Computer Networks xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

A learning automata based scheduling solution to the dynamic point coverage problem in wireless sensor networks

M. Esnaashari *, M.R. Meybodi

Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 6 September 2009
Received in revised form 28 February 2010
Accepted 29 March 2010
Available online xxxx
Responsible Editor: A. Kshemkalyani

Keywords:
Dynamic point coverage
Scheduling
Learning automata
Wireless sensor network

ABSTRACT

The dynamic point coverage problem in wireless sensor networks is to detect some moving target points in the area of the network using as few sensor nodes as possible. One way to deal with this problem is to schedule sensor nodes in such a way that a node is activated only at the times a target point is in its sensing region. In this paper we propose SALA, a scheduling algorithm based on learning automata, to deal with the problem of dynamic point coverage. In SALA each node in the network is equipped with a set of learning automata. The learning automata residing in each node try to learn the maximum sleep duration for the node in such a way that the detection rate of target points by the node does not degrade dramatically. This is done using the information obtained about the movement patterns of target points while passing throughout the sensing region of the nodes. We consider two types of target points; events and moving objects. Events are assumed to occur periodically or based on a Poisson distribution and moving objects are assumed to have a static movement path which is repeated periodically with a randomly selected velocity. In order to show the performance of SALA, some experiments have been conducted. The experimental results show that SALA outperforms the existing methods such as LEACH, GAF, PEAS and PW in terms of energy consumption.

© 2010 Elsevier B.V. All rights reserved.

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: covering (monitoring) the whole area of the network is the main objective of area coverage problem.
- 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. A common definition of this problem is to cover (monitor) some stationary or moving target points in the area of sensor network using as few sensor nodes as possible [1]. This problem can be addressed in different ways. One way is to design a deployment strategy which can best address the criterion of minimum number of required nodes [2–7]. This is an inflexible solution which cannot deal with topology changes, commonly occuring in sensor networks due to high probability of sensor failures. Another more flexible solution to the point coverage problem is to dynamically rearrange nodes assuming they have moving ability [8–12]. In this approach, nodes are first deployed

* Corresponding author. Tel.: +98 21 64542744.
E-mail addresses: Esnaashari@aut.ac.ir (M. Esnaashari), mmeybodi@aut.ac.ir (M.R. Meybodi).

1389-1286/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2010.03.014

randomly around the target points. After this initial random positioning, each node tries to find its best position using the position information of the target points and its surrounding nodes. 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 sleep-scheduling (or simply scheduling) mechanism for nodes in such a way that in each period of time, only nodes which can sense the target points in that period are awakened [13–20]. In terms of dynamicity, this solution can deal with changes that occur in the topology of the network and position of target points using a dynamic scheduling mechanism. In terms of energy consumption, this method best fits the sensor networks, since in each period of time, only nodes which can sense the target points in that period are awakened, and rest of the nodes are asleep, saving their energy.

Sleep-scheduling can be performed using a centralized or a decentralized approach. Centralized scheduling algorithms are suitable only for stationary targets or moving targets with known and static movement patterns. For targets with unknown and dynamically changing movement patterns, decentralized scheduling algorithms which are flexible enough to adapt themselves to these changes are required. In these decentralized scheduling schemes, usually some of the nodes having higher residual energy are awakened all of the times, monitoring the whole area for detecting target points. Whenever a target point is detected, awakened nodes track the target and estimate its movement path. Based on this estimation, awakened nodes activate the sleeping nodes in their vicinity which will be probably on the movement path of the target. Activating neighbor nodes is performed using some sort of notification messages. This mechanism has two major drawbacks; one is the overhead of the notification messages and the other is that sleeping nodes should have the ability to receive messages, and hence they cannot power off their receiving antenna. This means that a sleeping node can only switch off its processing unit, but its communicating unit must be in idle state, waiting for notification messages. According to [21], energy consumption of a sensor node in receiving and idle states is nearly equal to the energy consumed during transmission state. As the energy consumed by a processing unit is in the order of .001 of the energy consumed by a communicating unit, it is concluded that using these scheduling mechanisms, not so much energy saving can be gained in sleeping nodes and hence, the network lifetime is not prolonged too much.

To overcome the above drawbacks, in this paper a novel scheduling algorithm based on learning automata called SALA is proposed to deal with the problem of dynamic point coverage. In SALA, no notification messages are required to be exchanged between nodes and as a consequence, sleeping nodes can switch off their communicating units as well as their processing units, saving more energy and hence prolonging the network lifetime. In this algorithm each node in the network is equipped with a set of learning automata which try to learn the proper times for sleeping and awakening of that node (schedule) based on the movement patterns of the target points passing throughout the sensing region of the node. We consider two types of target points; events and moving objects. Events are assumed to occur periodically or based on a Poisson distribution and moving objects are assumed to have a static movement path which is repeated periodically with a randomly selected velocity. Experimental results show that SALA outperforms the existing methods such as LEACH[1] [29], GAF[2] [39], PEAS[3] [65] and PW[4] [15] in terms of energy consumption.

The rest of this paper is organized as follows. Section 2 gives a brief literature overview. Learning automata as a basic learning strategy used in SALA will be discussed in Section 3. The problem statement is given in Section 4. In Section 5 the proposed algorithm is presented. Simulation results are given in Section 6. Section 7 is the conclusion.

2. Related work

Scheduling algorithms can be classified into three categories; MAC layer algorithms, routing layer algorithms and application layer algorithms. Application layer algorithms can be further classified into grid-based algorithms, coverage-related algorithms and tracking-based algorithms.

2.1. MAC Layer scheduling algorithms

In this category of algorithms, scheduling is performed in the MAC layer [31–38,96–100]. Usually, this is done by allocating a slot for one node per neighborhood uniquely. This collision-free slot is used by that node for transmissions to any or all of its neighbors. Thus two nodes cannot be assigned the same slot if one station is within the range of the others, or if two stations have common neighbors. A common approach for allocating such collision-free slots is to use duty cycles. A duty cycle consists of an active section, in which the transceiver is powered on, and a sleep section, in which the transceiver is powered off. The objective of these algorithms is to allow communication without interference, while maximizing the number of parallel transmissions. Learning schemes are used in a number of MAC layer scheduling algorithms [97–100] to dynamically and adaptively adjust the length of the duty cycles of different sensor nodes.

2.2. Routing layer scheduling algorithms

This category of scheduling algorithms relates to the hierarchical routing [23–30,101,102]. In hierarchical routing algorithms, usually a number of clusters are formed in the network, each having one node as cluster head. Rests of the nodes in each cluster are called cluster members. Cluster members must send their readings to their cluster

[1] Low Energy Adaptive Clustering Hierarchy.
[2] Geographical Adaptive Fidelity.
[3] Probing Environment and Adaptive Sensing.
[4] Proactive Wakeup based on PEAS.

heads, therefore, cluster heads schedule the sending time of their members to prevent intra-cluster collisions. This way, a cluster member needs to be activated only at its scheduled sending times. Machine learning schemes can be used in cluster heads for selecting members adaptively based on their traffic rates [101,102].

2.3. Grid-based scheduling algorithms

In this category of scheduling algorithms [39–48], usually the network is partitioned into rectangular or hexagonal grids. Only one node is scheduled to be active in each grid at any time, to monitor that grid and to relay the data from other grids. Grid dimensions are calculated in such a way that a node residing within a grid can monitor the whole area of that grid and communicate with all nodes that reside in its neighboring grids.

2.4. Coverage-based scheduling algorithms

This category of scheduling algorithms, deals with the problem of covering (monitoring) either the entire area of the network or some target points (stationary or moving) in the area of the network using as little sensor nodes as possible [1]. Usually, a minimum number of nodes are selected to be active and monitor the area or the target points while the rests of the nodes are inactive and save energy. The node selection is repeated periodically or based on a certain schedule to allow balance energy consumption of all nodes. A number of centralized [4–6,49–53] and decentralized [7,25,52–68,103] methods are given in the literature for addressing this problem. In centralized methods, by assuming that the sink node has the topology information of the network, usually the problem is solved optimally using a linear integer programming approach or sub-optimally using a heuristic approach. In distributed methods, each node locally and periodically checks whether it is necessary for it to be active or not. It is necessary for a node to be active only if the sensing region of the node cannot be covered completely by its neighbors. Necessity for becoming an active node can be determined using a learning scheme [103].

2.5. Tracking-based scheduling algorithms

The last category of scheduling algorithms is tracking-based algorithms. These scheduling algorithms are used to address the tracking problem in which a number of targets are moving throughout the network and the objective of the network is to monitor the movement path of these targets. Since at any instance of time, only a fraction of the nodes are able to monitor a moving target, rest of the nodes can be inactive and save their energy. In these algorithms [15,69–79,104–106], usually the nodes which are currently monitoring the targets predict the movement path of the targets in some way. Based on this prediction, a subset of currently inactive nodes which are more probable to be able to monitor the moving targets in near future are scheduled to be activated. Activation is performed using some sort of notification messages. Note that these methods assume that inactive nodes have the ability to receive notification messages. To the best of our knowledge, the only usage of machine learning schemes in target tracking applications is in predicting the future positions of targets [104–106].

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 responds 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. Fig. 1 depicts the relationship between an automaton and its environment.

Environment can be defined by the triple $E = (\alpha, \beta, c)$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents a finite input set, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the output set, and $c = \{c_1, c_2, \ldots, c_r\}$ 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 (α, β, p, T) in which $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \ldots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \ldots, p_r\}$ 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 Eq. (1) for favorable responses, and Eq. (2) for unfavorable ones

$$\begin{aligned} p_i(n+1) &= p_i(n) + a \cdot (1 - p_i(n)), \\ p_j(n+1) &= p_j(n) - a \cdot p_j(n), \quad \forall j\ j \neq i, \end{aligned} \tag{1}$$

$$\begin{aligned} p_i(n+1) &= (1 - b) \cdot p_i(n), \\ p_j(n+1) &= \frac{b}{r-1} + (1 - b) p_j(n), \quad \forall j\ j \neq i. \end{aligned} \tag{2}$$

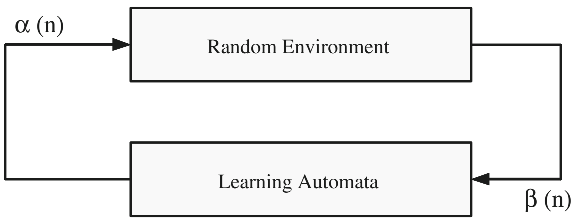

Fig. 1. Relationship between learning automata and its environment.

Learning automata is a stochastic model operating in the framework of reinforcement learning [10]. Reinforcement learning or learning with a critic is a framework of learning problems in which the teacher or the environment does not indicate the correct action, but provides only a scalar evaluative response to the selection of an action by the learner. Learning automata can be classified under the reinforcement learning schemes in the category of temporal-difference (TD) learning methods. TD learning is a combination of Monte Carlo ideas and dynamic programming ideas. Like Monte Carlo methods, TD methods can learn directly from raw experience without a model of the environment's dynamics. Like dynamic programming, TD methods update estimates based in part on other learned estimates, without waiting for a final outcome [109]. Sarsa [110], Q-learning [111], Actor-Critic methods [112] and R-learning [113] are other samples of TD methods. Learning automata differs from other TD methods in the following two ways: (1) the representation of the internal states (a set of action probabilities) and (2) the updating method of the internal states (Eqs. (1) and (2)).

Learning automata has found applications in parameter optimization, statistical decision making, telephone routing, pattern recognition, game playing, natural language processing, modeling biological learning systems and object partitioning [108]. Furthermore, learning automata is proved to perform well in the dynamic environments of computer networks. It is used in wireless networks for adaptive rate control [80,81], bandwidth control [82–84] and designing reliable transport layer protocol (Learning-TCP) [85]. Learning automata is also used in cellular radio networks to dynamically adjusting the number of guard channels [86–88]. Recently, a few attempts are made for applying learning automata to sensor networks for mobicast routing [89] and clustering [90].

4. Problem statement

In this section, we first give a sample application and then give the formal definition of the problem.

4.1. Sample applications

A trial wireless sensor network which is going to be tested in San Francisco is a sensor network that announces which of the parking spaces of the city is free at any moment [91]. This network uses a wireless sensor embedded in a 4-inch-by-4-inch piece of plastic, fastened to the pavement adjacent to each parking space. In this application, each sensor node has to monitor its parking space and reports the free times of the parking space to a local base station. The local base station then prepares the information of free parking spaces for drivers passing the area and requesting such information. From the viewpoint of the sensor network, cars coming into the parking spaces and getting out of them are moving target points which must be monitored. If a sensor node is able to switch periodically between active and sleep operation modes, it will have a longer lifetime than if it always remains in the active operation mode. Therefore, each node can utilize a local scheduling scheme for switching its operation mode between sleep and active modes. However this makes it possible for a driver to be mistakenly guided to a node's parking space, while it is occupied by another car. Such incorrect guides are acceptable while their rate is below an acceptable level.

As an alternative application, we can consider the habitat monitoring [94] such as the one reported by the College of Atlantic (COA) in the Great Duck Island for habitat monitoring of 5000 pairs of petrels nesting on the island [95]. To analyze the patterns of the organisms, the sensors are deployed both in the organisms' burrows as well as on the surface surrounding the burrows to monitor the differences. In order to determine why petrels nest in specific patches, data are gathered from both populated and unpopulated petrel patches. In order to monitor an entire field season (corresponding to a single petrel reproductive cycle), the sensors mange their power in such a way that they provide at least 7 months of continuous operation. In such application, it is rational to sacrifice precision to some degree in order to save more energy and prolong the network lifetime.

4.2. Formal definition of the problem

Consider a sensor network with N sensor nodes $s_1, s_2, \ldots, s_N$ which are scattered randomly throughout a large $L \times L$ rectangular field (Ω) in such a way that Ω is completely covered. All sensor nodes have the same sensing range of r. Each sensor node s_k has four different modes of operation [93] as follows:

- **On-duty** ($\boldsymbol{S_A C_A}$): both sensing and communicating units are switched on referred to as active mode.
- **Sensing Unit On-duty** ($\boldsymbol{S_A C_S}$): the sensing unit is switched on, but the communicating unit is switched off.
- **Communicating Unit On-duty** ($\boldsymbol{S_S C_A}$): the communicating unit is switched on, but the sensing unit is switched off.
- **Off-duty** ($\boldsymbol{S_S C_S}$): both sensing and communicating units are switched off referred to as sleep mode.

Note that in $S_A C_A$, $S_A C_S$, $S_S C_A$ and $S_S C_S$, index A stands for active and index S stands for sleep. For further simplicity in notation, we use index x in the above notations to refer to more than one operation mode, i.e. $S_A C_x$ refers to both $S_A C_A$ and $S_A C_S$ modes, and $S_x C_x$ refers to all four modes of operation. At any instance of time, a sensor node can be only in one of the above 4 operation modes. The operation mode of a sensor s_k at time instant t is denoted by $\mathrm{O}_{s_k}(t)$.

Let TP be a finite set of target points residing in Ω. TP is divided into two disjoint sets; *Moving objects* (TP^M) and *Events* (TP^E). A target point $tp_i^M \in TP^M$ is a moving object which has a continuous movement trajectory. We assume that the movement path of a target point tp_i^M is fixed throughout the lifetime of the network and repeated every T_e seconds, but the velocity of tp_i^M may change over time. On the other hand, a target point $tp_i^E \in TP^E$ is an event which occurs somewhere in Ω repeatedly every T_e seconds or randomly following a Poisson distribution and lasts for a short static or random duration.

Every T_e seconds is called an epoch (Ep). Each Ep^n starts at $\tau^S_{Ep^n} = (n-1)\cdot T_e, n = 1, 2, \ldots$ Each epoch Ep^n is further divided into N^R rounds (R) all having equal durations (τ_R). We refer to the mth round of epoch Ep^n by $R^{m,n}$ and to the mth round of all epochs by R^m. $R^{m,n}$ starts at $\tau^S_{R^{m,n}} = \tau^S_{Ep^n} + (m-1)\cdot\tau_R$ and lasts for τ_R. Since usually people have a repetitive movement paths in their day to day lives (from home to work in the morning-from work to home in the evening), in the sample application, every 24 h can be considered as one epoch. Furthermore, as the traffic pattern of a city changes in different hours of a day (and hence, occupancy rate of parking spaces differs in different hours), each hour of a day can be considered as one round.

We denote the Euclidean distance between a sensor node s_k located at $(x(s_k),y(s_k))$ and a target point tp_i located at $(x(tp_k),y(tp_k))$ as $d(s_k,tp_i)$, i.e. $d(s_k,tp_i) = \sqrt{(x(s_k)-x(tp_i))^2 + (y(s_k)-y(tp_i))^2}$.

Assuming the binary sensing model [22] and sensing range of r for all sensor nodes in the network, we say a target point $tp_i \in TP$ is sensed, detected or monitored by a sensor node s_k at time t if and only if $d(s_k,tp_i) < r$ and $O_{s_k}(t) = S_A C_x$. The network detects a target point $tp_i \in TP$ at time t if and only if at least one of the sensor nodes of the network detects tp_i at time t.

Definition 1. Network lifetime (T) gives the first point in time when Ω is not further completely covered by the network.

It should be mentioned that this definition of the network lifetime coincides with the *total network lifetime* metric given in [107] where the utilized criterion function (ψ^{**}) for network liveliness is *area coverage* and the length of the time interval during which the criterion must be satisfied (Δt^y_{**}) is set to zero.

Definition 2. Activation time of a target point $tp_i \in TP$ denoted by τ_{tp_i} is the summation of all the times during which tp_i can be detected by the network.

Definition 3. Activation time of a sensor node s_k denoted by τ_{s_k} is the summation of all the times during which the sensor is in $S_A C_x$ operation mode; that is $O_{s_k}(t) = S_A C_x$.

Definition 4. Detection time of a target point $tp_i \in TP$ denoted by $\tau^d_{tp_i}$ is the summation of all the times during which tp_i is detected by the network.

Definition 5. Detection time of a target point $tp_i \in TP$ by a sensor node s_k denoted by τ_{s_k,tp_i} is the summation of all the times during which tp_i is detected by s_k.

Definition 6. Network detection rate denoted by η_D is the rate of the target detection in the network and is defined according to Eq. (3). Network detection rate in a single round $R^{m,n}$ is referred to as $\eta^{m,n}_D$

$$\eta_D = \frac{\sum_{tp_i \in TP} \tau^d_{tp_i}}{\sum_{tp_i \in TP} \tau_{tp_i}}. \tag{3}$$

Definition 7. Network sleep rate denoted by η_S is defined as the ratio of the times during which nodes of the network are in sleep mode to the times during which they can be in sleep mode. η_S is defined according to Eq. (4). Network sleep rate in a single round $R^{m,n}$ is referred to as $\eta^{m,n}_S$

$$\eta_S = \frac{\sum_{s_k}\left(\frac{T-\tau_{s_k}}{T-\sum_{tp_i \in TP}\tau_{s_k,tp_i}}\right)}{N}. \tag{4}$$

The objective of the network is to detect the target points and report their location to a central node called sink. We assume that the network is clustered. Cluster heads are powerful and rechargeable nodes which are always in active mode. Each node can directly communicate with its cluster head and cluster heads form an infrastructure through which network data can be sent to the sink.

To prolong the network lifetime, a node will switch to the $S_x C_A$ operation mode only if it wants to communicate with its cluster head; otherwise, the communication unit of the node will be switched off. The sensing unit of a sensor node has to be switched on only if a target point is in its sensing range, but since nodes have no knowledge about the movement patterns of target points, they cannot calculate the exact times for switching their sensing units on or off. Decision on switching from $S_A C_x$ to $S_S C_x$ is quite simple; if no target point can be detected by the node, it turns off its sensing unit. For turning the sensing unit on, each node needs to know the time at which a target point enters its sensing region. Since this time is not known by a node, it instead uses a local scheduler which determines a duration after which the node must wake up. The node is informed of this duration before it goes to sleep. The local scheduling or simply scheduling of a sensor node s_k (Δ_{s_k}) is defined as follows.

Definition 8. The scheduling of a sensor node s_k in epoch Ep^n is denoted by the ordered list $\Delta^n_{s_k} = \langle T^{1,n}_{Sleep}(s_k), T^{2,n}_{Sleep}(s_k), \ldots, T^{N^R,n}_{Sleep}(s_k)\rangle$ where $T^{m,n}_{Sleep}(s_k)$ specifies the sleep duration (i.e. the duration of $S_S C_x$ operation mode) for node s_k in the round $R^{m,n}$.

Having the above definitions and assumptions, the problem is to locally learn the scheduling of each sensor node s_k using the movement patterns of target points during the lifetime of the network such that the network sleep rate (η_S) is maximized while the network detection rate (η_D) does not drop below an acceptable level. In the rest of this section, we give some definitions and a theorem which will be used later in the paper. The proof of the theorem is given in the Appendix.

Definition 9. $TP^{m,n}(s_k) = \langle tp^{m,n}_1(s_k), tp^{m,n}_2(s_k), \ldots, tp^{m,n}_{last}(s_k)\rangle$ is defined as list of target points which pass through the sensing region of sensor node s_k during round $R^{m,n}$ ordered according to their arrival times.

Definition 10. $\tau^{E}_{tp_i^{m,n}}(s_k)$ is defined as the time at which the target point $tp_i^{m,n}(s_k)$ enters into the sensing region of node s_k.

Definition 11. $\tau^{D}_{tp_i^{m,n}}(s_k)$ is defined as the time at which the target point $tp_i^{m,n}(s_k)$ exits from the sensing region of node s_k.

Theorem 1. *If for a sensor network we have that*

- *every node is activated at the beginning of any round,*
- *each node in the network immediately switches to the sleep operation mode if it cannot sense any target point,*
- *the network detection rate is one ($\eta_D = 1$),*

then the energy consumption of the network is minimized if and only if the sleeping duration of any node s_k in any round $R^{m,n}(T^{m,n}_{sleep}(s_k))$ is set to $\underline{T}^{m,n}_{sleep}(s_k)$ where

$$\underline{T}^{m,n}_{sleep}(s_k) = \gcd\left(\begin{array}{c}(\tau^{E}_{tp_1^{m,n}}(s_k) - \tau^{S}_{R^{m,n}}(s_k)),\\ (\tau^{E}_{tp_2^{m,n}}(s_k) - \tau^{D}_{tp_1^{m,n}}(s_k)),\ldots,\\ (\tau^{E}_{tp_{last}^{m,n}}(s_k) - \tau^{D}_{tp_{last-1}^{m,n}}(s_k))\end{array}\right); \quad \forall s_k, m, n \tag{5}$$

gcd is the greatest common divisor function.

5. SALA: a scheduling algorithm based on learning automata

5.1. Algorithm outline

The purpose of SALA is to find N^R different sleeping durations for different R^m rounds. For this reason, each node s_k in the network is equipped with N^R learning automata, $LA_k^1, LA_k^2, \ldots, LA_k^{N^R}$. Learning automaton LA_k^m gets activated during R^m rounds to help node s_k learn its proper sleeping duration during these rounds ($T^m_{sleep}(s_k)$).

At the beginning of each round $R^{m,n}$ (mth round of the nth epoch), all nodes are in active operation mode; that is $O_{s_k}(t) = S_A C_A$. The round is started asynchronously in each node s_k by activating its learning automaton LA_k^m. Learning automation LA_k^m upon its activation, selects a sleeping duration for node s_k ($T^m_{sleep}(s_k)$) to be used during that round. During round $R^{m,n}$, if node s_k senses any target point which passes through its sensing region, it collects information about this target point and sends collected data to the local base station. In addition, node s_k logs the entrance and exit times of this target point in its local database. During round $R^{m,n}$, whenever node s_k cannot sense any target point, it switches immediately to sleep operation mode ($S_S C_S$) and stays in this mode for the duration of $T^m_{sleep}(s_k)$. At the end of the round $R^{m,n}$, node s_k uses its local database to evaluate the suitability of the selected sleeping duration ($T^m_{sleep}(s_k)$). The result of this evaluation is then given to learning automaton LA_k^m as the reinforcement signal generated by the environment. This way learning automaton LA_k^m gradually learns the best sleeping duration for node s_k during R^m rounds.

5.2. Detailed description

Each learning automaton has three actions; *Extending*, *Shrinking* and *No change* which we call α_1,α_2 and α_3, respectively, and refer to the probabilities of selecting each as p_1, p_2 and p_3. Node s_k extends (shrinks) its sleeping duration $T^m_{sleep}(s_k)$ if learning automaton LA_k^m selects *Extending* (*Shrinking*) and does not change $T^m_{sleep}(s_k)$ if learning automaton LA_k^m selects *No change*.

At the beginning of the algorithm we have $T^{m,0}_{sleep}(s_k) = MinSleepDuration, \forall R^m, \forall s_k, p_1 = .8$ and $p_2 = p_3 = .1$. *MinSleepDuration* is the minimum allowable sleep time for the sensing unit of a node. Since $T^{m,0}_{sleep}(s_k)$ is set to *MinSleepDuration* at the beginning of the algorithm, we set p_1 to a higher value than p_2 and p_3 in order to allow $T^m_{sleep}(s_k)$ getting close to its proper value more quickly.

Each round $R^{m,n}$ is started asynchronously in each node s_k by activation of its learning automaton LA_k^m. At the startup of the round, node s_k uses its learning automaton LA_k^m to select its sleeping duration (more details on this step will be given in Section 5.2.1) for the current round ($T^{m,n}_{sleep}(s_k)$). Node s_k then enters a loop during which it checks if it can detect any target, and if so, s_k starts reporting its information to the local base station. Whenever no target can be detected, s_k goes to sleep operation mode for the duration of $T^{m,n}_{sleep}(s_k)$ seconds. After the sleep duration, s_k continues the loop from its beginning, i.e. checking for the presence of any target. The loop continues for the whole duration of the current round at the end of which s_k computes and sends the reinforcement signal (more details on this step will be given in Section 5.2.2) to the LA_k^m. The above procedure then repeats for the next round. Fig. 2 gives the pseudo code for the SALA algorithm executed by node s_k during the round $R^{m,n}$.

The following subsections provide details for how to use LA_k^m to select $T^{m,n}_{sleep}(s_k)$ at the startup of the $R^{m,n}$ round and how to compute the reinforcement signal for LA_k^m at the end of the $R^{m,n}$ round.

5.2.1. Selecting the sleeping duration

As we stated before, at the beginning of each round $R^{m,n}$, each node s_k uses its learning automaton LA_k^m to select its sleeping duration for that round ($T^{m,n}_{sleep}(s_k)$). This is done as follows. Learning automaton LA_k^m selects one of its actions randomly based on its action probability vector. We refer to the selected action as $\hat{\alpha}$. If the selected action ($\hat{\alpha}$) is *Extending*, then current sleeping duration is extended by a predetermined constant value (*ExtendValue*). If $\hat{\alpha}$ is *Shrinking*, then current sleeping duration is shrunk by a predetermined constant value (*ShrinkValue*). Otherwise, sleeping duration remains unchanged. More attention must also be paid not to extend or shrink the sleeping duration beyond the predetermined range [*MinSleepDuration*, *MaxSleepDuration*]. Fig. 3 gives the procedure of selection of the sleeping duration for round $R^{m,n}$ executed by node s_k.

The above procedure is used at the beginning of $R^{m,n}$ round to select the sleeping duration of node s_k for the round $R^{m,n}(T^{m,n}_{sleep}(s_k))$. $T^{m,n}_{sleep}(s_k)$ will also be changed at the end of $R^{m,n}$ round according to the movement patterns of the target points passed through the sensing region of s_k in that round. This is described in the next section.

```
1. $s_k$ sets $t = \tau^S_{R^{m,n}}$;
2. $s_k$ sets $O_{s_k}(t) = S_A C_x$;
3. $s_k$ uses $LA^m_k$ to select $T^{m,n}_{sleep}(s_k)$;
4. $s_k$ checks if any target can be detected;
5. If ($s_k$ detects any target point) Then
    5-1. $s_k$ stores $t$ as the entrance time of the target;
    5-2. $s_k$ sets $O_{s_k}(t) = S_A C_A$;
    5-3. While ($s_k$ detects the target point) Do
        5-3-1. $s_k$ reports data to the local base station;
        5-3-2. $s_k$ sets $t = t + 1$;
    5-4. $s_k$ Sets $O_{s_k}(t) = S_A C_S$;
    5-5. $s_k$ stores $t$ as the exit time of the target;
6. $s_k$ Sets $O_{s_k}(t) = S_S C_x$;
7. $s_k$ sleeps for $T^{m,n}_{sleep}(s_k)$ seconds;
8. If $T^{m,n}_{sleep}(s_k) > 0$ Then
    8-1. $s_k$ sets $t = t + T^{m,n}_{sleep}(s_k)$;
9. Else
    9-1. $s_k$ sets $t = t + 1$;
10. If ($m = N^R$) Then
    10-1. If ($t \geq \tau^S_{R^{1,n+1}}$) Then
        10-1-1. $s_k$ sends reinforcement signal to $LA^m_k$;
        10-1-2. $s_k$ starts the procedure for round $R^{1,n+1}$ from step 1;
    10-2. Else
        10-2-1. $s_k$ continues from step 4;
11. Else
    11-1. If ($t \geq \tau^S_{R^{m+1,n}}$) Then
        11-1-1. $s_k$ sends reinforcement signal to $LA^m_k$;
        11-1-2. $s_k$ starts the procedure for round $R^{m+1,n}$ from step 1;
    11-2. Else
        11-2-1. $s_k$ continues from step 4;
```

Fig. 2. Pseudo code for the SALA algorithm executed by node s_k during the round $R^{m,n}$.

5.2.2. Computing the reinforcement signal

Theorem 1 states that the best sleeping duration in terms of energy consumption and network detection rate for node s_k during round $R^{m,n}$ is $\underline{T}^{m,n}_{sleep}(s_k)$. This implies that the reinforcement signal to the learning automaton of node s_k better be computed based on $\underline{T}^{m,n}_{sleep}(s_k)$. But since the exact value of $\underline{T}^{m,n}_{sleep}(s_k)$ cannot be computed during the operation of the network (because $\tau^E_{tp^{m,n}_i}(s_k)$ is not known to s_k) we instead use an estimate of $\underline{T}^{m,n}_{sleep}(s_k)(\widehat{\underline{T}}^{m,n}_{sleep}(s_k))$ by recording the times at which target points enter and exit the sensing region of the node s_k (steps 5–1 and 5–5 of the SALA pseudo code). $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$ is computed at the end of the round $R^{m,n}$. Using $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$, the reinforcement signal for the learning automaton of node s_k is specified as follows:

- If the selected sleeping duration $(T^{m,n}_{sleep}(s_k))$ is below $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$ and the selected action of learning automaton is *Extending*, then the reinforcement signal is to reward the selected action.
- If the selected sleeping duration $(T^{m,n}_{sleep}(s_k))$ is below $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$ and the selected action of learning automaton is *Shrinking* or *No change*, then the reinforcement signal is to penalize the selected action.
- If the selected sleeping duration $(T^{m,n}_{sleep}(s_k))$ is above $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$ and the selected action of learning automaton is *Shrinking*, then the reinforcement signal is to reward the selected action.
- If the selected sleeping duration $(T^{m,n}_{sleep}(s_k))$ is above $\widehat{\underline{T}}^{m,n}_{sleep}(s_k)$ and the selected action of learning automaton is *Extending* or *No change*, then the reinforcement signal is to penalize the selected action.

```
1. $LA_k^m$ selects one of its actions randomly based on its action
   probability vector. We refer to the selected action as $\hat{\alpha}$;
2. If ( $\hat{\alpha} = \alpha_1$ ) Then
    2-1. $T_{sleep}^{m,n}(s_k) = T_{sleep}^{m,n-1}(s_k) + ExtendValue$;
    2-2. If ( $T_{sleep}^{m,n}(s_k)$ > MaxSleepDuration) Then
        2-2-1. $T_{sleep}^{m,n}(s_k)$ = MaxSleepDuration;
3. Else If ( $\hat{\alpha} = \alpha_2$ ) Then
    3-1. $T_{sleep}^{m,n}(s_k) = T_{sleep}^{m,n-1}(s_k) - ShrinkValue$;
    3-2. If ( $T_{sleep}^{m,n}(s_k)$ < MinSleepDuration) Then
        3-2-1. $T_{sleep}^{m,n}(s_k)$ = MinSleepDuration;
4. Else If ( $\hat{\alpha} = \alpha_3$ ) Then
    4-1. $T_{sleep}^{m,n}(s_k) = T_{sleep}^{m,n-1}(s_k)$;
```

Fig. 3. Procedure of selection of sleeping duration for round $R^{m,n}$ executed by node s_k.

- If the selected sleeping duration ($T_{sleep}^{m,n}(s_k)$) is equal to $\hat{\underline{T}}_{sleep}^{m,n}(s_k)$ and the selected action of learning automaton is *No change*, then the reinforcement signal is to reward the selected action.
- If the selected sleeping duration ($T_{sleep}^{m,n}(s_k)$) is equal to $\hat{\underline{T}}_{sleep}^{m,n}(s_k)$ and the selected action of learning automaton is *Extending* or *Shrinking*, then the reinforcement signal is to penalize the selected action.

At the end of round $R^{m,n}$, node s_k modifies the sleeping duration ($T_{sleep}^{m,n}(s_k)$) according to the movement patterns of the target points passed through its sensing region during round $R^{m,n}$. This is done using the procedure specified in Fig. 4.

In the above procedure, parameter $\gamma \geqslant 0$ is a constant which controls $T_{sleep}^{m,n}(s_k)$. As γ increases the more energy saving and lower detection rate are obtained (see experiment 5).

```
1. If ( $\hat{\underline{T}}_{sleep}^{m,n}(s_k)$ < MaxSleepDuration) Then
    1-1. $T_{sleep}^{m,n}(s_k) = \hat{\underline{T}}_{sleep}^{m,n}(s_k)$
2. Else If ( $\gamma \cdot \hat{\underline{T}}_{sleep}^{m,n}(s_k)$ < MinSleepDuration) Then
    2-1. $T_{sleep}^{m,n}(s_k)$ = MinSleepDuration
3. Else
    3-1. $T_{sleep}^{m,n}(s_k) = \gamma \cdot \hat{\underline{T}}_{sleep}^{m,n}(s_k)$
```

Fig. 4. Modification made to sleeping duration of node s_k at the end of round $R^{m,n}$.

Fig. 5 gives the complete flowchart of the SALA algorithm.

6. Experimental results

To evaluate the performance of SALA several experiments have been conducted and the results are compared with the results obtained for LEACH [29] from routing layer algorithms, GAF [39] from grid-based algorithms, PEAS [65] from coverage-related algorithms, Proactive Wakeup based on PEAS (PW) [15] from tracking-based algorithms and SALA in which $T_{sleep}^{m}(s_k) = 60$ (s), $m = 1, 2, \ldots, N^R$ which we call hereafter Periodic Sleep (PS). PS is an example of a static scheduling algorithm. The reason for comparison of LEACH with SALA is to demonstrate that for the problem of dynamic point coverage using a simple algorithm like LEACH, in which sensing devices of all sensor nodes are always active, is not efficient due to high energy overhead. We have also compared GAF with SALA as an example of a static scheduling algorithm to show that they are not suitable for dynamic environments such as the one in the dynamic point coverage problem. GAF algorithm has no mechanism for predicting the movement paths of target points to be used for scheduling the sensors.

Unless otherwise stated, the simulation environment is a 100 m $\times$ 100 m area through which 100 sensor nodes are scattered randomly. Sensing ranges (r) of sensor nodes are assumed to be 15 m. For energy consumption of nodes, we use the specifications of MEDUSA II sensor node given in [21]. Based on this specification, the power consumption of a node during the S_AC_A, S_AC_S, S_SC_A and S_SC_S operation modes are 24.58 mW, 9.72 mW, 14.86 mW, 0.02 mW, respectively. Energy required to switch a node from one operation mode to another operation mode is assumed to be negligible.

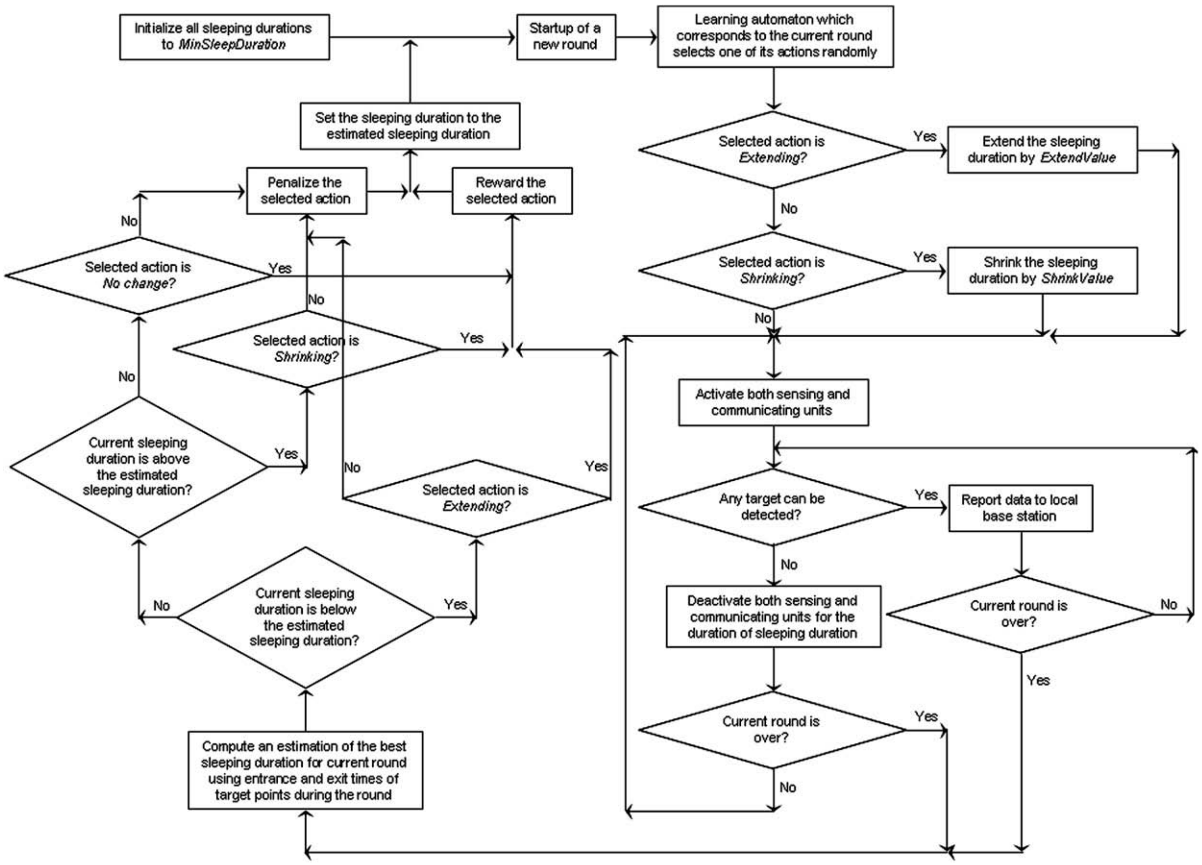
Initialize all sleeping durations to *MinSleepDuration*
Startup of a new round
Learning automaton which corresponds to the current round selects one of its actions randomly
Selected action is *Extending*?
Yes
Extend the sleeping duration by *ExtendValue*
No
Selected action is *Shrinking*?
Yes
Shrink the sleeping duration by *ShrinkValue*
No
Activate both sensing and communicating units
Any target can be detected?
Yes
Report data to local base station
No
Deactivate both sensing and communicating units for the duration of sleeping duration
Current round is over?
No
Yes
Current round is over?
Yes
No
Set the sleeping duration to the estimated sleeping duration
Penalize the selected action
Reward the selected action
Selected action is *No change*?
No
Yes
Selected action is *Shrinking*?
No
Yes
Current sleeping duration is above the estimated sleeping duration?
No
Yes
Selected action is *Extending*?
No
Yes
Current sleeping duration is below the estimated sleeping duration?
No
Yes
Compute an estimation of the best sleeping duration for current round using entrance and exit times of target points during the round

Fig. 5. Flowchart of the SALA algorithm.

MinSleepDuration MaxSleepDuration, *ExtendValue*, *ShrinkValue*, and γ are set to 0 s, 600 s, 2 s, 2 s, and .01, respectively. Learning parameters α and β are both set to .01.

In all experiments, following different types of moving target points are considered to exist in the area of the network:

- **Constant Events:** constant events have constant start times and durations. At the startup of the simulation, number of events is selected uniformly and randomly from the range [1,200]. The start time of each event is a constant which is uniformly and randomly selected from the range [1,86,400] (number of seconds per day) and the duration of each event is a constant which is uniformly and randomly selected from the range [60,900] (at least 1 min and at most 15 min). Each event has a static occurrence position which is selected uniformly and randomly in Ω. Events are repeated with the same start times and durations every epoch of the simulation.
- **Noisy Events:** noisy events are like constant events except that the number of events, their start times and their durations are affected in every epoch by a normally distributed random noise.
- **Poisson Events:** the occurrences of Poisson events follow the Poisson distribution. To generate such events, a number of Poisson random number generators (selected uniformly and randomly from the range [1,100]) are used. Each Poisson random number generator separately generates events in a randomly selected location in Ω.
- **Straight Path Moving Objects:** Each straight path moving object has a start position, stop position and a velocity. A straight path moving object starts from its start position and moves directly towards its stop position using its velocity. The start and stop positions are selected uniformly and randomly in Ω and the velocity is selected uniformly and randomly from the range $[0, MaxVelocity]$. There are 24 straight path moving objects each moves only in one of the rounds R^m, $m = 1,2,\ldots,N^R$ and repeats its movement path in every epoch.
- **Complex Path Moving Objects:** these target points are like straight path moving objects except that they are not moving from their start positions directly towards their stop positions. Instead, each complex path moving object has an ordered list of points $\langle(x^{start},y^{start}), (x^1,y^1), (x^2,y^2),\ldots,(x^{Stop},y^{Stop})\rangle$. A complex path moving object starts from its start point and moves directly towards (x^1,y^1) using its velocity. Whenever the target point reaches (x^1,y^1), it changes its direction towards (x^2,y^2). This movement continues until the target point reaches its stop position. The number of intermediate points for each moving object is selected randomly from the range [1,5].

All simulations have been implemented using J-Sim simulator [92]. J-Sim is a java based simulator which is implemented on top of a component-based software architecture. Using this component-based architecture, new

Table 1
Parameters used for simulating different target points in experiment 1.

Target point	Parameter	Distribution	Value
Constant events	Number of events	Uniform	[1,200]
	Event start time	Uniform	[1,86,400]
	Event duration	Uniform	[60,900]
Noisy events	Number of events	Normal	μ = randomly selected from the range [1,200] σ = 10
	Event start time	Normal	μ = randomly selected from the range [1,86,400] σ = 120
	Event duration	Normal	μ= Randomly selected from the range [60,900] σ = 120
Poisson events	λ	Poisson	1.5
	Event duration	Constant	600
Straight path moving objects	*MaxVelocity*	Uniform	[0,.5]
Complex path moving objects	*MaxVelocity*	Uniform	[0,.5]
	Number of intermediate points	Uniform	[1,5]

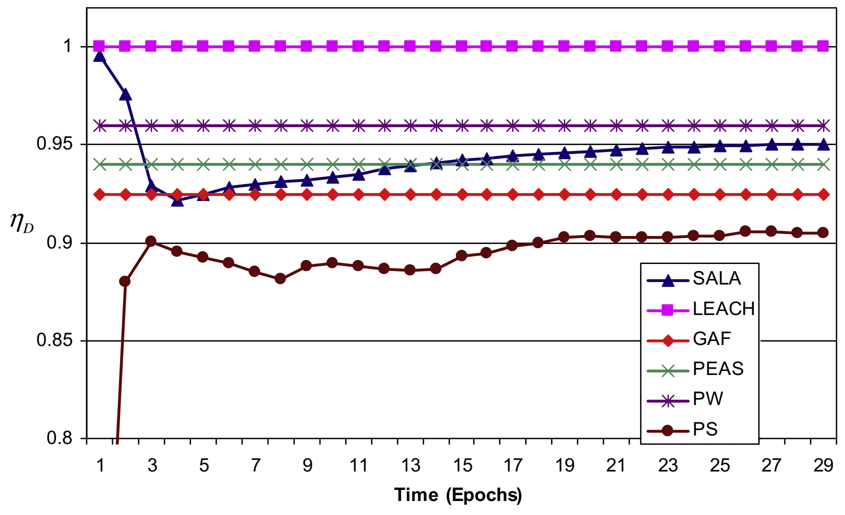

Fig. 6. Network detection rate (η_D).

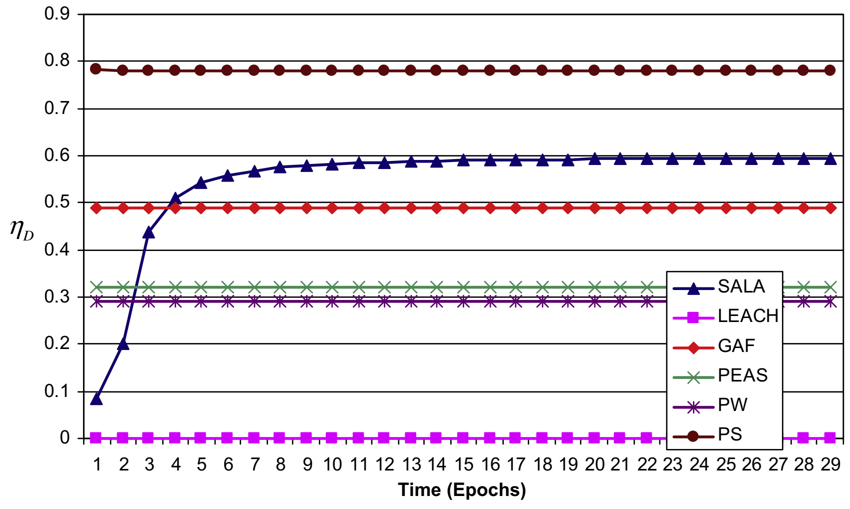

Fig. 7. Network sleep rate (η_S).

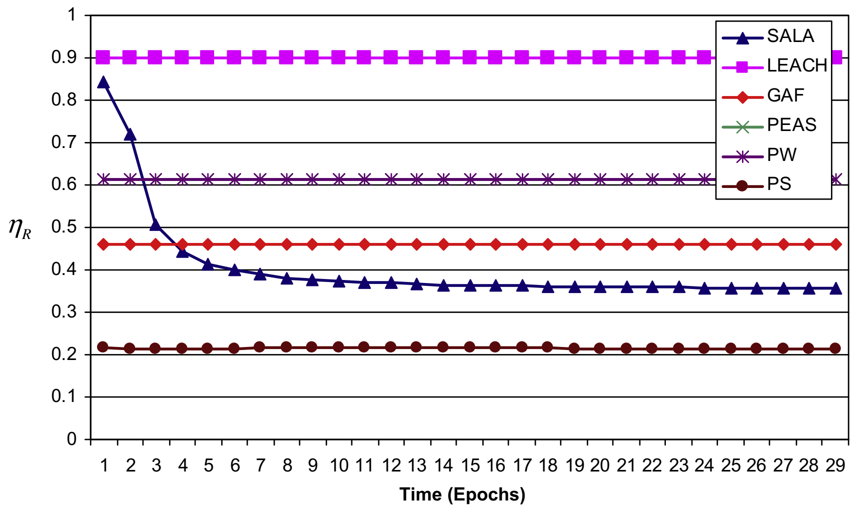

Fig. 8. Network redundant active rate (η_R).

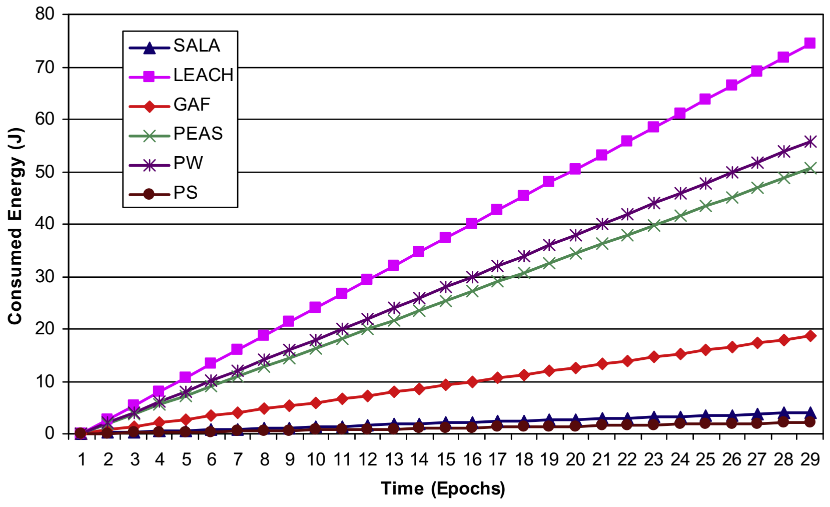

Fig. 9. Mean consumed energy for a node.

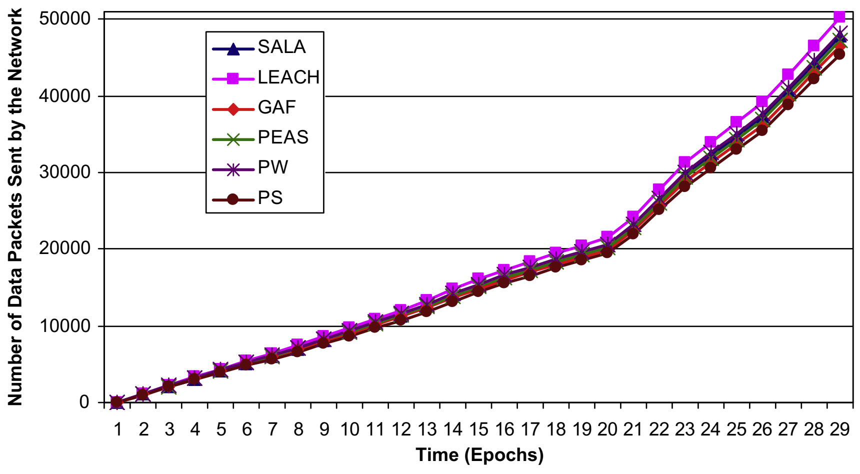

Fig. 10. Number of data packets transmitted by the network to the sink node.

protocols and algorithms can be designed, implemented and tested in the simulator without any changes to the rest of the simulator's codes. J-Sim simulator is written in java and hence has a number of benefits: (1) it enables to simulate large scale networks, (2) simulating complex experimental environments is easy in J-Sim, (3) it supports simulation of heterogeneous networks with different kinds of sensor nodes, (4) it is easy to add new battery models, radio models, energy models and sensing models to J-Sim, (5) implementation and debugging of new protocols and algorithms in J-Sim can be performed within smart java based IDEs, (6) simulations are platform-independent and can be performed on different operating systems and machines. J-Sim has been recently used as the network simulator in many research projects [114–122].

All reported results are averaged over 50 runs. We have used CSMA as the MAC layer protocol, free space model as the propagation model, binary sensing model and Omni-directional antenna.

6.1. Experiment 1

In this experiment, SALA is compared with GAF, LEACH, PEAS, PW and PS algorithms in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) defined by Eq. (6), mean energy consumption of nodes, number of data packets sent by the network to the sink node, number of data packets received at the sink node, delivery rate and delivery delay of data packets at the sink node

$$\eta_R = \frac{\sum_{s_k}\left(\frac{\tau_{s_k} - \sum_{tp_i \in TP} \tau_{s_k.tp_i}}{T}\right)}{N}. \quad (6)$$

Table 1 gives the parameters used for simulating the moving target points.

Figs. 6–13 give the results of comparison in terms of η_D, η_S, η_R, mean consumed energy of all nodes of the network, number of data packets sent by the network to the sink node, number of data packets received at the sink node, delivery rate and delivery delay of data packets at the sink node, respectively. As it can be seen from Fig. 6, network detection rate in SALA outperforms PEAS, GAF and PS, and about 5% worse than the LEACH for which the network detection rate is equal to 1. From this figure, it can also be concluded that it takes about 20 epochs for the SALA algorithm to stabilize its detection rate and learn the movement patterns of target points. Note that in every epoch of SALA algorithm, each learning automaton performs one action and receives one feedback from the environment.

Fig. 7 shows that for SALA the network sleep rate (η_S) is about 0.6 which is better than all the existing algorithms except PS algorithm. The reason that SALA performs worse than PS algorithm is that in PS all nodes are statically scheduled to periodically sleep for 60(s). Of Course, as indicated in Fig. 6, this results in a poor network detection rate for PS.

Fig. 8 shows that the fraction of the times during which nodes are in active mode and no target point is in their

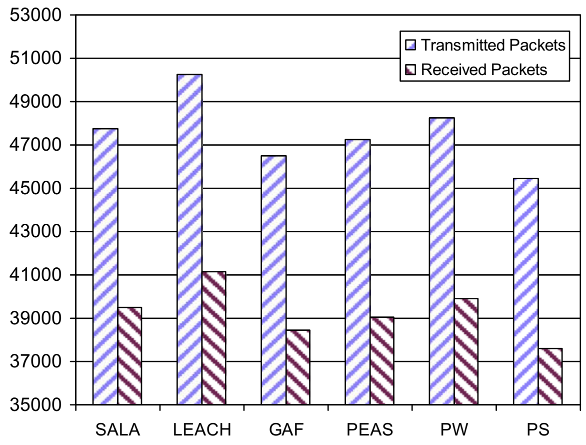

Fig. 12. Number of data packets transmitted and received within the network in different algorithms.

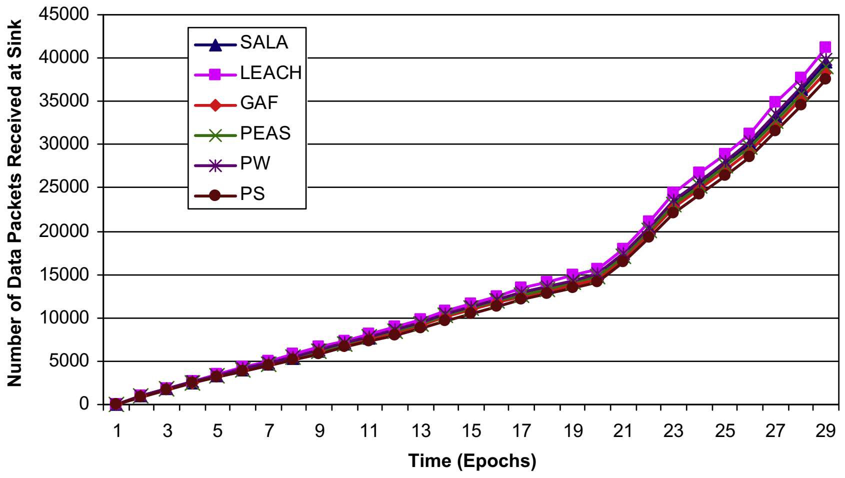

Fig. 11. Number of data packets received at the sink node.

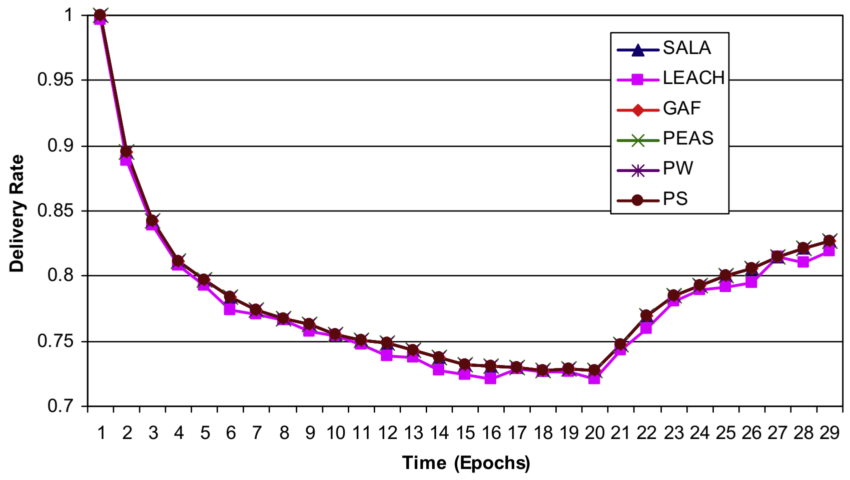

Fig. 13. Delivery rate of data packets at the sink node.

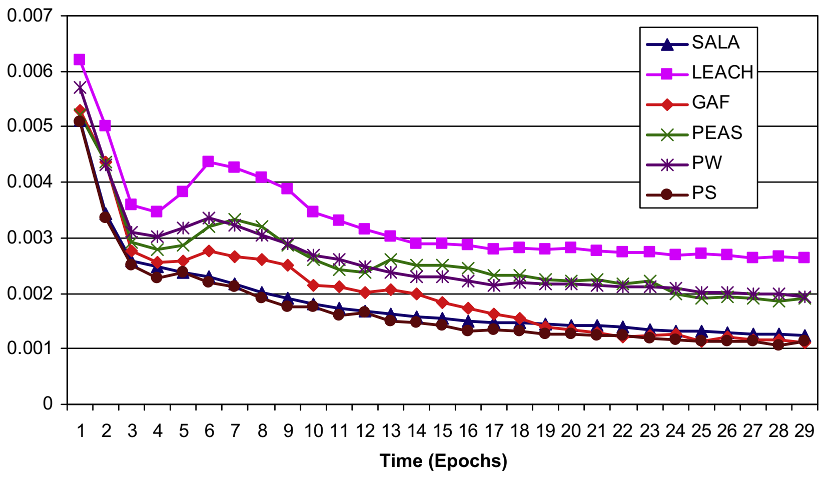

Fig. 14. Delivery delay of data packets at the sink node.

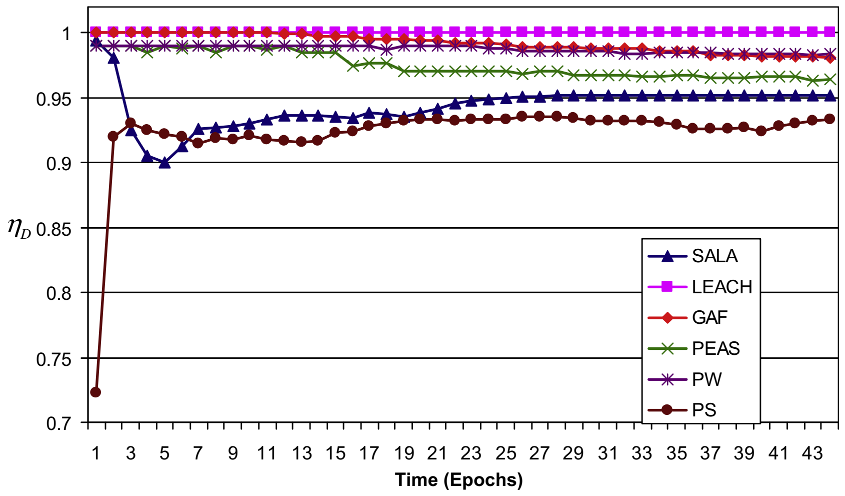

Fig. 15. Network detection rate (η_D).

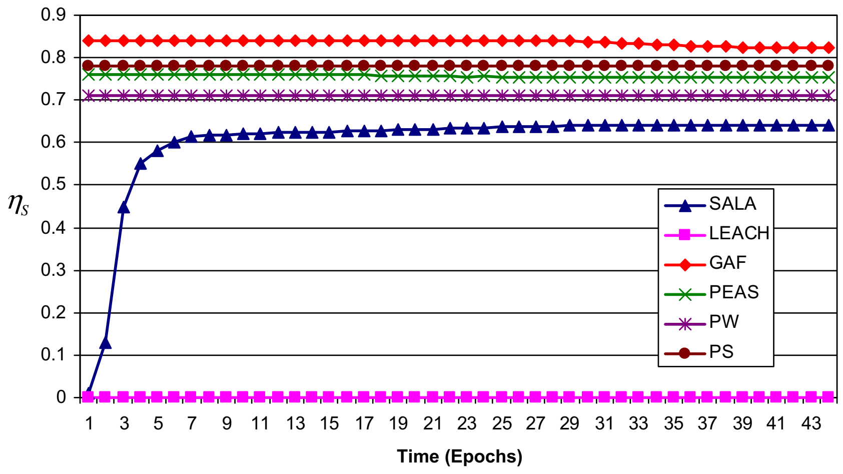

Fig. 16. Network sleep rate (η_S).

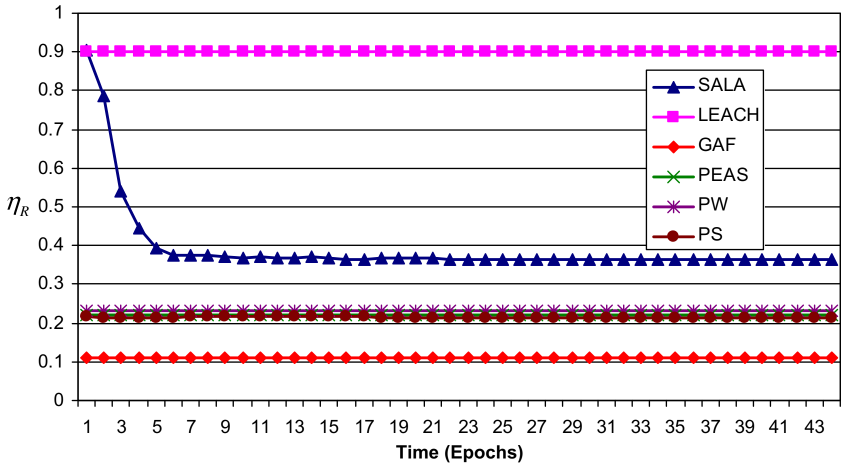

Fig. 17. Network redundant active rate (η_R).

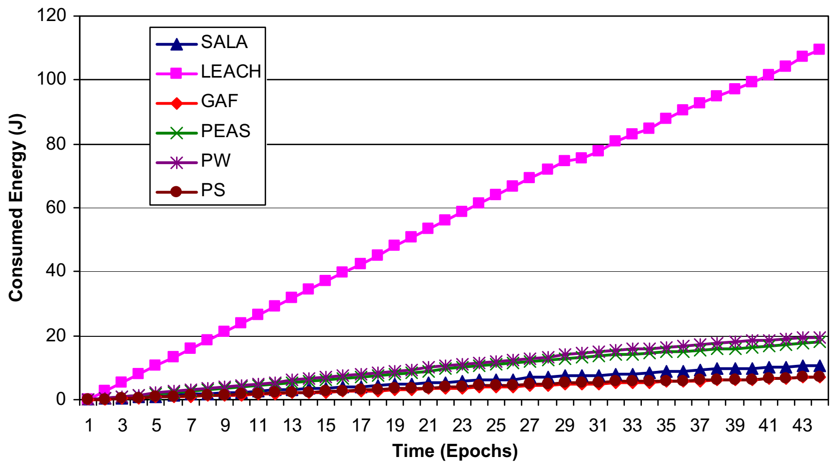

Fig. 18. Mean consumed energy for a node.

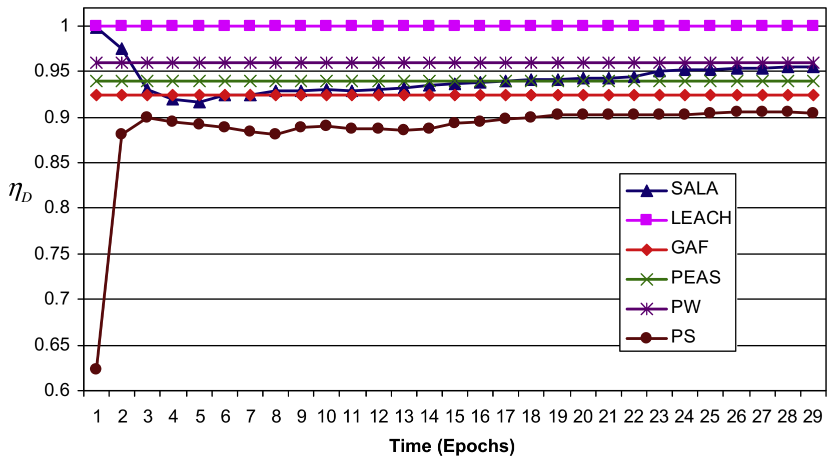

Fig. 19. Network detection rate (η_D).

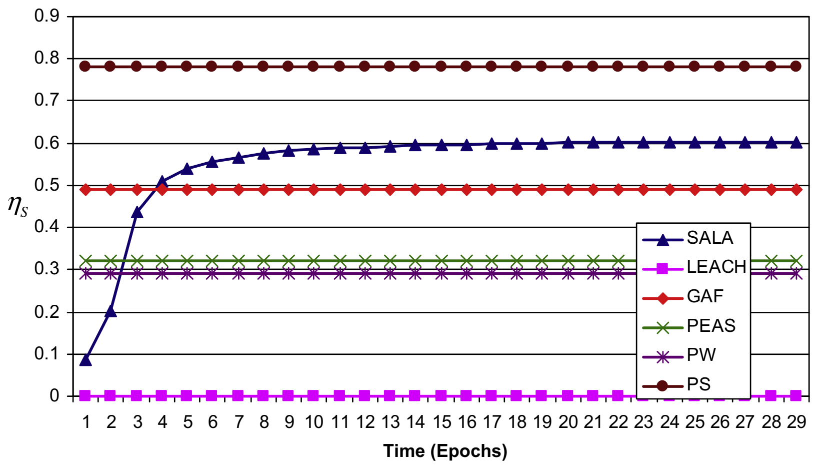

Fig. 20. Network sleep rate (η_S).

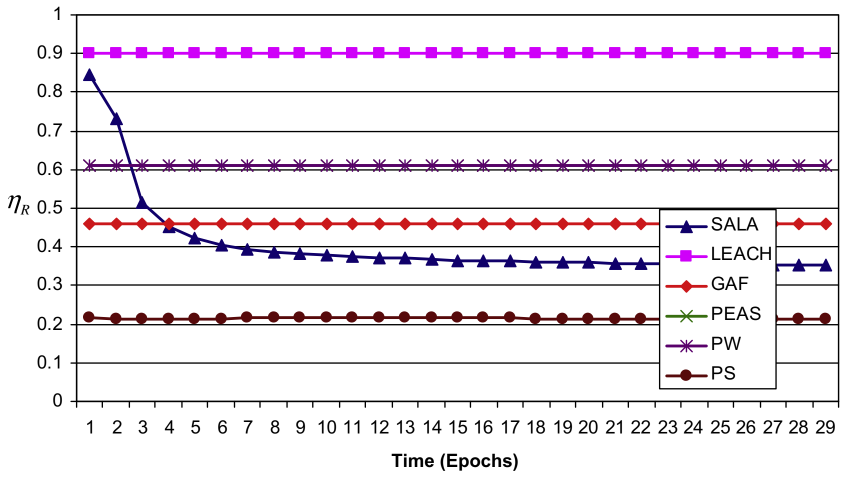

Fig. 21. Network redundant active rate (η_R).

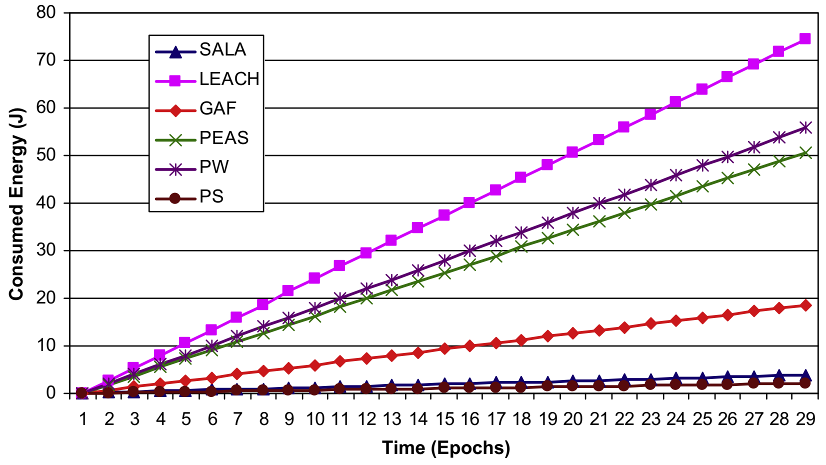

Fig. 22. Mean consumed energy for a node.

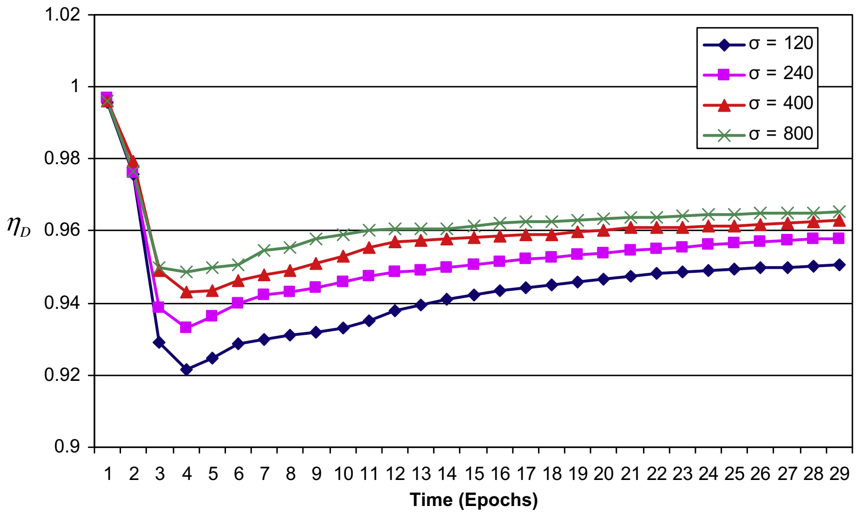

Fig. 23. Network detection rate (η_D).

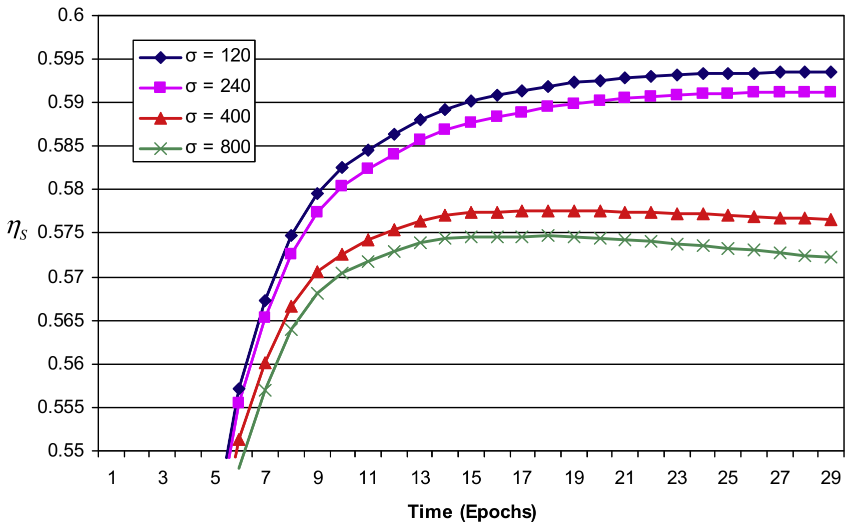

Fig. 24. Network sleep rate (η_S).

sensing region is about .35 on average for SALA which is better than all the other methods except PS. This is again due to the fact that in PS all nodes are statically scheduled to periodically sleep for 60(s).

Fig. 9 shows that for SALA, the mean energy consumption is lower than the mean energy consumption for all the existing algorithms except for PS algorithm.

From the results given in Figs. 6–9 we can conclude that the LEACH algorithm has the highest network detection rate and energy consumption and PS algorithm has the lowest network detection rate and energy consumption among the above mentioned algorithms. High network detection rate and energy consumption of LEACH is due to the fact that in this algorithm, all sensor nodes are always in active operation mode. Similarly, low network detection rate and energy consumption of PS algorithm is due to the fact that in this algorithm, sensor nodes stay in sleep operation mode more often than the other mentioned algorithms. From this we may say that in order to obtain a higher network detection rate, more energy must be paid. GAF, PEAS and PW algorithms are not always able to fully cover the entire area and hence a network detection rate less than 1 may be experienced.

Fig. 10 shows that the number of data packets transmitted by the sensor nodes in the network to the sink is highly dependent on the network detection rate. Higher detection rate means higher number of data packets transmitted to the sink node. This is due to the fact that a data packet is transmitted by a sensor to the sink when it detects a target point. Note that the total number of packets transmitted to the sink by the sensor nodes for PS algorithm is the highest and for LEACH algorithm is the lowest as compared with other algorithms.

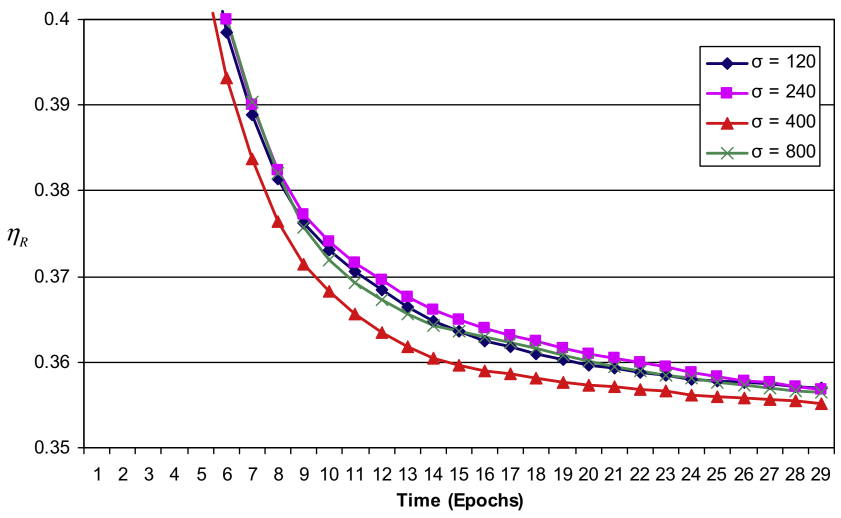

Fig. 25. Network redundant active rate (η_R).

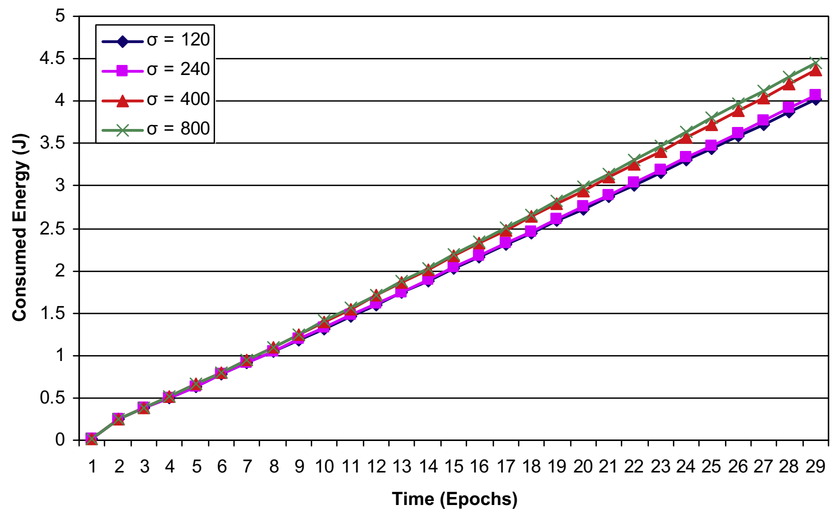

Fig. 26. Mean consumed energy for a node.

Fig. 11 shows the number of data packets received by the sink node for different algorithms. Note that the behaviors of all the algorithms with respect to the number of data packets received by the sink are similar. This is because of the fact that the data gathering process which includes sending data packets by the sensors to the local base stations and then forwarding the received data packets by the local base stations to the sink node, for all algorithms is the same.

Fig. 12 compares the number of data packets transmitted and received within the network at the end of the simulation for different algorithms. As it can be seen from this figure, number of data packets transmitted by the network to the sink node in different algorithms is in agreement with the network detection rates of these algorithms.

Delivery rates of data packets at the sink node in different algorithms are shown in Fig. 13. Again the behaviors of all algorithms with respect to delivering data packets to the sink node are almost the same. This is again because of similar data gathering process used by all algorithms.

Finally, Fig. 14 shows the delivery delay of data packets at the sink node for different algorithms. As it is seen, SALA has almost the lowest delivery delay. The algorithms in the decreasing order of their delivery delays are LEACH, PW, PEAS, GAF, SALA and PS. Delivery delay for LEACH is high because in LEACH a senor with a data packet has to delay its data transmission until its scheduled time. Delivery delays for PEAS and PW algorithms are high because of a high number of controlling packets which needs to be transmitted.

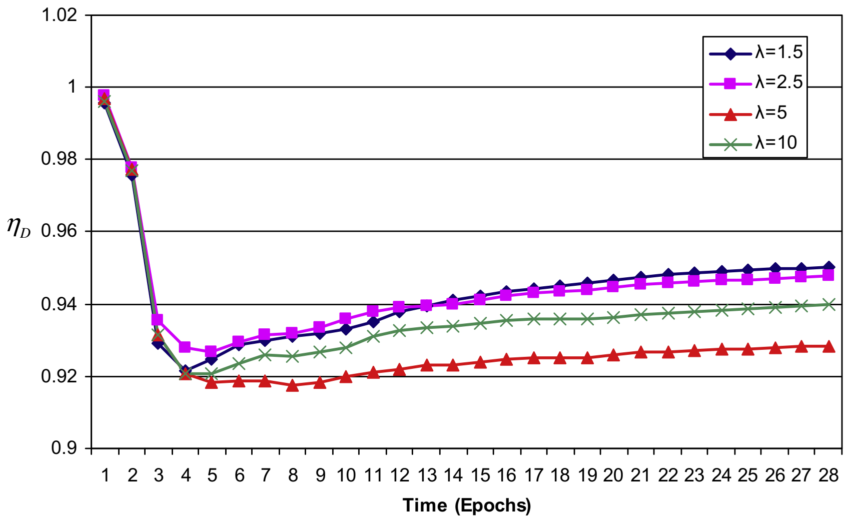

Fig. 27. Network detection rate (η_D).

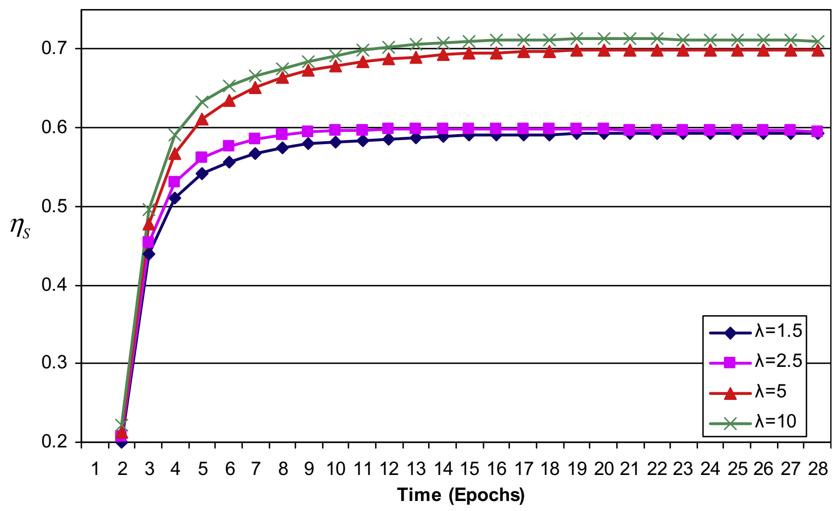

Fig. 28. Network sleep rate (η_S).

6.2. Experiment 2

In this experiment we compare SALA with LEACH, GAF, PEAS, PW and PS in terms of η_D, η_S, η_R and mean consumed energy for dense networks. The simulation settings of experiment 1 are also used for this experiment except for the number of sensor nodes which is set to 1000. Figs. 15–18 show the results of this experiment. The results indicate that GAF, PEAS and PW outperform SALA. This is because these algorithms unlike SALA use the operation modes of neighbors of a node to schedule that node. In dense networks, the average number of neighbors for a node is higher and hence such algorithms schedule more sensor nodes to be in sleep operation mode because of the availability of more information about the environment of the sensor nodes.

6.3. Experiment 3

This experiment is conducted to study the behavior of SALA in comparison to LEACH, PEAS, PW and PS algorithms in terms of η_D, η_S, η_R and mean consumed energy for large networks. The simulation settings of experiment 1 are also used for this experiment except for the number of sensor nodes which is set to 900 and the network area which is a 300 m × 300 m rectangular area. Figs. 19–22 give the simulation results of this experiment. From the results of this experiment we can say that the number of nodes and dimension of the network do not affect the performance of SALA significantly. The reasons that the performance does not vary significantly as these two parameters change are (1) the network considered in this paper is a clustered network with cluster heads as

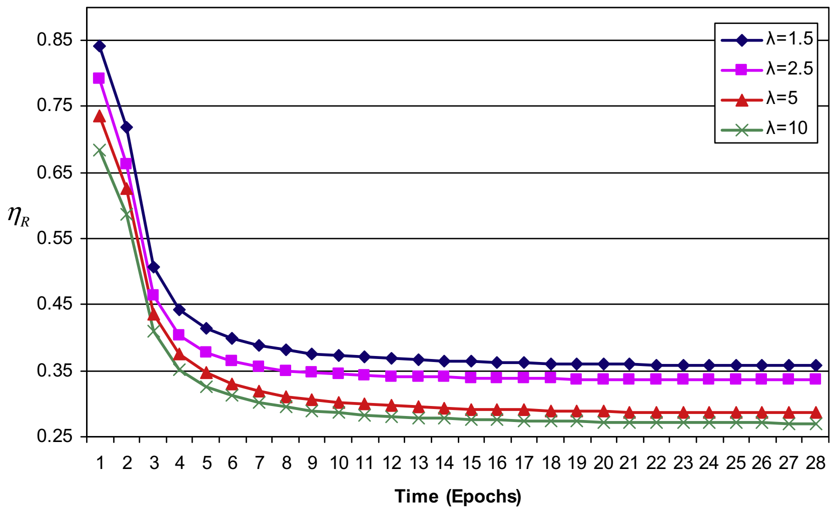

Fig. 29. Network redundant active rate (η_R).

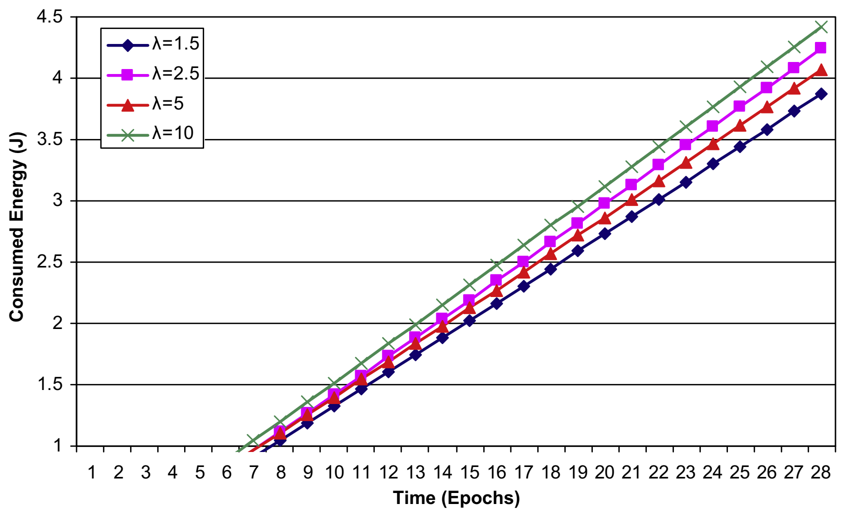

Fig. 30. Mean consumed energy for a node.

rechargeable nodes and (2) the algorithm executed in each sensor node uses local information within the node's cluster.

6.4. Experiment 4

In this experiment, we study the effect of the noise level in noisy events on the performance of SALA. For this purpose, we change the standard deviation of the normally distributed noise from 120 to 800. Figs. 23–26 show the results of this study. As it is shown, increasing the noise level does not affect the performance of SALA substantially. The robustness of SALA is because of two reasons: (1) utilization of a learning mechanism for making SALA adaptable to the environmental changes and (2) utilization of a controlling mechanism by the learning mechanism in order to have a controlled adaptation of the environmental changes. Controlled adaptation is achieved by proper choices of *MinSleepDuration* and *MaxSleepDuration* parameters.

6.5. Experiment 5

In this experiment, we evaluate the performance of SALA when the mean number of Poisson events per round (λ) varies. Figs. 27–30 show the results of this experiment when λ varies from 1.5 to 10. It can be seen from these figures that the performance of SALA does not depend significantly on the value of λ.

6.6. Experiment 6

This experiment is designed to evaluate the performance of SALA when the *MaxVelocity* of moving objects

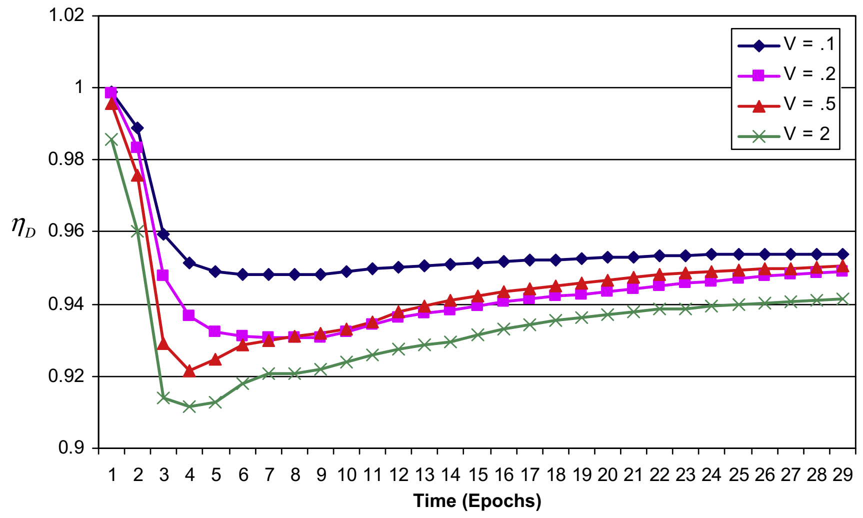

Fig. 31. Network detection rate (η_D).

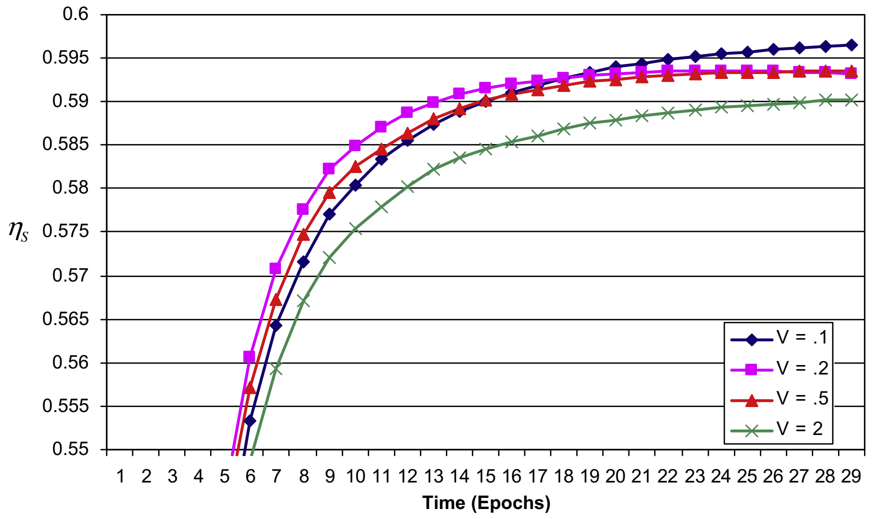

Fig. 32. Network sleep rate (η_S).

varies. For this purpose, we change *MaxVelocity* from .1 to 2. Figs. 31–34 show the results in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and the mean energy consumption of nodes. As indicated by these figures, the changes in the *MaxVelocity* do not affect the performance of SALA significantly.

6.7. Experiment 7

This experiment is conducted to study the effect of the parameter γ on the performance of SALA. In this experiment we use the target points of experiment 1. We also let γ varies in the range [0,.2]. Figs. 35–38 show the results of this experiment in terms of network detection rate (η_D), network sleep rate (η_S), network redundant active rate (η_R) and the mean energy consumption of nodes. From these figures we can conclude that (1) lower values of γ results in more network detection rate at the expense of more energy consumption, more redundant active times and less sleep times and (2) higher values of γ results in more energy saving in the network at the expense of poor network detection rate.

Fig. 39 gives the network detection rate versus mean energy consumption of nodes as γ varies. As it can be seen from this figure, changing γ in the range [.01,.5] affects both the network detection rate and the mean energy consumption of the nodes. Changing γ outside of this range does not affect these two parameters significantly. To explain this phenomenon, we use the concept of sensitivity of a node to γ. A node is sensitive to γ if changing in the value of γ changes its detection rate and energy consumption. From the algorithm, it is evident that a node s_k is sensitive to γ if $\underline{\widehat{T}}^{m,n}_{sleep}(s_k) > MaxSleepDuration$ and $\gamma \cdot \underline{\widehat{T}}^{m,n}_{sleep}(s_k) >$

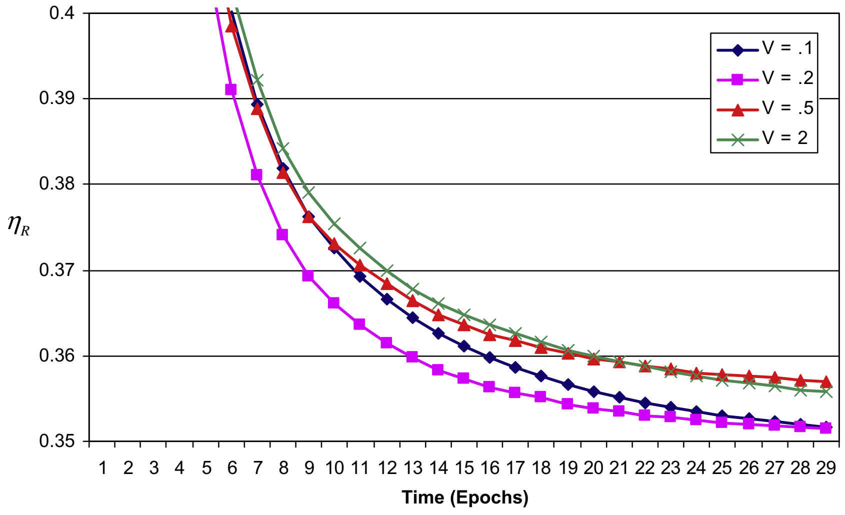

Fig. 33. Network redundant active rate (η_R).

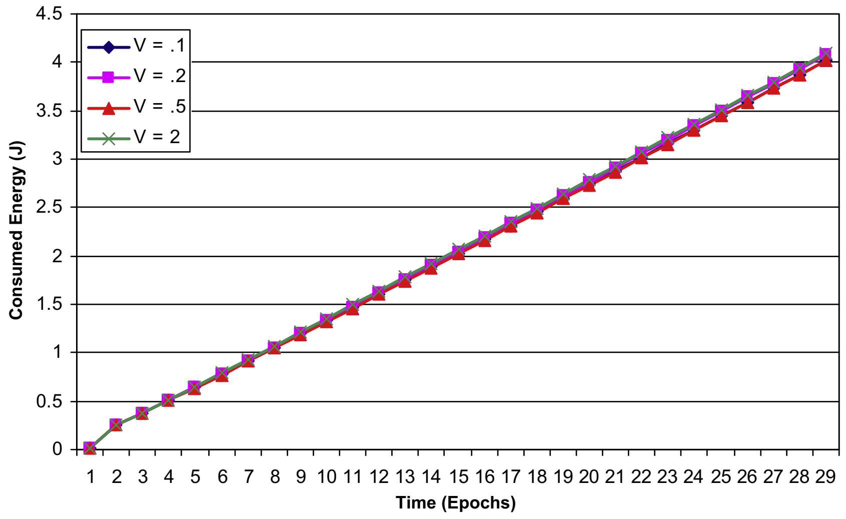

Fig. 34. Mean consumed energy for a node.

MinSleepDuration. Changing γ out of [.01,.5] range, decreases the number of sensitive nodes. This is because choosing large values for γ results in $\widehat{T}^{m,n}_{sleep}$ to be selected equal to *MaxSleepDuration* and choosing small values for γ results in $\widehat{T}^{m,n}_{sleep}$ to be selected equal to *MinSleepDuration*. Decreasing the number of sensitive nodes causes the rate of changes in the network detection rate and the mean energy consumption of nodes to decrease. For $\gamma = 0$ or $\gamma \geqslant \frac{MaxSleepDuration}{MinSleepDuration}$, all nodes become insensitive to γ which causes the mean energy consumption of nodes and the detection rate to remain fixed during the lifetime of the network.

Fig. 40 compares SALA with LEACH, GAF, PEAS and PW in terms of the network detection rate and the mean energy consumption of nodes. This figure shows that there exists a value for $\gamma(\gamma = .001)$ below which SALA outperforms GAF, PEAS and PW in terms of network detection rate and mean energy consumption of nodes. Comparing SALA with LEACH in which nodes are always active, the proposed algorithm reaches a network detection rate very near to that of LEACH by consuming only about 9% of the energy consumed by LEACH.

Determination of γ for an application is very crucial and is a matter of cost versus precision. For higher network detection rate, higher price must be paid. For example, as it is indicated in Fig. 40, for network detection rate to be above 95%, each node must consume about 6.503(J) on average during the lifetime of the network whereas to obtain a network detection rate of about 85%, each node must consume only about 1.981(J) on average. Reaching highest precision which can be obtained by setting γ to zero results in maximum energy consumption by the network.

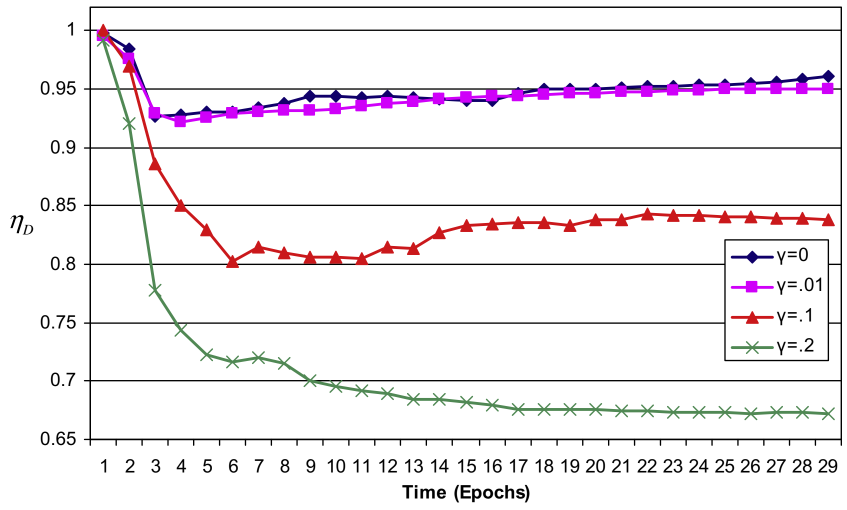

Fig. 35. Network detection rate (η_D).

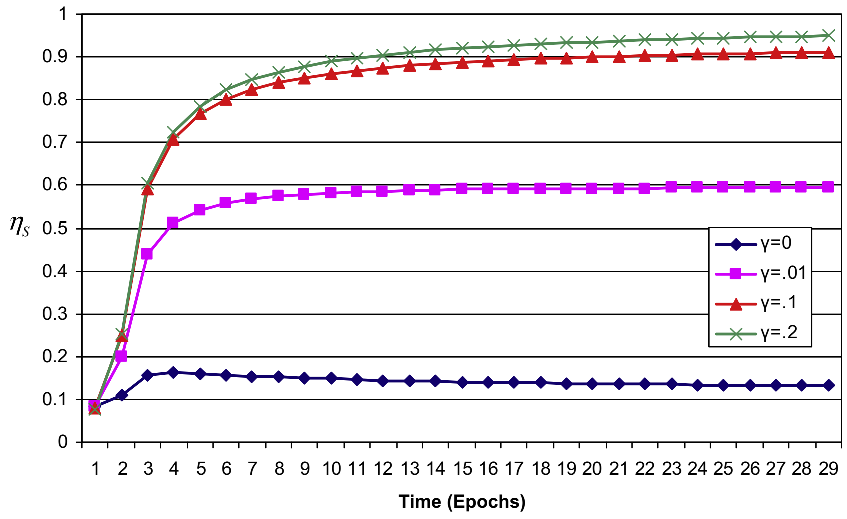

Fig. 36. Network sleep rate (η_S).

6.8. Summary of results

In this study, we compared the performance of SALA algorithm in terms of network detection rate and mean consumed energy with LEACH, GAF, PEAS, PW and PS algorithms. Comparisons were made for small, large, dense and non-dense networks and showed that:

– In terms of mean consumed energy, SALA and PS algorithms outperform almost all of the other algorithms in any of the small, large, dense and non-dense networks. Of course, superiority of these two algorithms over other algorithms in non-dense networks is significantly higher than in dense networks. The only exception to this superiority is the mean consumed energy of GAF algorithm in dense networks which is almost equal to that of SALA and PS algorithms. Note that PS algorithm is a simple case of SALA algorithm in which sleeping duration of any node during any round is equal to *MinSleepDuration*.

– In terms of network detection rate, SALA and LEACH algorithms are the only two algorithms that have robust performance over networks of different sizes and densities. Network detection rates of GAF, PEAS, PW and PS algorithms in non-dense networks are significantly lower than in dense networks.

– There is a tradeoff between network detection rate and mean consumed energy; for a higher network detection rate, more energy must be consumed. SALA algorithm unlike other algorithms has a parameter (γ) for controlling this tradeoff. There exists a value for parameter $\gamma(\gamma = .001)$ below which SALA reaches a network detec-

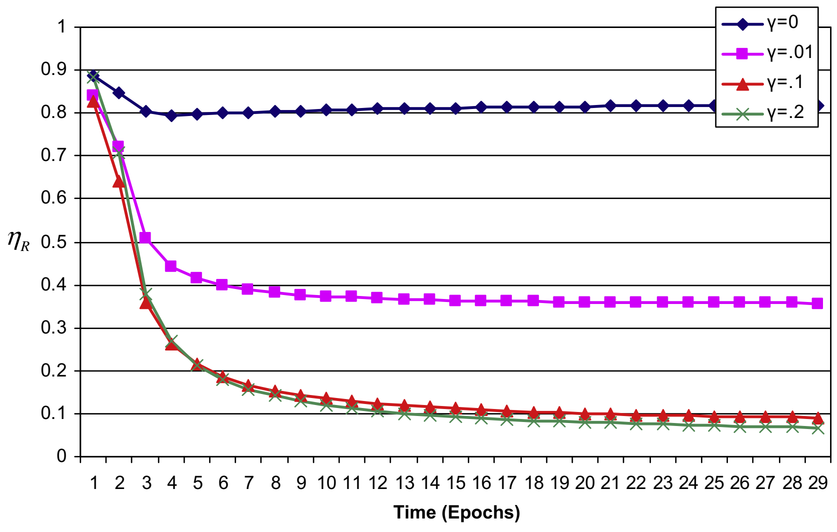

Fig. 37. Network redundant active rate (η_R).

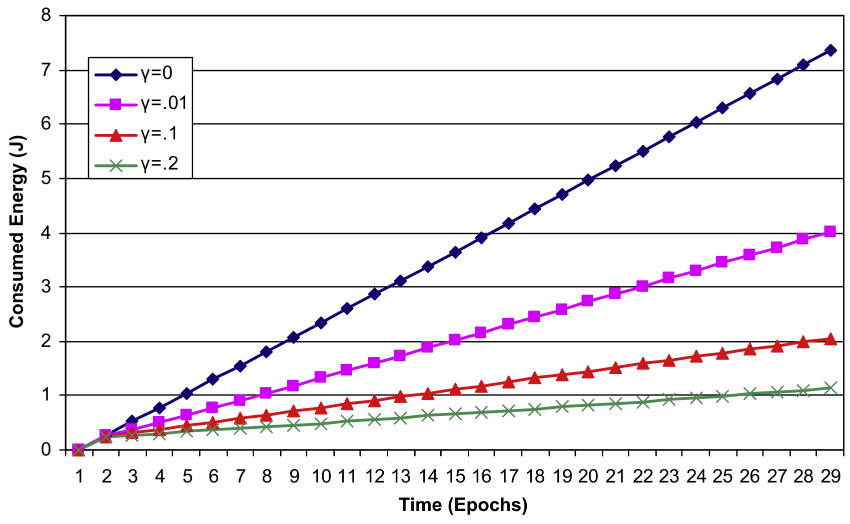

Fig. 38. Mean consumed energy for a node.

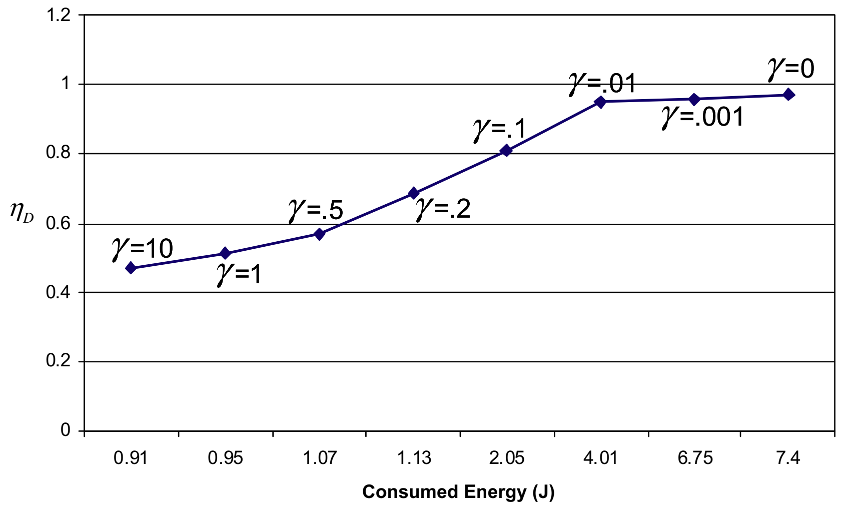

Fig. 39. Network detection rate vs. mean energy consumption of nods as γ varies.

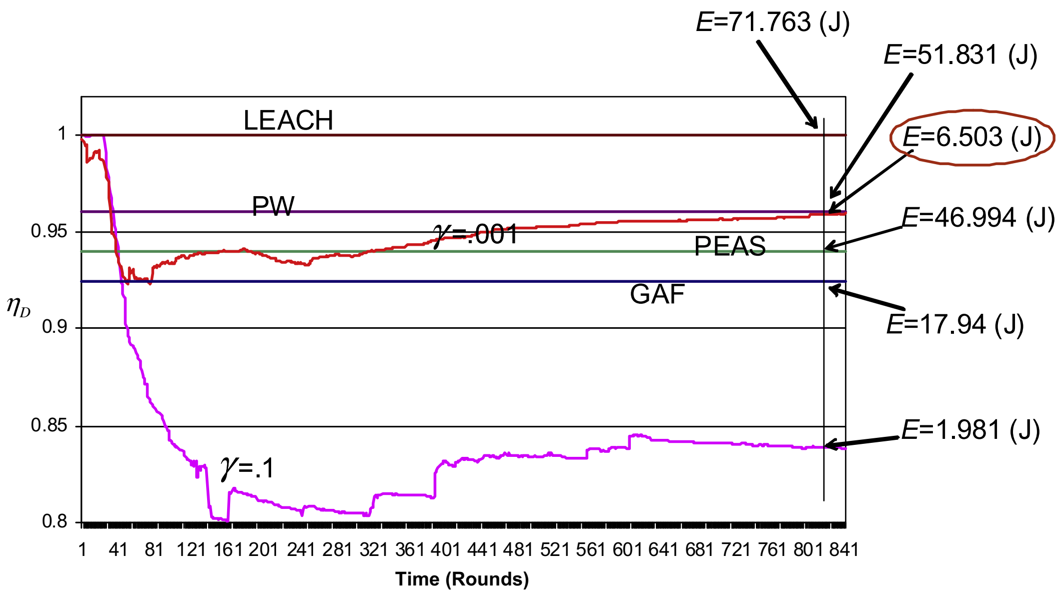

Fig. 40. Comparison of the proposed algorithm with LEACH, GAF, PEAS and PW in terms of the network detection rate and the mean energy consumption of nodes.

tion rate very near to that of LEACH and outperforms GAF, PEAS, PW and PS. On the other hand, there exists a value for parameter γ ($\gamma = .1$) above which SALA outperforms LEACH, GAF, PEAS, PW and PS in terms of mean consumed energy. In fact determination of γ for an application is very crucial and is a matter of cost versus precision.

Furthermore, a number of experiments conducted to study the behavior of the SALA algorithm when parameters of movement patterns of target points, such as the velocity of moving targets and the frequency of occurrences of events, change. These experiments showed that such changes do not affect the performance of SALA algorithm in terms of network detection rate and mean consumed energy substantially. In other words, SALA algorithm is robust against changes in the parameters of the target points' movement patterns.

7. Conclusion

In this paper, an algorithm based on learning automata called SALA for the problem of detecting and monitoring a number of moving target points in an area by sensor networks was proposed. We dealt with this problem by designing a dynamic scheduling strategy for activating and deactivating nodes based on the movement patterns of target points. Unlike other solutions to this problem in which some sort of notification messages are used to activate sleep nodes, in SALA, no notification message is exchanged between sensor nodes and instead, each node separately learns the best scheduling strategy using its set of learning automata. Using this strategy, the nodes do not wait for notification messages from their neighbors which results in a large amount of energy saving.

The experimental results showed that there exists a value for parameter $\gamma(\gamma = .001)$ below which SALA outperforms GAF, PEAS and PW, in terms of network detection rate and mean energy consumption of nodes. The experiments also showed that if γ chosen properly SALA reaches a network detection rate very near to that of LEACH, a method in which nodes are always active, by consuming only about 9% of the energy consumed by LEACH. In fact determination of γ for an application is very crucial and is a matter of cost versus precision. Furthermore, the experimental results showed that SALA is robust against changes in the parameters of the target points' movement patterns such as the velocity of moving targets and the frequency of occurrences of events.

Appendix A

Before we prove the main theorem we state and prove several lemmas.

Lemma 1. *If in a sensor network we assume that*

- *node s_k be activated at the beginning of round $R^{m,n}$,*
- *node s_k immediately switches to the sleep operation mode if it cannot sense any target point,*

then node s_k can fully detects all the target points passing through its sensing region during round $R^{m,n}$ if and only if $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$ given by Eq. (5).

Proof of Lemma 1. The proof of necessity condition: We prove the necessity condition of the lemma by induction on the order of arrival of target points into the sensing region of node s_k.

Basis of induction: We show that the first target point can be detected. Let $tp^{m,n}_1(s_k)$ be the first target point passing through the sensing region of the node s_k during round $R^{m,n}$. Target point $tp^{m,n}_1(s_k)$ enters the sensing region of node s_k either before $(\tau^E_{tp^{m,n}_1}(s_k) \leqslant \tau^S_{R^{m,n}}(s_k))$ or after $(\tau^E_{tp^{m,n}_1}(s_k) > \tau^S_{R^{m,n}}(s_k))$ the beginning of the round. Considering $\tau^E_{tp^{m,n}_1}(s_k) \leqslant \tau^S_{R^{m,n}}(s_k)$, s_k is able to fully detect target point $tp^{m,n}_1(s_k)$ due to the fact that node s_k is activated at the beginning of round $R^{m,n}$ (the first assumption of the lemma). Considering $\tau^E_{tp^{m,n}_1}(s_k) > \tau^S_{R^{m,n}}(s_k)$ and the second assumption of the lemma we can conclude that node s_k switches to the sleep operation mode at $\tau^S_{R^{m,n}}(s_k)$. Now let $t^{m,n}_{d-e,0}(s_k) = \tau^E_{tp^{m,n}_1}(s_k) - \tau^S_{R^{m,n}}(s_k)$ be the difference between $\tau^S_{R^{m,n}}(s_k)$ and the time that target point $tp^{m,n}_1(s_k)$ enters the sensing region of node s_k $(\tau^E_{tp^{m,n}_1}(s_k))$. If $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$, then $t^{m,n}_{d-e,0}(s_k)$ is divisible by $T^{m,n}_{sleep}(s_k)$. Since node s_k switches to sleep operation mode at time $\tau^S_{R^{m,n}}(s_k)$ and $t^{m,n}_{d-e,0}(s_k)$ is divisible by $T^{m,n}_{sleep}(s_k)$, then it can be concluded that it certainly wakes up at $\tau^E_{tp^{m,n}_1}(s_k)$ and fully detects $tp^{m,n}_1(s_k)$.

Induction hypothesis: The ith target point that enters the sensing region of node $s_k(tp^{m,n}_i(s_k))$ can be fully detected.

Induction proof: The $(i+1)$ th target point that enters the sensing region of node $s_k(tp^{m,n}_{i+1}(s_k))$ can be fully detected. We will show that if node s_k fully detects the ith target point $tp^{m,n}_i(s_k)$, then it can fully detect the next target point that enters into its sensing region $(tp^{m,n}_{i+1}(s_k))$ as well. Assume that the target point $tp^{m,n}_i(s_k)$ has been fully detected by node s_k. By the second assumption of the lemma, node s_k switches to the sleep operation mode at the departure time of $tp^{m,n}_i(s_k)(\tau^D_{tp^{m,n}_i}(s_k))$. Let $t^{m,n}_{d-e,i}(s_k) = \tau^E_{tp^{m,n}_{i+1}}(s_k) - \tau^D_{tp^{m,n}_i}(s_k)$ be the difference between the time that the ith target point $tp^{m,n}_i(s_k)$ departs the sensing region of node s_k and the time that the next target point $tp^{m,n}_{i+1}(s_k)$ enters into the sensing region of node s_k. If $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$, then $t^{m,n}_{d-e,i}(s_k)$ is divisible by $T^{m,n}_{sleep}(s_k)$. Since node s_k switches to sleep operation mode at $\tau^D_{tp^{m,n}_i}(s_k)$ and $t^{m,n}_{d-e,i}(s_k)$ is divisible by $T^{m,n}_{sleep}(s_k)$, then it can be concluded that it certainly wakes up at $\tau^E_{tp^{m,n}_{i+1}}(s_k)$ and fully detects $tp^{m,n}_{i+1}(s_k)$.

The proof of sufficiency condition: We show that if node s_k fully detects every target point which passes through its sensing region during round $R^{m,n}$ then $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$. To show this we use contradiction. Assume that $T^{m,n}_{sleep}(s_k)$ is not a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$ which implies that there exists $t^{m,n}_{d-e,i}(s_k) = \tau^E_{tp^{m,n}_{i+1}}(s_k) - \tau^D_{tp^{m,n}_i}(s_k)$ which is not divisible by $T^{m,n}_{sleep}(s_k)$. Now we consider the following two cases:

- $tp^{m,n}_i(s_k)$ **is not detected by sensor node** s_k:This case can not occur because it is in contradiction with the assumption that s_k fully detects every target point which passes through its sensing region during round $R^{m,n}$.
- $tp^{m,n}_i(s_k)$ **is detected by sensor node** s_k: If $tp^{m,n}_i(s_k)$ is detected by sensor node s_k, then by the second assumption of the lemma, node s_k switches to the sleep operation mode at $\tau^D_{tp^{m,n}_i}(s_k)$. Since node s_k switches to the sleep operation mode at $\tau^D_{tp^{m,n}_i}(s_k)$ and $t^{m,n}_{d-e,i}(s_k)$ is not divisible by $T^{m,n}_{sleep}(s_k)$, it can be concluded that it does not wake up at $\tau^E_{tp^{m,n}_{i+1}}(s_k)$ which implies that $tp^{m,n}_{i+1}(s_k)$ cannot be fully

detected. This is again in contradiction with the assumption that s_k fully detects every target point which passes through its sensing region during round $R^{m,n}$. □

Lemma 2. *If for a sensor network we have that*

- *all the nodes are activated at the beginning of every round,*
- *any node in the network immediately switches to sleep operation mode if it cannot sense any target point,*

then the network fully detects all the target points ($\eta_D = 1$) if and only if $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$ given by Eq. (5).

Proof of Lemma 2. By Lemma 1 and the fact that all nodes are activated at the beginning of every round (assumption 1) we can conclude that the network can fully detects all the target points, that is $\eta_D = 1$ if and only if $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$. □

Definition 12

$E(T^{m,n}_{sleep}(s_k))$ is defined as the amount of energy consumed by a node s_k during round $R^{m,n}$ using the sleeping duration of $T^{m,n}_{sleep}(s_k)$.

Lemma 3. *The energy consumed by a node during a round is a decreasing function of the sleeping duration of that node.*

Proof of Lemma 3. Let $T^{m,n}_{1,sleep}(s_k)$ and $T^{m,n}_{2,sleep}(s_k)$ be two different sleeping durations for node s_k during round $R^{m,n}$. If $T^{m,n}_{1,sleep}(s_k) < T^{m,n}_{2,sleep}(s_k)$, then we can conclude that node s_k switches between the active operation mode and the sleep operation mode more frequently when it uses $T^{m,n}_{1,sleep}(s_k)$ as the sleeping duration. This implies that $E(T^{m,n}_{1,sleep}(s_k)) > E(T^{m,n}_{2,sleep}(s_k))$ and hence the lemma. □

Lemma 4. *If the sleeping duration of any node during any round is set to its maximum possible value then the energy consumed by the network will be minimized.*

Proof of Lemma 4. The proof of this lemma is immediate from Lemma 3. □

Proof of Theorem 1. Lemma 2 states that when $\eta_D = 1, T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$. This implies that if $\eta_D = 1$, then the maximum possible value for the sleeping duration of node s_k during round $R^{m,n}$ is $\underline{T}^{m,n}_{sleep}(s_k)$. Lemma 4 states that if the sleeping duration of any node during any round is set to its maximum possible value, then the energy consumption of the network will be minimized. Therefore, if $T^{m,n}_{sleep}(s_k)$ is set to $\underline{T}^{m,n}_{sleep}(s_k)$ for any node s_k in any round $R^{m,n}$, then the energy consumption of the network will be minimized.

Conversely, assume that the energy consumed by the network is minimum. By Lemma 2, $T^{m,n}_{sleep}(s_k)$ is a divisor of $\underline{T}^{m,n}_{sleep}(s_k)$. By Lemma 3, if $T^{m,n}_{sleep}(s_k) < \underline{T}^{m,n}_{sleep}(s_k)$, then $E(T^{m,n}_{sleep}(s_k)) > E(\underline{T}^{m,n}_{sleep}(s_k))$ which is in contradiction with the assumption that the energy consumed by the network is minimum. This implies that $T^{m,n}_{sleep}(s_k)$ must be equal to $\underline{T}^{m,n}_{sleep}(s_k)$. □

References

[1] M. Ilyas, I. Mahgoub, Handbook of Sensor Networks: Compact Wireless and Wired Sensing Systems, CRC Press, London, Washington, DC, 2005.
[2] S.S. Dhillon, K. Chakrabarty, S.S. Iyengar, Sensor placement for grid coverage under imprecise detections, in: Proceedings of the 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 Parallel and Distributed Computing 64 (7) (2004) 788–798.
[4] H. Chen, H. Wu, N-F. Tzeng, Grid-based approach for working node selection in wireless sensor networks, in: Proceedings of the IEEE International Conference 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 54 (8) (2005) 978–991.
[6] F. Pedraza, A. García, A.L. Medaglia, Efficient coverage algorithms for wireless sensor networks, in: Proceedings 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, in: Second International Conference 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, in: Proceedings 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, Ph.D. Dissertation Proposal, University of Southern California, February 2004.
[11] B. Shucker, J.K. Bennett, Target tracking with distributed robotic macrosensors, in: Proceedings of the MILCOM 2005, Atlantic City, New Jersey, October, 2005.
[12] R. Olfati-Saber, Distributed tracking for mobile sensor networks with information-driven mobility, in: Proceedings 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, in: Second Internationl Workshop on Information Processing in Sensor Networks, Palo Alto, CA, USA, April 2003.
[14] R. Gupta, S.R. Das, Tracking moving targets in a smart sensor network, in: Proceedings of the IEEE VTC, vol. 5, October 2003, pp. 3035–3039.
[15] Ch. Gui, P. Mohapatra, Power conservation and quality of surveillance in target tracking sensor networks, in: Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MOBICOM 2004), Philadelphia, PA, USA, September–October 2004.
[16] G. He, J.C. Hou, Tracking targets with quality in wireless sensor networks, in: 13th IEEE International Conference 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, in: World Congress in Computer Science, 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, in: International Conference on Mobile Computing and Networking, Los Angeles, CA, USA, 2006.
[19] B. Liu, P. Brass, O. Dousse, Mobility improves coverage of sensor networks, in: Proceedings of the 6th ACM International 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 Infocom 2007 Minisymposia, August 2007.
[21] V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, Energy-aware wireless sensor networks, IEEE Signal Processing Magazine (2002–2003).
[22] K. Chakrabarty, S.S. Iyengar, H. Qi, E. Cho, Grid coverage for surveillance and target location in distributed sensor networks, IEEE Transactions on Computers 51 (2002) 1448–1453.

[23] K.K. Koustuv Dasgupta, P. Namjoshi, An efficient clustering-based heuristic for data gathering and aggregation in sensor networks, in: Proceedings of the IEEE Wireless Communications and Networking Conference, 2003.
[24] G. Bontempi, Y. Le Borgne, An adaptive modular approach to the mining of sensor network data, in: Workshop on Data Mining in Sensor Networks, SIAM SDM, Newport Beach, CA, USA, April 2005.
[25] C. Liu, K. Wu, J. Pei, A dynamic clustering and scheduling approach to energy saving in data collection from wireless sensor networks, in: Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05), Santa Clara, California, USA, September, 2005.
[26] O. Younis, S, Fahmy, An experimental study of routing and data aggregation in sensor networks, in: Proceedings of the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN), held in conjunction with the 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS-2005), November 2005.
[27] R. Virrankoski, A. Savvides, TASC: topology adaptive spatial clustering for sensor networks, in: Second IEEE International Conference on Mobile Ad Hoc and Sensor systems, Washington, DC, November, 2005.
[28] S. Soro, W. Heinzelman, Prolonging the lifetime of wireless sensor networks via unequal clustering, in: Proceedings of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (IEEE WMAN'05), April 2005.
[29] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: Proceedings of 33rd Hawaii International Conference on System Science (HICSS'00), January 2000.
[30] O. Younis, S. Fahmy, Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach, in: Proceedings of the IEEE INFOCOM, vol. 1, March 2004, pp. 629–640.
[31] S.P. Chaudhuri, D.B. Johnson, An adaptive scheduling protocol for multi-scale sensor network architecture, in: IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), Santa Fe, New Mexico, June 2007.
[32] A.M. Chou, V. Li, Slot allocation strategies for TDMA protocols in multihop packet radio networks, in: Proceedings of the INFOCOM, 1992, pp. 710–716.
[33] I. Chlamtac, A. Farago, Making transmission schedules immune to topology changes in multi-hop packet radio networks, IEEE/ACM Transactions on Networks 2 (1994) 23–29.
[34] V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, Energy-efficient collision-free medium access control for wireless sensor networks, in: Proceedings of the First International Conference on Embedded Networked Sensor Systems (Sensys'03), New York, 2003, pp. 181–192.
[35] L. Bao, J.J. Garcia-Luna-Aceves, A new approach to channel access scheduling for ad hoc networks, in: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom'01), New York, 2001, pp. 210–221.
[36] M. Sichitiu, Cross-layer scheduling for power efficiency in wireless sensor networks, in: Proceedings of the IEEE INFOCOM, vol. 1, March 2004.
[37] Y. Nam, T. Kwon, H. Lee, H. Jung, Y. Choi, Guaranteeing the network lifetime in wireless sensor networks: a MAC layer approach, Elsevier Computer Communications 30 (2007) 2532–2545.
[38] Sh. Liu, K.W. Fan, P. Sinha, Dynamic sleep scheduling using online experimentation for wireless sensor networks, in: Proceedings of the SenMetrics, San Diego, July 2005.
[39] Y. Xu, J.S. Heidemann, D. Estrin, Geography-informed energy conservation for ad hoc routing, in: Proceedings of the Mobile Computing and Networking, 2001, pp. 70–84.
[40] J. Salzmann, S. Kubisch, F. Reichenbach, D. Timmermann, Energy and coverage aware routing algorithm in self organized sensor networks, in: Proceedings of Networked Sensing Systems (INSS'07), June 2007.
[41] C. Schurgers, V. Tsiatsis, M. Srivastava, STEM: topology management for energy efficient sensor networks, in: Proceedings of the IEEE Aerospace Conference, March 2002.
[42] R. Akl, U. Sawant, Grid-based coordinated routing in wireless sensor networks, in: Proceedings of 4th IEEE Conference on Computer Communications and Networking, January 2007.
[43] G. Wang, G. Cao, T. La Porta, W. Zhang, Sensor relocation in mobile sensor networks, in: Proceedings of the IEEE INFOCOM, March 2005.
[44] J.N. Al-Karaki, R. Ul-Mustafa, A.E. Kamal, Data aggregation in wireless sensor networks – exact and approximate algorithms, in: Proceedings of the IEEE Workshop on High Performance Switching and Routing, April 2004, pp. 241–245.
[45] F. Araújo, L. Rodrigues, On the monitoring period for fault-tolerant sensor networks, in: Proceedings of Latin-American Symposium on Dependable Computing, 2005, pp. 174–190.
[46] Z. Jiang, J. Wu, A. Agah, B. Lu, Topology control for secured coverage in wireless sensor networks, in: Proceedings of the 3rd IEEE International Workshop on Wireless and Sensor Networks Security (WSNS'07), October 2007.
[47] R.P. Liu, G. Rogers, S. Zhou, J. Zic, Topology control with hexagonal tessellation, International Journal of Sensor Networks 2 (1/2) (2007) 91–98.
[48] R.P. Liu, G. Rogers, S. Zhou, Honeycomb architecture for energy conservation in wireless sensor networks, in: Proceedings of IEEE Globecom, San Francisco, November 2006.
[49] Y. Wu, S. Fahmy, N.B. Shroff, Optimal QoS-aware sleep/wake scheduling for time-synchronized sensor networks, in: Invited Sessions on Optimization of Communication Networks 40th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, 2006.
[50] M. Bagheri, M. Hefeeda, Randomized k-coverage algorithms for dense sensor networks, in: Proceedings of the IEEE INFOCOM 2007 Minisymposium, Anchorage, AK, May 2007, pp. 2376–2380.
[51] M. Bagheri, M. Hefeeda, H. Ahmadi, A Near Optimal K-Coverage Algorithm for Large-scale Sensor Networks, Technical Report TR 2006-10, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, May 2006.
[52] B. Wang, K.C. Chua, V. Srinivasan, W. Wang, Scheduling sensor activity for point information coverage in wireless sensor networks, in: Proceedings of the WiOpt, 2006.
[53] B. Wang, K.C. Chua, V. Srinivasan, W. Wang, Sensor density for complete information coverage in wireless sensor networks, in: Proceedings of the EWSN 2006, Zurich, Switzerland, February 2006.
[54] B. Liang, J. Frolik, X.S. Wang, A predictive QoS control strategy for wireless sensor networks, in: The 1st Workshop on Resource Provisioning and Management in Sensor Networks (PRMSN 05) in conjunction with the 2nd IEEE MASS, Washington DC, November 2005.
[55] J. Frolik, QoS control for random access wireless sensor networks, in: Wireless Communications and Networking Conference (WCNC04), Atlanta, March 2004.
[56] J.W. Branch, G.G. Chen, B.K. Szymanski, ESCORT: energy-efficient sensor network communal routing topology using signal quality metrics, in: Proceedings of the 4th IEEE International Conference on Networking (IEEE ICN'05), Reunion Island, France, April 2005.
[57] Y. Cai, M. Li, W. Shu, M-Y. Wu, ACOS: an area-based collaborative sleeping protocol for wireless sensor networks, International Journal of Ad Hoc & Sensor Wireless Networks (2006).
[58] Ch-f. Hsin, M. Liu, Network coverage using low duty-cycled sensors: random & coordinated sleep algorithms, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN 2004), Berkeley, California, USA, April 26–27, 2004.
[59] J. Wu, Sh. Yang, Coverage issue in sensor networks with adjustable ranges, in: Proceedings of 33rd International Conference on Parallel Processing Workshops (ICPP 2004 Workshops), Montreal, Quebec, Canada, August 2004.
[60] M. Sigalas, G. Vouros, Poster abstract: energy efficient area coverage in arbitrary sensor networks, in: Proceedings of the EWSN 2006, Zurich, Switzerland, February 2006.
[61] R. Zheng, G. He, X. Liu, Location-free Coverage Maintenance in Wireless Sensor Networks, Technical Report UH-CS-05-15, Dept. of Computer Science, University of Houston, 2005.
[62] A. Giusti, A.L. Murphy, G.P. Picco, Decentralized scattering of wake-up times in wireless sensor networks, in: Proceedings of the 4th European Conference on Wireless Sensor Networks (EWSN), Delft, The Netherlands, Lecture Notes on Computer Science, vol. 4373, Springer, January 2007, pp. 245–260.
[63] H. Zhang, J.C. Hou, Maintaining Sensing Coverage and Connectivity in Large Sensor Networks, Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless and Peer-to-Peer Network, Auerbach Publications, 2006. Chapter 28, pp. 453–475.
[64] S.K. Das, W. Choi, Coverage-adaptive random sensor selection with latency control for application-specific, in: Proceedings of the IEEE International Conference on Control and Automation (ICCA2005), Budapest, Hungary, June 2005.
[65] F. Ye, G. Zhong, J. Cheng, S. Lu, L. Zhang, PEAS: a robust energy conserving protocol for long-lived sensor networks, in: Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), 2003, pp. 28–37.

[66] N. Bisnik, N. Jaggi, Randomized scheduling algorithms for wireless sensor networks, Work in Progress, 2006.
[67] O.H. Sanli, H. Cam, X. Cheng, EQoS: an energy efficient QoS Protocol for wireless sensor networks, in: Proceedings of the 2004 Western Simulation Multi Conference, San Diego, CA, January 2004.
[68] T.S. Rappaport, Wireless Communications, Principles and Practice, Prentice Hall, New Jersey, 1996.
[69] Kh.Th. Soe, Increasing lifetime of target tracking wireless sensor networks, in: Proceedings of World, Academy of Science Engineering and Technology, August 2008, pp. 2070–3740.
[70] B. Jiang, B. Ravindran, H. Cho, Energy efficient sleep scheduling in sensor networks for multiple target tracking, Lecture Notes in Computer Science (2008) 498–509.
[71] L. Arienzo, M. Longo, An energy-efficient strategy for target tracking through wireless sensor networks, in: Gruppo Telecomunicazioni Teoria dell'Informazione Conference, Florence, Italy, June 2008.
[72] X. Wang, J.J. Ma, L. Ding, D.W. Bi, Robust forecasting for energy efficiency of wireless multimedia sensor networks, Sensors Journals 7 (2007) 2779–2807.
[73] X. Wang, J.J. Ma, Sh. Wang, D.W. Bi, Cluster-based dynamic energy management for collaborative target tracking in wireless sensor networks, Sensors Journals 7 (2007) 1193–1215.
[74] Y.M.A. Khalifa, E. Okoene, M.B. Al-Mourad, Autonomous intelligent agent-based tracking systems, ICGST-ACSE Journal 7 (1) (2007) 21–31.
[75] M.K. Watfa, An energy efficient approach to dynamic coverage in wireless sensor networks, Journal of Networks 1 (4) (2006) 10–20.
[76] T. He, P. Vicaire, T. Yan, L. Luo, L. Gu, G. Zhou, Achieving real-time target tracking using wireless sensor networks, in: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), April 2006.
[77] H. Yang, B. Sikdar, Lightweight target tracking protocol using ad hoc sensor networks, in: Proceedings of the IEEE VTC, Stockholm, Sweden, May 2005.
[78] S. Bhattacharya, G. Xing, Ch. Lu, G.C. Roman, O. Chipara, B. Harris, Dynamic wake-up and topology maintenance protocol with spatiotemporal guarantees, in: Proceedings of 4th International Symposium on Information Processing in Sensor Networks, 2005, pp. 28–34.
[79] W. Zhang, G. Cao, Optimizing tree reconfiguration for mobile target tracking in sensor networks, in: Proceedings of Twenty-third Annual Joint Conference of the INFOCOM, vol. 4, 2004, pp. 2434–2445.
[80] M. Haleem, R. Chandramouli, Adaptive downlink scheduling and rate selection: a cross layer design, Special issue on Mobile Computing and Networking, IEEE Journal on Selected Areas in Communications 23(6) (2005).
[81] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Exploiting locality of demand to improve the performance of wireless data broadcasting, IEEE Transactions on Vehicular Technology 55 (4) (2006) 1347–1361.
[82] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Learning-automata-based polling protocols for wireless LANs, IEEE Transactions on Communications 51 (3) (2003) 453–463.
[83] P. Nicopolitidis, G.I. Papadimitriou, M.S. Obaidat, A.S. Pomportsis, Carrier-sense-assisted adaptive learning MAC protocol for distributed wireless LANs, International Journal of Communication Systems 18 (7) (2005) 657–669.
[84] P. Nicopolitidis, G.I. Papadimitriou, A.S. Pomportsis, Distributed protocols for ad-hoc wireless LANs: a learning-automata-based approach, Ad Hoc Networks Journal 2 (4) (2004) 419–431.
[85] B.V. Ramana, C.S.R. Murthy, Learning-TCP: a novel learning automata based congestion window updating mechanism for ad hoc wireless networks, in: Proceedings of 12th IEEE Interantional Conference on High Performance Computing, December 2005, pp. LNCS 454–464.
[86] H. Beigy, M.R. Meybodi, A learning automata based dynamic guard channel scheme, Lecture Notes on Information and Communication Technology, vol. 2510, Springer Verlag, 2002.
[87] H. Beigy, M.R. Meybodi, An adaptive uniform guard channel algorithm: a learning automata approach, Lecture Notes in Intelligent Data Engineering and Automated Learning, Springer Verlag, LANCES 2690, Berlin, Heidelberg, Germany, 2003, pp. 405–409.
[88] H. Beigy, M.R. Meybodi, Learning automata based dynamic guard channel algorithms, Journal of High Speed Networks, in press.
[89] M. Golipour, M.R. Meybodi, LA-Mobicast: a learning automata based Mobicast routing protocol for wireless sensor networks, Sensor Letters 6 (2) (2008) 305–311.
[90] M. Esnaashari, M.R. Meybodi, A cellular learning automata based clustering algorithm for wireless sensor networks, Sensor Letters 6 (5) (2008) 723–735.
[91] <http://www.nytimes.com/2008/07/12/business/12newpark.html>.
[92] A. Sobeih, W.P. Chen, J.C. Hou, L.C. Kung, N. Li, H. Lim, H.Y. Tyan, H. Zhang, J-Sim: a simulation and emulation environment for wireless sensor networks, IEEE Wireless Communications 13 (4) (2006) 104–119.
[93] L. Wang, Y. Xiao, A survey of energy-efficient scheduling mechanisms in sensor networks, Mobile Networks and Applications 11 (5) (2006) 723–740.
[94] J.R. Polastre, Design and Implementation of Wireless Sensor Networks for Habitat Monitoring, M.Sc. Thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2003.
[95] J. Ambagis, Census and Monitoring Techniques for Leach's Storm Petrel (Oceanodroma Leucorhoa), M.Sc. Thesis, College of the Atlantic, Bar Harbor, 2002.
[96] M.F. Munir, F. Filali, Low-energy, adaptive, and distributed MAC protocol for wireless sensor–actuator networks, in: 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, September 2007, pp. 1–5.
[97] K. Stone, M. Colagrosso, Efficient duty cycling through prediction and sampling in wireless sensor networks, Wireless Communications and Mobile Computing 7 (9) (2007) 1078–1102.
[98] Zh. Liu, I. Elhanany, RL-MAC: a QoS-aware reinforcement learning based MAC protocol for wireless sensor networks, in: International Conference on Networking, Sensing and Control, 2006, pp. 768–773.
[99] T.V. Dam, K. Langendoen, An adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, New York, USA, 2003, pp. 171–180.
[100] T. Zheng, S. Radhakrishnan, V. Sarangan, PMAC: an adaptive energy-efficient MAC protocol for wireless sensor networks, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005, pp. 65–72.
[101] A. Muttreja, A. Raghunathan, S. Ravi, N.K. Jha, Active learning driven data acquisition for sensor networks, in: Proceedings of the 11th IEEE Symposium on Computers and Communications, 2006, pp. 929–934.
[102] S. Coleri, A. Puri, P. Varaiya, Power efficient system for sensor networks, in: Proceedings of the 8th IEEE Symposium on Computers and Communications, 2003, pp. 837–842.
[103] H. Mostafaei, M.R. Meybodi, M. Esnaashari, EEMLA: energy efficient monitoring of wireless sensor network with learning automata, in: International Conference on Signal Acquisition and Processing, Bangalore, India, 2010, pp. 107–111.
[104] M.G. Rabbat, R.D. Nowak, Decentralized source localization and tracking, in: Proceedings of the (ICASSP'04) Acoustics, Speech, and Signal Processing, IEEE, 2004.
[105] B.S. Malhotra, A.A. Aravind, Energy efficient on-site tracking of mobile target in wireless sensor networks, in: Proceedings of the 2004 Intelligent Sensors, Sensor Networks and Information Processing Conference, 2004.
[106] S. Balasubramanian, I. Elangovan, S.K. Jayaweera, K.R. Namuduri, Energy-aware, collaborative tracking with ad-hoc wireless sensor networks, in: IEEE Wireless Communications and Networking Conference, IEEE, 2005.
[107] I. Dietrich, F. Dressler, On the lifetime of wireless sensor networks, ACM Transactions on Sensor Networks 5 (1) (2009) 5:1–5:39.
[108] B.J. Oommen, D.C.Y. Ma, Deterministic learning automata solutions to the equi-partitioning problem, IEEE Transactions on Computers 37 (1) (1998) 2–14.
[109] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
[110] G.A. Rummery, M. Niranjan, On-Line Q-Learning Using Connectionist Systems, Technical Report, Cambridge University, England, 1994.
[111] C.J.C.H. Watkins, Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, England, 1989.
[112] A.G. Barto, R. Sutton, C. Anderson, Neuron-like elements that can solve difficult learning control problems, IEEE Transactions on Systems Man and Cybernetics SMC-13 (5) (1983) 835–846.
[113] A. Schwartz, A reinforcement learning method for maximizing undiscounted rewards, in: Proceedings of the 10th International Conference On Machine Learning, Morgan Kaufman, San Mateo, CA, 1993, pp. 298–305.

[114] E. Felemban, C.G. Lee, E. Ekici, R. Boder, S. Vural, Probabilistic QoS guarantee in reliability and timeliness domains in wireless sensor networks, in: Proceedings of the 24th INFOCOM, March, 2005, pp. 2646–2657.

[115] W.B. Jaballah, N. Tabbane, Multi path multi speed contention window adapter, International Journal of Computer Science and Network Security 9 (2) (2009) 113–118.

[116] D. Pompili, T. Melodia, I.F. Akyildiz, Routing algorithms for delay-insensitive and delay-sensitive applications in underwater sensor networks, in: Proceedings of the International Conference on Mobile Computing and Networking (MobiCOM), 2006, pp. 298–309.

[117] N. Dimokas, D. Katsaros, Y. Manolopoulos, Cooperative caching in wireless multimedia sensor network, Mobile Networks and Applications Journal 13 (3–4) (2008) 337–356.

[118] J. Kim, P. Bentley, Ch. Wallenta, Danger is ubiquitous: detecting malicious activities in sensor networks using the dendritic cell algorithm, in: Proceedings of the 5th International Conference on Artificial Immune Systems, September, 2006, pp. 390–403.

[119] F. Agyei-Ntim, K. Newman, Lifetime estimation of wireless body area sensor network for patient health monitoring, in: Proceedings of the 31st Annual IEEE International EMBS Conference, September, 2009, pp. 1659–1662.

[120] R. Tyan, G.M.P. O'Hare, M.J. O'Grady, C. Muldoon, EDLA Tradeoffs for wireless sensor network target tracking, in: Proceedings of the 29th International Conference on Distributed Computing Systems, 2009.

[121] A. Papadimitriou, D. Katsaros, Y. Manolopoulos, Query sensitive storage for wireless sensor networks, in: Proceedings of 13th Panhellenic Conference on Informatics, 2009, pp. 25–29.

[122] J. Ch. Chin, I.H. Hou, J.C. Hou, C. Ma, N.S. Rao, M. Saxena, M. Shankar, Y. Yang, D.K.Y. Yau, A sensor-cyber network testbed for plume detection, identification, and tracking, in: Proceedings of 6th International Symposium on Information Processing in Sensor Networks, 2007, pp. 541–542.

Mehdi Esnaashari received the B.S. and M.S. degrees in Computer Engineering both from the Amirkabir University of Technology in Iran, in 2002 and 2005, respectively. Currently, he is a Ph.D. student in Computer Engineering Department at the Amirkabir University of Technology, Tehran, Iran. His research interests include computer networks, learning systems and soft computing.

M.R. Meybodi received the B.S. and M.S. degrees in Economics from the Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degrees from the Oklahoma University, USA, in 1980 and 1983, respectively, in Computer Science. Currently he is a Full Professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an Assistant Professor at the Western Michigan University, and from 1985 to 1991 as an Associate Professor at the Ohio University, USA. His research interests include, channel management in cellular networks, learning systems, parallel algorithms, soft computing and software development.