem Statement

We are interested in the dynamic point coverage problem in which a single target point is moving throughout the area of the sensor network and should be detected by nodes which are close enough and have the ability to sense its movement. The problem is to determine the precise times of going to off state and coming back to on for each node based on the movement strategy of the target point. We assume that only one ‘on period’ and one ‘off period’ are sufficient for scheduling of each node. More specifically, the target passes the sensing area of node i every T_{i1} time slices for a duration of T_{i2} time slices, and the problem at hand is to determine these two time slices for each node in the network. Figure 1 gives a sample scenario for movement of a target point in the area of sensor network which can be sufficiently scheduled by one 'on period' and one 'off period' in each node.

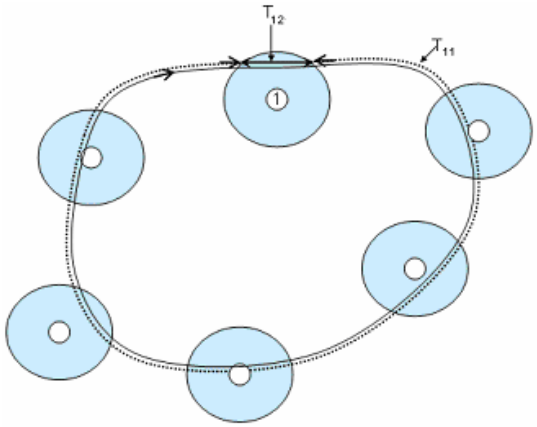


Fig. 1. A sample movement scenario of a target point in a sensor network which can be sufficiently scheduled by one 'on period' and one 'off period' in each node. T_{12} and T_{11} show the suitable on and off periods of node 1.

Determination of going to off state (or equivalently T_{i2}) is not so complicated. Whenever node i cannot sense the target anymore, it should go to off state. On the other hand, finding the precise time of coming back to on state (or equivalently T_{i1}) is more complicated. If the movement strategy of the target point is static, saying that T_{i1} and T_{i2} are static for every node i, these values can be determined by allowing each node to be in on state for a complete duration of 'sensing-not sensing-sensing' or 'not sensing-sensing-not sensing-sensing'. This way, precise times of T_{i1} and T_{i2} can be determined for every node i. Assuming such a static strategy of movement is not so realistic. Even for targets with static movement strategy, many external parameters such as wind speed may change the durations of T_{i1} and T_{i2}. In this paper, we assume that these periods may change during the lifetime of the network and we are interested in a scheduling algorithm which can learn these values and address their changes adaptively.

The sensor field is represented by a two-dimensional rectangular area. The detection by each sensor is modeled as a circle on the two-dimensional field. The center of the circle denotes the sensor while the radius denotes the detection range of the sensor. We consider a binary detection model in which a target is detected (not detected) with complete certainty by the sensor if it is inside (outside) the sensor's detection circle.

More formally, let *s* be an individual sensor node on the sensor field located at point *(x, y)*. Each sensor node has a detection range of *r*. For any target point *P* at *(i, j)*, we denote the Euclidean distance between *s* and *P as* $d_{ij}(x, y)$, i.e. $d_{ij}(x, y) = \sqrt{(x-i)^2 + (y-j)^2}$. Equation (1) shows the binary sensor model [22] that expresses the coverage $c_{ij}(x, y)$ of a target point at *(i, j)* by sensor *s* at *(x,y)*.

$$c_{ij}(x, y) = \begin{cases} 1; & if\ d_{ij}(x, y) < r \\ 0; & otherwise \end{cases} \quad (1)$$

3. Learning Automata

Learning automata is an abstract model which randomly selects one action out of its finite set of actions and performs it on a random environment. Environment then evaluates the selected action and responses to the automata with a reinforcement signal. Based on selected action, and received signal, the automata updates its internal state and selects its next action. Figure 2 depicts the relationship between an automata and its environment.

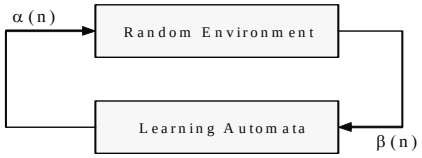


Fig. 2. Relationship between learning automata and its environment

Environment can be defined by the triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents a finite input set, $\beta = \{\beta_1, \beta_2, ..., \beta_r\}$ represents the output set, and $c = \{c_1, c_2, ..., c_r\}$ is a set of penalty probabilities, where each element c_i of c corresponds to one input of action α_i. An environment in which β can take only binary values 0 or 1 is referred to as P-model environment. A further generalization of the environment allows finite output sets with more than two elements that take values in the interval [0, 1]. Such an environment is referred to as Q-model. Finally, when the output of the environment is a continuous random variable which assumes values in the interval [0, 1], it is referred to as an S-model. Learning automata are classified into fixed-structure stochastic, and variable-structure stochastic. In the following, we consider only variable-structure automata.

A variable-structure automaton is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \cdots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \cdots, p_r\}$ represents the action probability set, and finally $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ represents the learning algorithm. This automaton operates as follows. Based on the action probability set *p*, automaton randomly selects an action α_i, and performs it on the environment. After receiving the environment's reinforcement signal, automaton updates its action probability set based on equations (2) for favorable responses, and equations (3) for unfavorable ones.

$$\begin{aligned} p_i(n+1) &= p_i(n) + a.(1 - p_i(n)) \\ p_j(n+1) &= p_j(n) - a.p_j(n) \qquad \forall j \ \ j \neq i \end{aligned} \quad (2)$$

$$\begin{aligned} p_i(n+1) &= (1-b).p_i(n) \\ p_j(n+1) &= \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j \ \ j \neq i \end{aligned} \quad (3)$$

4. Proposed Method

Each node *i* in the network is equipped with a learning automaton. This learning automaton helps the node in determining it's suitable off state period (T_{i1}). For this purpose, each node assumes it's off state period to be 1 time slice at the beginning of the algorithm. Later on, through the interaction of the learning automaton of the node with its environment, this time period will be enhanced until it precisely coincides with its correct value. Learning automaton of each node has three actions; *extending* or *shortening* the off state period, and *no change* action. At the beginning of the algorithm, *extending* action has the probability very near to 1 and *shortening* and *no change* actions both have the probability very near to 0. This is because at the beginning, the off period is very short and it is desired for it to extend and get closed to T_{i1} as soon as possible. As the algorithm goes on, the off period gradually increases and gets closed to T_{i1}, and hence the probability of *extending* action should decrease to near 0 whereas the probability of *no change* action increases to near 1. *Shortening* action never gets large probability values, since it only controls the off period from becoming too large and has no direct effect on it.

The proposed algorithm is as follows: Each node starts in on state, sensing its surrounding area to determine whether the target is currently in its sensing area or not. If the target is not in its sensing range, the node continues in on state since it can sense the target. An upper bound for this period should be considered (*MAX_WAITING*) in order to prevent endless waiting for a target which never passes through the sensing area of the node. Nodes which pass *MAX_WAITING* limit and cannot sense any target, go to off state for a very long duration (*RESTART_PERIOD*). After this duration, they come to on state and start executing the algorithm from

the beginning again. This should be done to tackle with probable changing in the target's moving path.

If the node can sense the target before *MAX_WAITING* limit, it continues in on state, keep on monitoring the target positions. Whenever the node cannot sense the target anymore, it decides to change to off state, but before going to off state, it asks its learning automaton to determine the suitable off period for the node. Learning automaton of the node, selects one of its actions, which is whether to extend or shorten the previous off period or not change it at all. Based on the selected action of the automaton, the off period of the node will be extended, shortened or remained unchanged. This period would not be less than 1 time slice and more than *MAX_PERIOD*, so further checking will be done, not to shorten one-time slice off periods or extend *MAX_PERIOD*-time slice ones. Node goes to off state for the specified duration and then comes back to on state. When node comes back to on state, it checks for some short time slots (*CHECKING_PERIOD*) to see whether the target is in its sensing area or not and based on the result of this checking, makes the reinforcement signal to its learning automaton as follows:

- If the target is not in the sensing range of the node during *CHECKING_PERIOD*, it shows that the off period is smaller than required and hence *shortening* and *no change* actions are penalized or *extending* action is rewarded according to the equations 3 and 2 respectively.
- If target is in the sensing range of the node immediately when it comes back to on state, it shows that the off period is greater than required. This is because the target might come into sensing range of the node before the node comes back to on. So *shortening* action is rewarded or *extending* and *no change* actions are penalized according to the equations 2 and 3 respectively.
- If target can be detected during *CHECKING_PERIOD*, but not immediately when the node comes back to on state, the off period is almost perfect and requires no extending or shortening. So, in this case, *no change* action is rewarded or *extending* and *shortening* actions are penalized.

After the above procedure is done, if the target is detected, node continues in on state until the target cannot be sensed anymore and at this time, next off period should be determined using learning automaton of the node as explained before. If the target isn't found in *CHECKING_PERIOD*, node goes to off state for a very short period (*ALTERNATING_PERIOD*) and comes back to on state after that, checking for the presence of the target for duration of *ALTERNATING_PERIOD*. Going to off state and coming back to on state for duration of *ALTERNATING_PERIOD* is repeated until the target can be detected by the node. At this time, node continues in on state until the target cannot be sensed anymore, then next off period is determined using learning automaton of the node as explained before. Figure 3 shows the pseudo code of the main loop each node follows at the beginning of the algorithm. Figure 4 gives the pseudo code of the procedure each node follows when it can sense the target. Finally, figure 5 presents the pseudo code of the procedure each node follows upon coming back to on state from off state.

```
Procedure Main
    state = ON; //Start in ON state
    offPeriod = 1; //Starts with 1-time slice
                   // off period
    totalRun = 0;

    Do {
        totalRun++;
        if (target is not in sensing area) {
            if (totalRun >= MAX_WAITING)
                break;
            else
                Continue;
        }
    } while (true);

    if (totalRun >= MAX_WAITING) {
        state = OFF; //Go to off state
        startOffTimer(RESTART_PERIOD);
        return;
    }

    senseTarget();
End
```

Fig. 3. Pseudo code of the main loop each node follows at the beginning of the algorithm

```
Procedure senseTarget()
    Do {
        if (target is in sensing area) {
            monitorTargetPosition();
            continue;
        }
        break;
    } while (true);

    //Target is not in sensing area anymore
    action = LA.selectAction();
    if (action == EXTENDING) {
        offPeriod++;
        if (offPeriod > MAX_PERIOD)
            offPeriod = MAX_PERIOD;
    } else if (action == SHORTENING) {
        offPeriod--;
        if (offPeriod <= 1)
            offPeriod = 1;
    } else { //Action is No CHANGE
        //Do nothing
    }

    state = OFF; //Go to off state
    //Starting off timer for offPeriod time
    startOffTimer(offPeriod);
End
```

Fig. 4. Pseudo code of the procedure each node follows when it can sense the target

```
Procedure offTimerIsExpired
    if (offPeriod == RESTART_PERIOD) {
        Main();
        return;
    }
    if (offPeriod == ALTERNATING_PERIOD) {
        if (target is in sensing area) {
            senseTarget();
             return;
        } else {
            onPeriodCount = 0;
            Do {
                onPeriodCount++;
                if (target is in sensing area) {
                    senseTarget();
                    return;
                }
            } while (onPeriodCount <
                            ALTERNATING_PERIOD);
            state = OFF; //Go to off state
            startOffTimer(ALTERNATING_PERIOD);
            return;
        }
    }
    if (target is in sensing area) {
        if (LA.selectedAction==EXTENDING)
            LA.penalize(EXTENDING);
        else
          if (LA.selectedAction==SHORTENING)
            LA.reward(SHORTENING);
        else //Action was No CHANGE
            LA.penalize(NO CHANGE);
        senseTarget();
        return;
    } else { //Target is not in sensing range
        checkCount = 1;
        DO {
            if (target is in sensing area)
                break;
            checkCount++;
        } while (checkCount < CHECKING_PERIOD)
        if (checkCount >= CHECKING_PERIOD) {
            if (LA.selectedAction==EXTENDING)
                LA.reward(EXTENDING);
            else if
               (LA.selectedAction==SHORTENING)
                LA.penalize(SHORTENING);
            else //Action was No CHANGE
                LA.penalize(NO CHANGE);
            state = OFF; //Go to off state
            startOffTimer(ALTERNATING_PERIOD);
            return;
        } else { //off period is long enough
            if (LA.selectedAction==EXTENDING)
                LA.penalize(EXTENDING);
            else if
               (LA.selectedAction==SHORTENING)
                LA.penalize(SHORTENING);
            else //Action was No CHANGE
                LA.reward(NO CHANGE);
            senseTarget();
            return;
        }
    }
End
```

Fig. 5. Pseudo code of the procedure each node follows upon coming back to on state

5. Experimental Results

To evaluate the performance of the proposed method several experiments have been conducted and the proposed method is compared with methods given in [14] and [15]. The algorithm given in [15] is called Proactive Wakeup (PW).

For simulating a scenario in which the target passes through the sensing region of each node i every T_{i1} for duration of T_{i2}, we use an approach very similar to the approach used in [23]. Assume a road in which target enters from one end and exits from the other end at some constant or variable speed v bounded by v_{max} as it is depicted in figure 6. This entrance and exit is repeated periodically by the target. In our simulations, without loss of generality, we assume that the target will return to the entrance point whenever it exits from the other end of the road, i.e. T_{out} in figure 6 is assumed to be zero. 100 sensor nodes are scattered randomly through the area of the road. For simplicity, we assume the road to be of rectangular shape with dimensions of 1000m x 50m. Sensing radius of sensor nodes are assumed to be the same and equal to 5m. Each time slice in the simulations is considered to be 1 ms long.

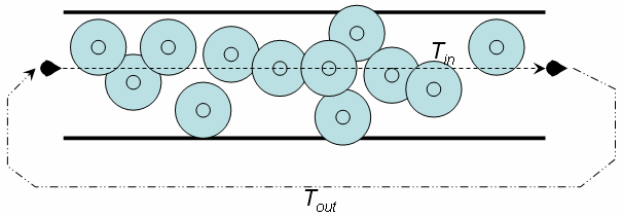


Fig. 6. Simulation scenario: Sensor nodes are scattered randomly through the area of a road and target enters the road from left and exits from right. This sequence is repeated every T_{out} (ms)

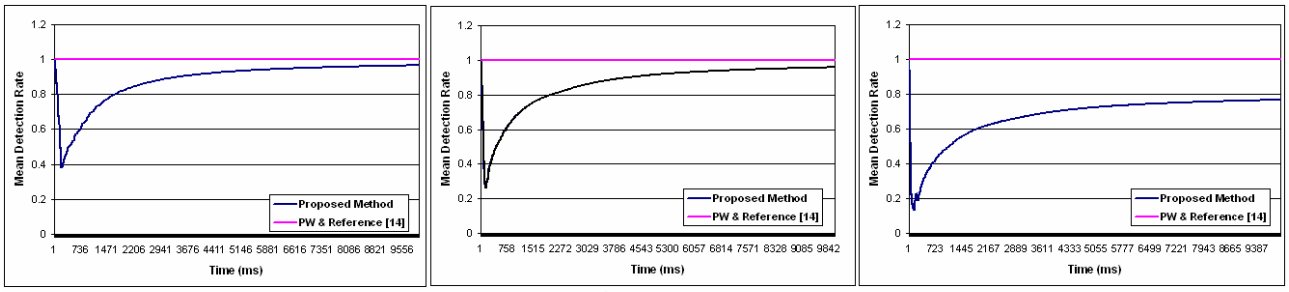
For communication energy estimation, we use a first order radio model described in [24]. In this model, energy required to run the transmitter or receiver circuitry is $E_{elec}=50\ nJ/bit$ and the transmitter amplifier requires $E_{amp}=100\ pJ/bit/m^2$. Energy required to transmit a data packet of size l bits from node i to node j is given by $T_{ij}=lE_{elec}+lE_{amp}d^2_{ij}$, where d_{ij} is the distance between node i and node j. Energy required to receive a l bit packet for any node j is given by $R_i=lE_{elec}$. Also for sleep and idle mode energy consumption, we use the specifications of MEDUSA II sensor node given in [21]; energy consumed during sleep and idle mode would be equal to .02 mW and 22 mW respectively. Energy required to switch a node from sleep to active mode is assumed to be negligible.

5.1. Experiment 1

In the first experiment, we consider the speed of the target to be constant. Experiment is repeated for three different speeds; .5 m/s, 1 m/s and 2 m/s. The purpose of the experiment is to study the detection rate, off period rate and energy consumption of nodes during the simulation. Detection rate for each node i is the rate of target detection by that node given by the equation (4).

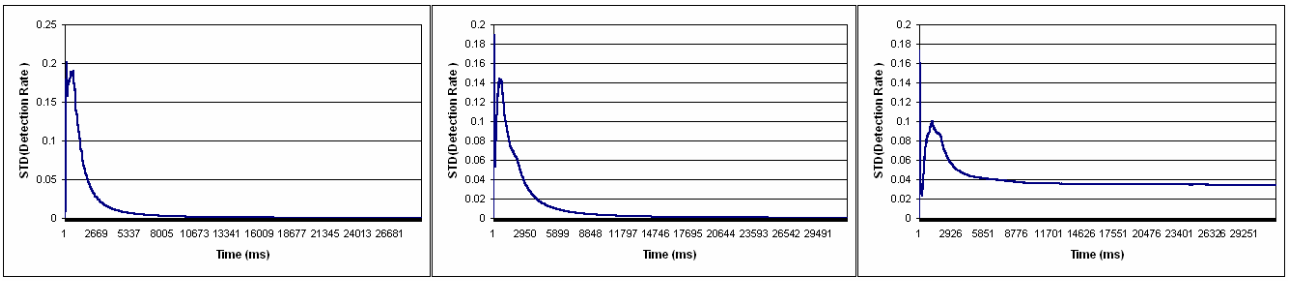
$$DR_i = \frac{DT_i}{CDT_i} \tag{4}$$

In this equation, DT_i (Detection Time slice) is the number of time slices in which the node is in on state and detects the target, and CDT_i (Can be Detected Time slice) is the number of time slices in which, the target can be detected



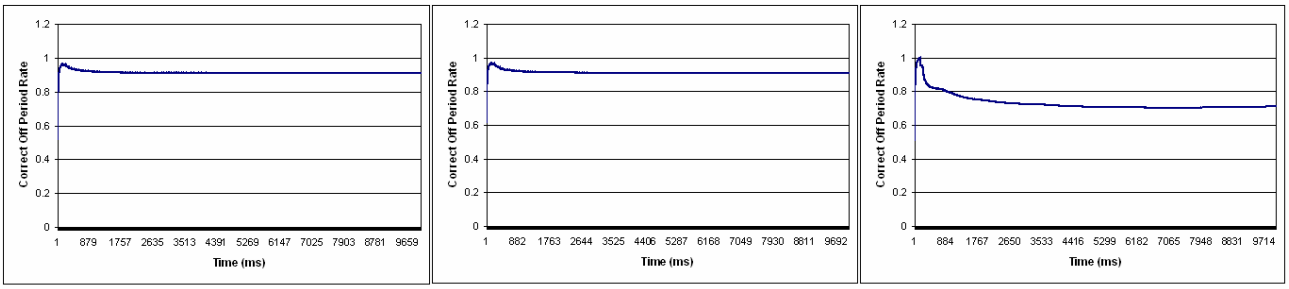
a) $v = .5$ (m/s) b) $v = 1$ (m/s) c) $v = 2$ (m/s)

Fig. 7. Mean detection rate of all 100 nodes when target's speed is constant, a) v = .5 (m/s), b) v = 1 (m/s) and c) v = 2 (m/s)



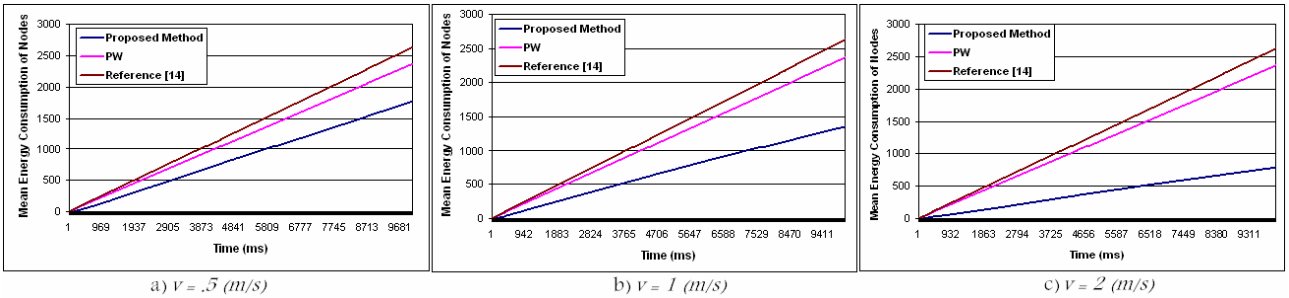
a) $v = .5$ (m/s) b) $v = 1$ (m/s) c) $v = 2$ (m/s)

Fig. 8. Standard deviation (STD) of detection rate of all nodes when target's speed is constant, a) v = .5 (m/s), b) v = 1 (m/s) and c) v = 2 (m/s)



a) $v = .5$ (m/s) b) $v = 1$ (m/s) c) $v = 2$ (m/s)

Fig. 9. Mean off period rate of all 100 nodes when target's speed is constant, a) v = .5 (m/s), b) v = 1 (m/s) and c) v = 2 (m/s)



a) $v = .5$ (m/s) b) $v = 1$ (m/s) c) $v = 2$ (m/s)

Fig. 10. Mean energy consumption of all nodes when target's speed is constant, a) v = .5 (m/s), b) v = 1 (m/s) and c) v = 2 (m/s)

by the node regardless of the status of the node which may be on or off. Similarly, off period rate for each node *i* is given by the equation (5) in which OT_i (Off Time slice) is the number of time slices in which the node is off and COT_i (Can be Off Time slice) is the number of time slices in which the target is not in the vicinity of the node, and hence the node can be off.

$$OR_i = \frac{OT_i}{COT_i} \qquad (5)$$

Figure 7 gives the average *DR* for 100 nodes for the proposed method and methods given in [14] and [15]. As it can be seen, average detection rate for methods given in [14] and [15] are almost equal and very near to 1. This is because in these methods, nodes, which are currently monitoring the target, estimate its movement path and activate the nodes which are on this path and the target will be in their vicinity in near future. This figure also shows the effect of learning on the detection rate in the proposed method; as it can be seen, detection rate first drops dramatically, but as the time goes on and each node's automaton learns its best off period, it starts increasing and eventually converges to some value near 1. Finally, it can be seen from figure 7 that for v = 2 m/s, detection rate converges to .8, which is not suitable. This is because the speed of

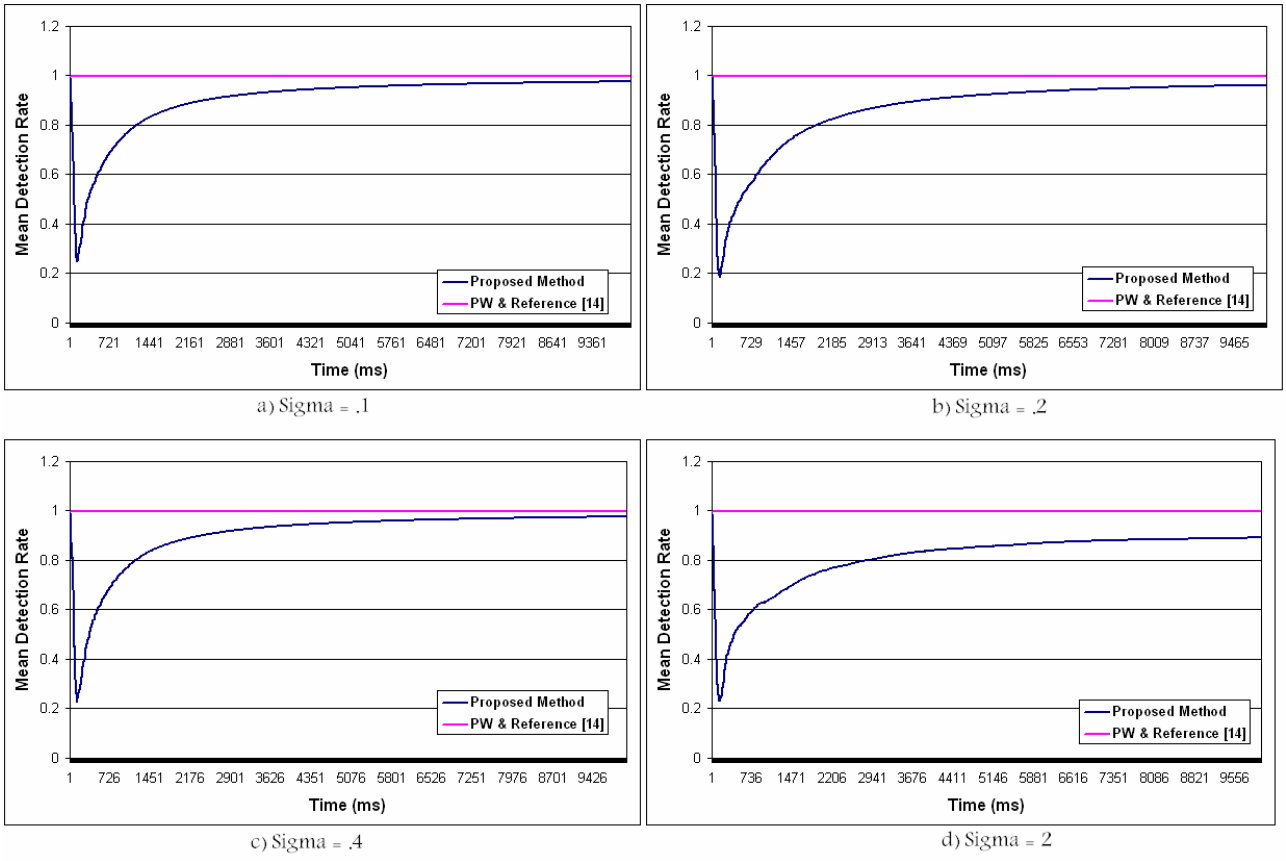


Fig. 11. Mean detection rate of all 100 nodes when target's speed is randomly selected from a normal distribution with mean=1 and a) Sigma = .1, b) Sigma = .2, c) Sigma = .4 and d) Sigma = 2

the target is very high, and hence it passes through the sensing region of the node very quickly, making its detection more difficult. Figure 8 depicts the standard deviation (STD) of the detection rate for 3 different speeds. As it can be seen from this figure, STD is very near to 0 which means that detection rates of all 100 nodes have nearly the same value, i.e. all nodes learn their best off period time.

Mean off period rate of all 100 nodes is depicted in figure 9 for three different speeds mentioned earlier. This figure shows that the off period rate converges to at least near .8 for all these speeds. Off period rate of methods given in [14] and [15] isn't depicted in this figure, since in these methods, nodes only go to idle mode in which only sensing device is off and communication device is active, and hence off period for these methods would be equal to zero.

Finally, figure 10 shows the mean energy consumption of all nodes in the sensor network for the proposed method and methods given in [14] and [15]. It can be seen from this figure that energy consumption of the nodes in the proposed method on average is about 29% and 33% of energy consumption of the nodes in methods given in [14] and [15] respectively. This noticeable reduction in energy consumption of the nodes is due to the fact that in the proposed method, nodes can go to off state, while in other methods, they can only go to idle state.

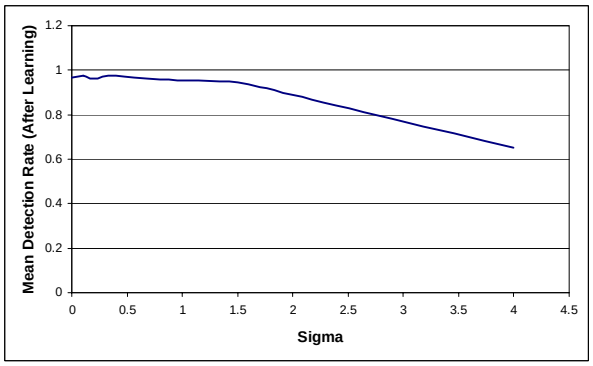


Fig. 12. Mean detection rate of all nodes after learning for different standard deviations

5.2. Experiment 2

Experiment 2 is similar to experiment 1, except for that in this experiment, target's speed is not constant, but is selected randomly in each time slice from a normal distribution with mean 1 and different standard deviations. Figure 11 depicts the mean detection rate of all 100 nodes for different standard deviations. Figure 12 gives the mean detection rate of all nodes after learning for different standard deviations. This figure shows that for standard deviations less than 1.5, the performance of the proposed method doesn't change very much, but for larger standard deviations (more difference in speed values in different time slices), the performance is dropped dramatically due to the fact

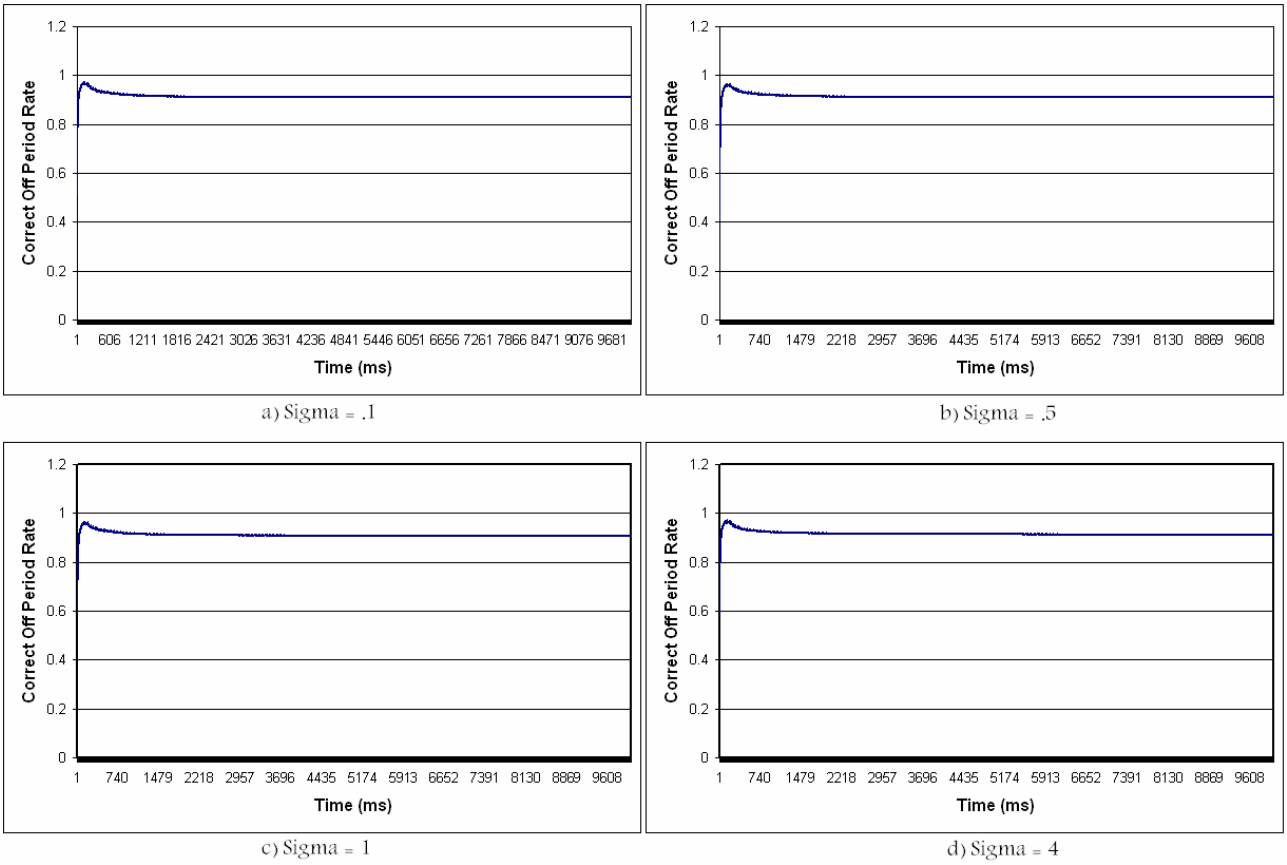


Fig. 13. Mean off period rate of all 100 nodes when target's speed is randomly selected from a normal distribution with mean=1 and a) Sigma = .1, b) Sigma = .5, c) Sigma = 1 and d) Sigma = 4

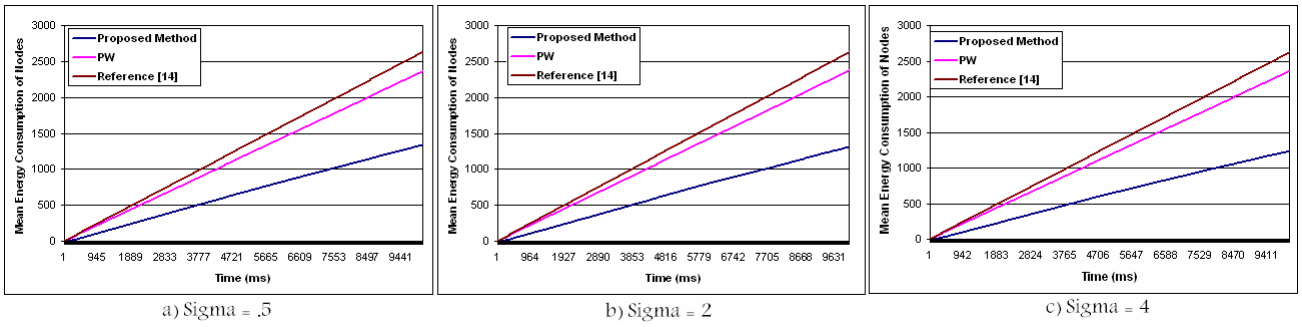


Fig. 14. Mean energy consumption of all nodes when target's speed is randomly selected from a normal distribution with mean=1 and a) Sigma = .5, b) Sigma = 2 and c) Sigma = 4

that quick and large changes in the target's speed doesn't allow learning to be occurred.

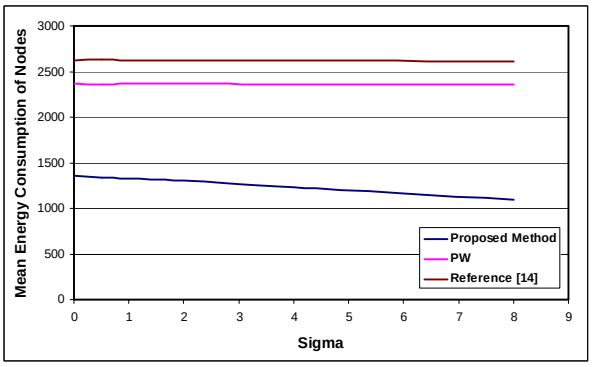


Fig. 15. Mean energy consumption of nodes after learning for different standard deviations

Figure 13 gives the mean off period rate of all 100 nodes in the sensor network. This figure shows that off period rate doesn't change dramatically with changes in the standard deviation of the target's speed. This means that changes in the target's speed only affects the detection rate, and has no or very few effects on the correct off period rate. In other words, with quick and large changes in target's speed, nodes would be in off state more often, and hence detection rate decreases, while off period rate doesn't change very much.

Finally, figure 14 shows the mean energy consumption of all nodes in the sensor network for the proposed method and methods given in [14] and [15]. It can be seen from this figure that energy consumption of the nodes in the proposed method on average is about 51% and 56% of energy consumption of the

nodes in methods given in [14] and [15] respectively. Figure 15 depicts the mean energy consumption of the nodes as the standard deviation of the target's speed increases. It can be seen from this figure that in the proposed method, less energy is consumed while the standard deviation of target's speed in increase. This is due to the fact that nodes would be in off state more often when target's speed changes more quickly as it was mentioned earlier.

5.3. Experiment 3

This experiment is conducted to better understand the convergence behavior of learning automata residing in the nodes of the network. We study the action probabilities of an automaton residing in a randomly selected node in the network. Figures 16 to 18 depict the action probability of different actions of this automaton as the time passes on. From figure 16 it can be seen that *extending* action probability starts from .8, goes up to near 1 and then, after the off period matches to its real value, starts decreasing to near 0. Figure 17 shows that *no change* action probability starts from .1, decreases to near 0 and then, after the off period matches to its real value, starts increasing to near 1. Finally, *shortening* action probability doesn't change dramatically and always has a value near 0 as it can be seen in figure 18. These three figures justify our discussion at the beginning of section 4 about the action probabilities of different actions of automata residing in different nodes of the sensor network.

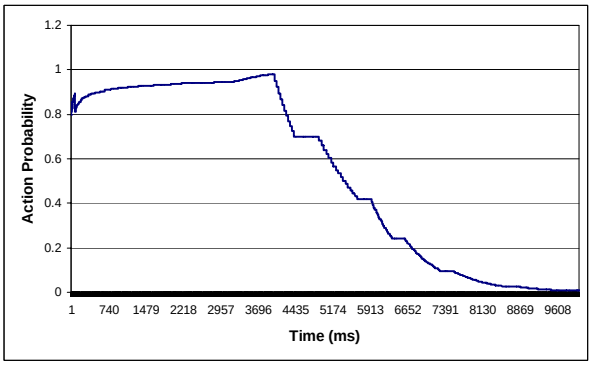


Fig. 16. *Extending* action probability

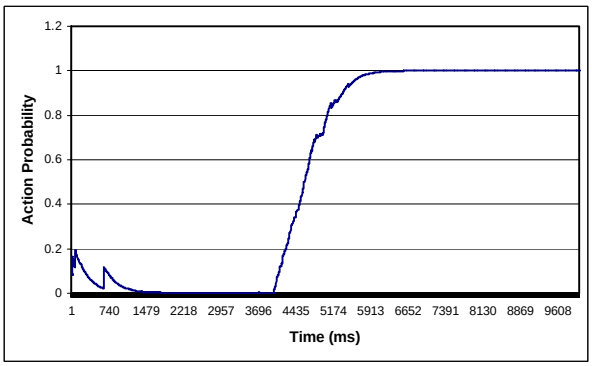


Fig. 17. *No change* action probability

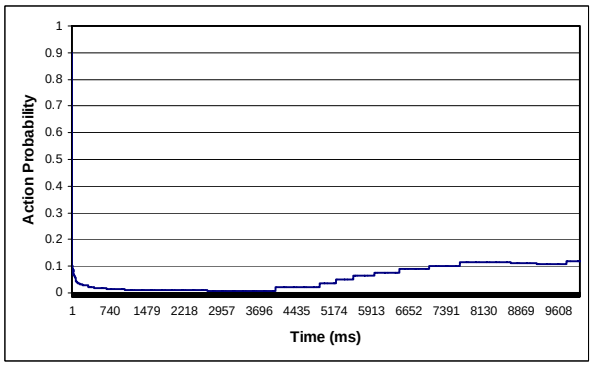


Fig. 18. *Shortening* action probability

6. Conclusion

In this paper, we propose a completely different and novel solution based on learning automata to the problem of dynamic point coverage in wireless sensor networks. Previous works in this area use some sort of notification messages which should be exchanged between nodes currently in on state monitoring the target and those currently in off state, but with high probability should soon change to on state and monitor the target. This requires the nodes in off state to have their receiving units on which leads to high energy consumption and consequently shortening the lifetime of the network. In contrast, the proposed method in this paper makes use of a learning automaton in each node to make it capable of learning it's off and on periods based on the target movement path. This way, no notification messages will be required between the nodes, and hence nodes which are in off state can completely switch their receiving units off, saving as much energy as possible. Experimental results show that our method can save much more energy in each node and better prolonging the lifetime of the network than other similar methods in the expense of a bit less precision in the detection of the target.

References

[1] M. Ilyas, I. Mahgoub, "Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems", *CRC Press*, London, Washington, D.C., 2005.

[2] S. S. Dhillon, K. Chakrabarty, S. S. Iyengar, "Sensor placement for grid coverage under imprecise detections", *Proc. International Conference on Information Fusion (FUSION 2002)*, 2002, pp. 1581-1587.

[3] Y. Zou, K. Chakrabarty, "Uncertainty-aware and coverage-oriented deployment for sensor networks", *Journal of Paralle and Distributed Computing*, Vo. 64, No. 7, pp. 788-798, July 2004.

[4] H. Chen, H. Wu, N-F. Tzeng, "Grid-based Approach for Working Node Selection in Wireless Sensor Networks", *Proc. of IEEE Intl. Conf. on Communications 2004 (ICC 2004)*, 2004.

[5] Y. Zou, K. Chakrabarty, "A Distributed Coverage- and Connectivity-Centric Technique for Selecting Active Nodes in Wireless Sensor Networks", *IEEE Transactions on Computers*, Vol. 54, No. 8, pp. 978-991, August 2005.

[6] F. Pedraza, A. García, A. L. Medaglia, "Efficient coverage algorithms for wireless sensor networks", *Proc. of the 2006 Systems and Information Engineering Design Symposium*, 2006.

[7] Zh. Dingxing, X. Ming, Ch. Yingwen, W. Shulin, "Probabilistic Coverage Configuration for Wireless Sensor Networks", *2nd Intl. Conf. on Wireless Communications, Networking and Mobile Computing (WiCOM 2006)*, Wuhan, September 2006.

[8] M. Schwager, J. McLurkin, D. Rus, "Distributed Coverage Control with Sensory Feedback for Networked Robots", *Proc. of Robotics: Science and Systems*, Philadelphia, PA, August, 2006.

[9] M. A. Batalin, G. S. Sukhatme, "Multi-robot Dynamic Coverage of a Planar Bounded Environment", *CRES-03-011*, 2003.

[10] B. Jung, "Cooperative Target Tracking using Mobile Robots", *PhD dissertation proposal*, University of Southern California, February 2004.

[11] B. Shucker, J. K. Bennett, "Target Tracking with Distributed Robotic Macrosensors", *Proc. of MILCOM 2005*. Atlantic City, New Jersey. October, 2005.

[12] R. Olfati-Saber, "Distributed Tracking for Mobile Sensor Networks with Information- Driven Mobility", *Proc. of the 2007 American Control Conference*, New York, NY, July 2007, pp. 4606–4612.

[13] S. Pattem, S. Poduri, Bh. Krishnamachari, "Energy-Quality Tradeoffs for Target Tracking in Wireless Sensor Networks", *Second Intl. Workshop on Information Processing in Sensor Networks*, Palo Alto, CA, USA, April 2003.

[14] R. Gupta and S.R. Das, "Tracking Moving Targets in a Smart Sensor Network", *Proc. IEEE VTC*, vol. 5, pp. 3035-3039, Oct. 2003.

[15] Ch. Gui, P. Mohapatra, "Power conservation and quality of surveillance in target tracking sensor networks", *Proc. of the 10th Annual Intl. Conf. on Mobile Computing and Networking (MOBICOM 2004)*, Philadelphia, PA, USA, September-October, 2004.

[16] G. He, J. C. Hou, "Tracking Targets with Quality inWireless Sensor Networks", *13th IEEE Intl. Conf. on Network Protocols (ICNP 2005)*, Boston, MA, USA, November 2005.

[17] M. K. Watfa, S. Commuri, "A Reduced Cover Approach to Energy Efficient Tracking using Wireless Sensor Networks", *World Congress in Computer Sceince, Computer Engineering and Applied Computing*, Las Vegas, Nevada, USA, June 2006.

[18] J. Jeong, S. Sharafkandi, D. H. C. Du, "Energy-aware scheduling with quality of surveillance guarantee in wireless sensor networks", *Intl. Conf. on Mobile Computing and Networking*, Los Angeles, CA, USA, 2006.

[19] B. Liu, P. Brass, O. Dousse, "Mobility Improves Coverage of Sensor Networks", *Proc. of the the 6th ACM Intl. Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc '05)*, Urbana-Champaign, Illinois, USA, May 2005, pp. 300–308.

[20] J. Jeong, T. Hwang, T. He, D. Du, "MCTA: Target Tracking Algorithm based on Minimal Contour in Wireless Sensor Networks", *In IEEE Infocom2007 Minisymposia*, August 2007.

[21] V. Raghunathan, C. Schurgers, S. Park, M. B. Srivastava, "Energy-Aware Wireless sensor networks", *IEEE Singal Processing Magazine*, 2002-03.

[22] K. Chakrabarty, S. S. Iyengar, H. Qi and E. Cho, "Grid coverage for surveillance and target Location in distributed sensor networks", *IEEE Transactions on Computers*, vol. 51, pp. 1448-1453, 2002.

[23] J. Jeong, S. Sharafkandi, D. H. C. Du, "Energy-aware scheduling with quality of surveillance guarantee in wireless sensor networks", *Intl. Conf. on Mobile Computing and Networking*, Los Angeles, CA, USA, 2006.

[24] K. K. Koustuv Dasgupta, P. Namjoshi, "An efficient clustering-based heuristic for data gathering and aggregation in sensor networks", *Proc. of the IEEE Wireless Communications and Networking Conf.*, 2003.