Data aggregation in sensor networks using learning automata

Mehdi Esnaashari · M. R. Meybodi

Abstract One way to reduce energy consumption in wireless sensor networks is to reduce the number of packets being transmitted in the network. As sensor networks are usually deployed with a number of redundant nodes (to overcome the problem of node failures which is common in such networks), many nodes may have almost the same information which can be aggregated in intermediate nodes, and hence reduce the number of transmitted packets. Aggregation ratio is maximized if data packets of all nodes having almost the same information are aggregated together. For this to occur, each node should forward its packets along a path on which maximum number of nodes with almost the same information as the information of the sending node exist. In many real scenarios, such a path has not been remained the same for the overall network lifetime and is changed from time to time. These changes may result from changes occurred in the environment in which the sensor network resides and usually cannot be predicted beforehand. In this paper, a learning automata-based data aggregation method in sensor networks when the environment's changes cannot be predicted beforehand will be proposed. In the proposed method, each node in the network is equipped with a learning automaton. These learning automata in the network collectively learn the path of aggregation with maximum aggregation ratio for each node for transmitting its packets toward the sink. To evaluate the performance of the proposed method computer simulations have been conducted and the results are compared with the results of three existing methods. The results have shown that the proposed method outperforms all these methods, especially when the environment is highly dynamic.

M. Esnaashari · M. R. Meybodi (✉)
Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology, Tehran, Iran
e-mail: mmeybodi@aut.ac.ir

M. Esnaashari
e-mail: esnaashari@aut.ac.ir

M. R. Meybodi
School of Computer Science, Institutes for Studies in Theoretical Physics and Mathematics (IPM), Tehran, Iran

Keywords Sensor networks · Data aggregation · Learning automata

1 Introduction

Sensor nodes are very prone to failure due to their limited resources and abilities. One major reason for this failure is exhausting energy resources and for this reason energy is a vital factor in sensor networks. Multi-hop routing which is a way for gathering information of every node in every part of the network at the sink is one of the major energy consuming tasks performed throughout the lifetime of a sensor network. For this reason, each node should try to find a path to the sink using which minimum energy is consumed by the network. One way to do so is for every node to compute the amount of energy consumed on each path separately, and then to choose the path with the least energy consumption [1]. One problem with such a method is that, it needs a way of computing energy consumption along each path. Another way is to make use of data aggregation technique. In data aggregation technique, packets with related information are combined together in intermediate nodes to form a single packet before further forwarding to the sink. This reduces the number of transmitted packets in the network and as a result, less energy is consumed. Using this method, a node has to select a path along which maximum number of nodes exists with related packets. We refer to this path for each node as the *Aggregation Path* of that node hereafter.

Finding *Aggregation Path* for each node is not so simple considering only local information. A simple approach to find an approximation of this path is to use a greedy algorithm. Each node tries to forward its packets to a neighbor which has related information to that of the node as it is done in [2]. The problem of finding *Aggregation Path* for each node becomes even more complicated if the information possessed by each node differs during the lifetime of the network. In such a situation, *Aggregation Path* of each node changes from time to time and hence, nodes should adapt themselves to these changes.

In this paper a new method based on learning automata has been proposed to make nodes capable of adapting themselves to changes occurred in their *Aggregation Paths*. In the proposed method, each node in the network is equipped with one learning automaton. These learning automata collectively learn the *Aggregation Path*s of all nodes of the network and adapt nodes to the changes occurred in the paths.

To evaluate the performance of the proposed method several experiments have been conducted and the results are compared with the results of three different methods: (i) LEACH algorithm [3], (ii) method given in [2] which performs data aggregation with no learning, and (iii) method given in [4] which performs data aggregation using Q-learning. The results have shown that the proposed method outperforms all these methods, especially when *Aggregation Paths* are highly dynamic.

The rest of this paper is organized as follows. Section 2 gives an overview on related works reported on data aggregation in sensor networks. Learning automata as a basic learning strategy used in the proposed method will be discussed in Sect. 3. The problem statement is given in Sect. 4 and in Sect. 5 the proposed method is presented. Simulation results are given in Sect. 6. Section 7 is the conclusion.

2 Related work

Some of the methods reported recently for data aggregation in sensor networks such as [5–7], are query based methods. In these methods a query is generated at the sink and then is broadcasted through the network. Each node, upon receiving this query, processes it partially, and then passes it on to its neighbors. After the query is processed completely, its result will be sent back to the sink. In this scenario, some nodes just process the query, and some others, propagate it, receive partial results, aggregate the results, and send them back to the sink.

In some other works, such as [3, 8–12], the network is first clustered, and then cluster heads are used for aggregating data packets in each cluster separately. Among these methods, LEACH[1] protocol which was introduced by Heinzelman et al. attracted more attention. The operation of LEACH is separated into two phases: the setup phase and the steady state phase. During the setup phase, a predetermined fraction of nodes p, elect themselves as cluster heads by comparing a chosen random number with a predefined threshold. In the steady state phase, cluster heads, aggregate reported data from their cluster member nodes and forward aggregated data to the sink. Lotfinezhad and Liang in [13] try to investigate the effect of partially correlated data on the performance of the clustering methods for data aggregation. They have shown that partially correlated data has a strong effect on the clustering performance, meaning that the lower is the data correlation; the lower is the clustering performance. They have also shown that the amount of energy consumed in a single node is strongly related to its position in the network; a node far from the sink may consume more energy than other nodes. They have also shown that the network lifetime is inversely proportional to total consumed energy.

Some other methods are also reported for data aggregation in sensor networks. Guestrin et al. in [14] try to approximate the data generated in a single node for a specific period of time with a third order equation, and send the coefficients of this equation to the sink instead of data itself. Dasgupta et al. in [15] assume that each node in the network has the ability to aggregate data. Based on this assumption, they present a method for maximizing network lifetime. They define the network lifetime as the time elapsed before the first node dies due to energy exhaustion. This method tries to find an aggregation tree rooted at the sink spanning all the sensors and maximizes the lifetime of the network. For this purpose, they first find all possible aggregation trees and then, make use of a heuristic approach to find the one which can maximize the lifetime of the network.

Beaver and Sharaf in [2] propose an algorithm which tries to find paths along which, sensors belonging to the same group (i.e., sensing the same phenomenon) reside. In this algorithm, sink will initiate a "path construction" packet through the network. Each node, upon receiving this packet for the first time, set its parent to the sender of the packet, and then forwards the packet to its neighbors. If a node receives "path construction" again, it checks whether sender of the packet belongs to the same group as itself or not. If so, this node, changes its parent to this new sender of "path construction". This way, each path to the sink will consist of mostly the nodes belonging to the same group and

[1] Low Energy Adaptive Clustering Hierarchy.

hence aggregation ratio along the path will be increased. This method performs well in situations where nodes do not change their groups during the lifetime of the network.

Very few efforts in data aggregation make use of learning. Radivojak et al. in [16] uses machine learning technique to force sensor nodes to send only specific data required by the sink. Learning algorithm used in this method is executed in the sink and its results are propagated throughout the network. Beyens et al. in [4] use a Q-learner in each node forcing the node to send its data via a path along which aggregation ratio is maximum. They regard the sensor network as a Multi-Agent System (MAS) in which the agents are the sensor nodes. Each agent has a number of actions equal to the number of its neighbor nodes. Each action is to select one of the neighbors for forwarding the packets of the node towards the sink. A Q-value is associated with each action. When data is routed from the node A to the sink via neighbor node B, a feedback is given by B to A. The feedback is computed based on the number of hops from B to the sink and the aggregation ratio gained in B. This feedback is used in A to update the Q-value of the action corresponding to the selection of B. After the learning, the neighbor corresponding to the action with the highest Q-value is chosen for forwarding the node's data towards the sink.

Learning automata is proved to perform well in the dynamic environments of wireless, ad hoc and sensor networks. Haleem and Chandramouli in [17] use learning automata to address a cross-layer design for joint user scheduling and adaptive rate control for downlink wireless transmission. The proposed method tends to ensure that user defined rate requests are satisfied by the right combination of transmission schedules and rate selections. Nicopolitidis et al. in [18] propose a bit rate control mechanism based on learning automata for broadcasting data items in wireless networks. A learning automaton is used in the server which learns the demand of wireless clients for each data item. As a result of this learning, the server is able to transmit more demanded data items by the network more frequently. Same authors in [19] propose a learning automata-based polling protocol for wireless LANs in which the access point uses a learning automaton to assign to each station a portion of the bandwidth proportional to the station's need. A decentralized approach of the above method is also given in [20, 21]. Ravana and Morthy in [22] propose Learning-TCP, a novel learning automata-based reliable transport protocol for wireless networks, which efficiently adjusts the congestion window size and thus reduces the packet losses. Learning automata is also used in cellular radio networks to dynamically adjusting the number of guard channels [23–25]. Recently, a few attempts are made for applying learning automata to sensor networks [26–28]. In [26], Gholipour and Meybodi propose a learning automata-based mobicast routing protocol for wireless sensor networks. The proposed protocol uses learning automata to adaptively determine the location and the shape of the forwarding zone in such a way that the number of activated nodes in each forwarding hop along the routing path is almost the same. A clustering algorithm using cellular learning automata is proposed by Esnaashari and Meybodi in [27]. In this clustering algorithm, each node is equipped by a learning automaton. Learning automaton of each node in cooperation with the learning automata of the neighboring nodes determines the role of the node to be a cluster head or a cluster member. Same authors in [28] propose a learning automata-based scheduling solution to the dynamic point coverage problem. In the proposed scheduling algorithm, the learning automaton of each node learns the sleep duration of that node based on the movement pattern of a single moving target point in the network.

3 Learning automata

Learning automata is an abstract model which randomly selects one action out of its finite set of actions and performs it on a random environment. Environment then evaluates the selected action and responses to the automata with a reinforcement signal. Based on selected action and received signal, the automata updates its internal state and selects its next action. Figure 1 depicts the relationship between an automata 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

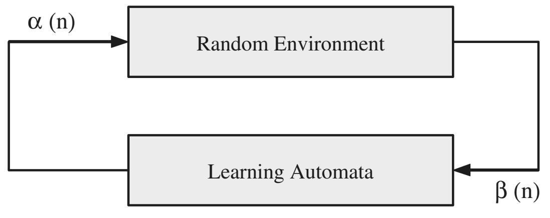

Fig. 1 Relationship between learning automata and its environment

variable-structure stochastic. In the following, we consider only variable-structure automata.

A variable-structure automaton is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2,\ldots, \alpha_r\}$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.

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

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

In these two equations, a and b are reward and penalty parameters, respectively. For $a = b$, learning algorithm is called L_{R-P}[2] for $b \ll a$, it is called $L_{R\varepsilon P}$[3] and for $b = 0$, it is called L_{R-I}.[4] For more information the reader may refer to [29, 30]. The only application of learning automata to sensor networks has been reported in [26].

4 Problem statement

Consider N sensor nodes $s_1, s_2,\ldots, s_N$ which are scattered randomly throughout a large $L \times L$ rectangular area (A) to monitor a certain phenomenon ζ such as temperature, humidity, etc. The location of each node s_i is presented by (x_i, y_i). Every node s_i is activated periodically, measures the ζ at the moment and reports the measured value of ζ ($\widehat{\zeta}_i$) to the sink node using a multi-hop routing scheme. We say $\widehat{\zeta}_i$ and $\widehat{\zeta}_j$are *almost equal* and their values can be aggregated using MEAN operator if $\left|\zeta_i - \zeta_j\right| < \varepsilon$ for a small value $\varepsilon > 0$. Considering a dense network, measured value of ζ is almost equal in some certain region ($\Omega \subset A$) around each node i, and hence $\widehat{\zeta}_i$ can be aggregated with $\widehat{\zeta}_j$, $\forall s_j \in \Omega$. ζ partitions the A into M regions $\Omega_1, \Omega_2,\ldots, \Omega_M$ with the following properties:

- $\left|\widehat{\zeta}_i - \widehat{\zeta}_j\right| < \varepsilon; \quad \forall s_i, s_j \in \Omega_k, \quad k = 1, 2, \ldots, M$
- $\Omega_i \neq \emptyset; \quad for\, i = 1, 2, \ldots, M$
- $\Omega_i \cap \Omega_j = \emptyset; \quad for\ i \neq j$
- $\bigcup_{i=1}^{M} \Omega_i = A$

[2] Linear Reward-Penalty.

[3] Linear Reward epsilon Penalty.

[4] Linear Reward Inaction.

We assume that each Ω_k can be approximated with a simple circle with center (X_k, Y_k) and radius R_k. Also we assume that ζ changes from time to time in different parts of the network and hence, $\Omega_1, \Omega_2,\ldots, \Omega_M$ are not static regions, but their positions and radiuses vary during the network lifetime.

The aim of the network is to periodically collect in the sink node the estimate of the position, the radius and the estimated $\widehat{\zeta}$ ($\widehat{Z_k}$) about each region Ω_k. For this to be achieved each node i reports its position and estimated $\widehat{\zeta}_i$ to the sink node periodically.

We define a data gathering round to be the time during which all nodes of the network be activated once and report their information to the sink. Each data gathering round is identified by a unique number called data gathering round number which starts from one and increases by one at the end of each data gathering round. Using the process of data aggregation one can reduce the number of packets received at the sink in each data gathering round approximately to its ideal number which is the number of regions in the environment at that round. Without data aggregation, N data packets so many of which are redundant are received by the sink node one form each node. Performing data aggregation provides longer lifetime for the network. Network lifetime is defined to be the time elapsed from the network startup to the time at which a node in the network dies [15]. To measure network lifetime we use *Round of Death* metric defined as the data gathering round number at which the first death in the network occurs.

To specify the problem more clearly, without loss of generality, we make use of a specific application in which nodes have the ability to sense the temperature, hence ζ is temperature in this application. We assume that several circular climates exist in the environment each having a different temperature, and moving along a specific direction with a specific velocity. The velocity and direction of the movement of each climate may be changed randomly in time. These climates and their temperatures partition the area of the network into different parts (Ω) each having a different temperature. Figure 2 depicts such an environment with 18 sensor nodes, and five climates. Dashed circles in this figure give different partitions of the network based on the estimate of temperature in different nodes. Note that these dashed circles have some area in common, since each Ω is estimated by a circle. However, it is not a complicated task for the sink node to make a more accurate estimation of Ω's in which these common areas are removed.

Each sensor node reports the temperature of the climate in which it resides. If it happens for a node to reside in more than one climate region, its sensor will sense the temperature equal to the mean temperature of all climates it resides in. A large static climate is assumed to be in the

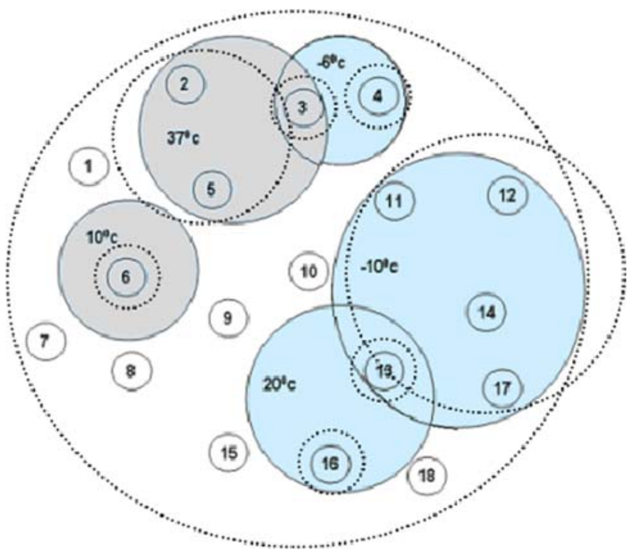

Fig. 2 A schematic of the simulated environment with 18 sensor nodes, and five different climates. Dashed circles give different partitions of the network based on the estimate of the temperature

environment which covers the whole area A of the network and its temperature is assumed to be 0. With the above assumptions, one can simply compute the temperature each node reports in Fig. 2. For example, s_2 and s_5 report 37, s_4 reports -6, and s_3 reports mean of 37 and -6 which is 15.5. In addition s_1, s_7, s_8, s_9, s_{10}, s_{15}, and s_{18} report 0 as their estimations.

5 The proposed method

In the network every node s_i has an internal clock which activates the node every t seconds to measure ζ and report it to the sink using a multi-hop routing scheme. The proposed routing scheme consists of two phases: route discovery phase and route selection phase. The route discovery phase of the routing scheme is first used to find for each node a number of alternative routes towards the sink. The route selection phase is then used to select a route among all alternative routes using which more aggregation can be performed.

Data aggregation is performed along each route to the sink wherever it is possible. Data and location aggregation will be done at the same time; that is wherever data aggregation is performed, location aggregation is also performed. In location aggregation, the positions of the nodes whose data are aggregated are averaged to find an estimate of the center of the region to which they belong. The estimated radius of the region then becomes the distance between the estimated center and the farthest node. Without location aggregation, a single aggregated packet resulted from aggregating n different packets from n different nodes, must contain the positions of all n nodes. This makes the aggregated packet too long which consumes higher energy for transmission.

In order to have more aggregation in the network each node must try to route its packets toward the sink using a path along which more aggregation can be performed. Since the positions of the regions changes dynamically, the routing algorithm must adapt itself to these changes. To achieve the above two goals each node in the network is equipped with a learning automaton which is responsible for selecting the next best hop to forward the packets towards the sink during a data gathering round. In the rest of this section, we first describe the first version of the proposed algorithm which we call it the basic algorithm in order to give the main idea. Then two improvements to the basic algorithm are given to further improve the aggregation ratio and prolong the network lifetime.

5.1 Route discovery phase

The route discovery algorithm is used to discover all possible paths for each node of the network towards the sink. The algorithm is based on the simple flooding method. Sink node initiates a "Path Construction" packet and broadcasts it throughout the network. This packet contains a parameter called "DistanceToSink" which is set to one at the sink node. Other nodes wait to receive the "Path Construction". Upon receiving this packet, three different cases may be arisen:

a) A "Path Construction" packet with lower "DistanceToSink" than that of current packet has been previously received at this node. In this case, the newly received packet is ignored.
b) The previously received "Path Construction" packets have equal "DistanceToSink" with that of current packet. In this case, sender of the "Path Construction" is added to the routing table as a new alternative path to the sink, "DistanceToSink" is increased by one, and the packet is broadcasted over the network again.
c) The previously received "Path Construction" packets have greater "DistanceToSink" than that of current packet. This means that a path with fewer numbers of hops to the sink is found. So, all the previously found paths are removed from the routing table. Then sender of the currently received "Path Construction" is added as a path to the sink, "DistanceToSink" is increased by one, and finally the packet is broadcasted over the network again.

The algorithm continues until each node in the network discovers all its paths with the same number of hops toward the sink. For each path, only storing the next hop in the routing table of the node will be sufficient.

At the end of the route discovery phase, each node in the network has a list of all its neighbors using which it can forward its packets toward the sink. We use *RoutingList$_i$* (*RL$_i$*) to refer to this list for node s_i where $|RL_i|$ is the number of entries in this list.

5.2 Route selection phase

After the route discovery phase is over each node in the network is activated every t seconds and reports its information to the sink node.

In this phase each node uses a learning automaton for selecting the next best hop to forward its packets towards the sink during each data gathering round. This way, each node gradually learns the best neighbor (the neighbor with the most related data) for forwarding its packets toward the sink.

The learning automaton associated to node s_i referred to by LA_i has $|RL_i|$ actions whose probability of selecting each is initially set to $\frac{1}{|RL_i|}$. Each action of the LA_i corresponds to the selection of one of the neighboring nodes listed in the RL_i to be used for forwarding packets of s_i towards the sink. After *node* s_i is activated and has measured the temperature, it asks its learning automaton to select one of its neighboring nodes (actions) for forwarding the measured temperature toward the sink. Data are aggregated during the route selection phase. The action selected by the automaton will be rewarded or penalized based on the acknowledgment received from the selected neighboring node. How to penalize or reward the selected action will be described later.

Between two subsequent activations of node s_i it may receive two different packet types from its neighbors; data packet and acknowledgment packet. A data packet contains measured temperature by the sender of the packet and its location whereas acknowledgment packet is the response of a neighbor to a data packet sent by s_i to that neighbor. A data packet transmitted by node s_i contains three different parameters:

– K_i which specifies the number of packets aggregated into that packet
– $\widehat{\zeta_{i,\mathrm{K}_i}}$ which is the aggregated data computed using Eq. 3:

$$\widehat{\zeta_{i,\mathrm{K}_i}} = \frac{\widehat{\zeta_i} + \sum_{j=1}^{n} \widehat{\zeta_j}}{n+1} \tag{3}$$

In the above equation, n is the number of packets received at node s_i.

– $\left(x_{i,\mathrm{K}_i}, y_{i,\mathrm{K}_i}\right)$ which is the aggregated location and computed using Eq. 4:

$$x_{i,\mathrm{K}_i} = \frac{x_i + \sum_{j=1}^{n} x_j}{n+1} \qquad y_{i,\mathrm{K}_i} = \frac{y_i + \sum_{j=1}^{n} y_j}{n+1} \tag{4}$$

Received data packets are temporary stored and will be processed upon the next activation of s_i.

In the following we discuss what a node performs when activated and what a node performs during the time between two subsequent activations.

Node s_i upon its nth activation performs the following operations:

– s_i sets $\mathrm{K}_i = 1$.
– s_i senses its surrounding environment to measure $\widehat{\zeta_i}$.
– s_i sets $\widehat{\zeta_{i,\mathrm{K}_i}} = \widehat{\zeta_i}, \quad x_{i,\mathrm{K}_i} = x_i, \quad y_{i,\mathrm{K}_i} = y_i$.
– If no data packet is received during the $(n-1)$th activation and nth activation of s_i then
  s_i creates a data packet containing $\widehat{\zeta_{i,\mathrm{K}_i}}, (x_{i,\mathrm{K}_i}, y_{i,\mathrm{K}_i})$ and K_i.
– else if any data packet is received then
  For each received packet from any neighbor j do
    – If the inequality (5) is satisfied then
      • s_i performs data aggregation using a recursive version of the Eq. 3 given in Eq. 6.
      • s_i performs location aggregation using a recursive version of Eq. 4 given in Eq. 7.
      • s_i sets $\mathrm{K}_i = \mathrm{K}_i + \mathrm{K}_j$.
      • s_i uses Eq. 8 to compute the data aggregation ratio ($DAR_{i,j}$).
      • $DAR_{i,j}$ is sent back to node s_j via an acknowledgment packet as the feedback from the environment.
  s_i creates a data packet containing $\widehat{\zeta_{i,\mathrm{K}_i}}$, $\left(x_{i,\mathrm{K}_i}, y_{i,\mathrm{K}_i}\right)$ and K_i.
– LA_i selects one of its actions (say action k) to determine the neighbor to which newly made data packet should be forwarded.
– Newly created data packet and the data packets received from the neighbors of s_i, for which inequality (3) is not satisfied, are transmitted to neighbor k.

$$\left|\widehat{\zeta_{i,\mathrm{K}_i}} - \widehat{\zeta_{j,\mathrm{K}_j}}\right| < \varepsilon \tag{5}$$

ε is a threshold which specifies the maximum difference between measured temperatures below which aggregation can be performed.

$$\widehat{\zeta_{i,\mathrm{K}_i}} = \frac{\mathrm{K}_i \cdot \widehat{\zeta_{i,\mathrm{K}_i}} + \mathrm{K}_j \cdot \widehat{\zeta_{j,\mathrm{K}_j}}}{\mathrm{K}_i + \mathrm{K}_j} \tag{6}$$

$$x_{i,\mathrm{K}_i} = \frac{\mathrm{K}_i \cdot x_{i,\mathrm{K}_i} + \mathrm{K}_j \cdot x_{j,\mathrm{K}_j}}{\mathrm{K}_i + \mathrm{K}_j}$$
$$y_{i,\mathrm{K}_i} = \frac{\mathrm{K}_i \cdot y_{i,\mathrm{K}_i} + \mathrm{K}_j \cdot y_{j,\mathrm{K}_j}}{\mathrm{K}_i + \mathrm{K}_j} \tag{7}$$

$$DAR_{i,j} = U\left(\frac{Min(\widehat{\zeta_{i,\mathrm{K}_i}} + \varepsilon,\ \widehat{\zeta_{j,\mathrm{K}_j}} + \varepsilon) - Max(\widehat{\zeta_{i,\mathrm{K}_i}} - \varepsilon,\ \widehat{\zeta_{j,\mathrm{K}_j}} - \varepsilon)}{2\varepsilon}\right) \tag{8}$$

where

$$U(x) = \begin{cases} x; & x \geq 0 \\ 0; & x < 0 \end{cases} \tag{9}$$

Upon receiving the acknowledgment packet containing $DAR_{k,i}$ by $\underline{s_i}$, LA_i penalizes or rewards its selected action (action k). If $DAR_{k,i}$ is greater than threshold *AcceptRate*, then action k is rewarded according to Eq. 10

$$\begin{aligned} p_k(n+1) &= p_k(n) + \alpha.(DAR_{k,i}).(1 - p_k(n)) \\ p_l(n+1) &= p_l(n) - \alpha.(DAR_{k,i}).p_l(n) \qquad \forall l l \neq k \end{aligned} \tag{10}$$

and otherwise, it is penalized according to Eq. 11

$$\begin{aligned} p_k(n+1) &= (1 - \beta.(1 - DAR_{k,i})).p_k(n) \\ p_l(n+1) &= \frac{\beta.(1-DAR_{k,i})}{r-1} + (1 - \beta.(1 - DAR_{k,i}))p_l(n) \\ &\qquad \forall l l \neq k \end{aligned} \tag{11}$$

where in Eqs. 10 and 11, α is the reward rate, and β is the punishment rate.

If by any means, node s_i does not receive any acknowledgment from node s_k, LA_i penalizes its selected action according to Eq. 11 considering $DAR_{k,i}$ as zero. Node s_i then retransmits its measured temperature via one of its neighboring nodes listed in RL_i other than node s_k. If no other neighboring node exists in RL_i, node s_i transmits a *Route-Request* packet to all of its neighboring nodes. If a neighboring node l receives the packet and s_i is not listed in its RL_l, it replies with a *RouteReply* message. Node s_i gathers the received *RouteReply* packets and adds them to its RL_i. The LA_i of s_i is reinitialized with all actions having the same action probabilities. If node s_i does not receive any *Route-Reply* packet, then it can be regarded as a dead node which at this time, the network lifetime is considered to be over.

5.3 Improvements

In this section we discuss two improvements to the method proposed in the previous section. For the sake of clarity in presentation in the rest of this paper we call the proposed method as "Basic Algorithm", and the basic algorithm with the first improvement as "Algorithm 1" and with the second improvement as "Algorithm 2".

5.3.1 *Algorithm 1*

In Algorithm 1 unlike the basic algorithm in which only the aggregation ratio between the node and the next node in the routing path is considered for choosing a neighboring node, the aggregation ratio between the node and its two next nodes in the routing path is considered for choosing a neighboring node. For the network of Fig. 3, if the basic algorithm is used then node s_{10} chooses one of its neighbors s_7, s_8 and s_9 at random for sending its packets towards the sink, whereas Algorithm 1 chooses node s_7. This is because the second node from s_{10} in the routing path passed through s_7 is node s_6 which is in the same region as s_{10} and hence its information can be aggregated with information of s_{10}.

Algorithm 1 performs the following. Each node s_i upon receiving a data packet from its neighbor s_j in its nth activation time, computes $DAR_{i,j}$ using Eqs. 8 and 9, and then makes use of Eq. 12 for computing *Enhanced-DAR*$_{i,j}$. Once *Enhanced-DAR*$_{i,j}$ is computed, this will be sent back to node s_j as the response of node s_i.

$$\text{Enhanced} - DAR_{i,j} = \text{Max}(DAR_{i,j}, MRR_i(n)) \tag{12}$$

In Eq. 10, MRR_i stands for Maximum Received Reward and is defined by Eq. 13.

$$MRR_i(n) = \max_{1 \leq t \leq n}\left(DAR_{k,i}; \quad for\ k \in RL_i\right) \tag{13}$$

The max operation is performed over all responses received by node s_i from the network startup until the time it is computed.

Using this algorithm, in the network of Fig. 3, the computed *Enhanced-DAR*$_{i,j}$ at node s_7 is more than that of nodes s_8 and s_9. This is because $DAR_{i,j}$ in all three nodes are equal, but *MRR* at node s_7 is more than that in nodes s_8 and s_9 which results in higher probability of selecting node s_7 by node s_{10}.

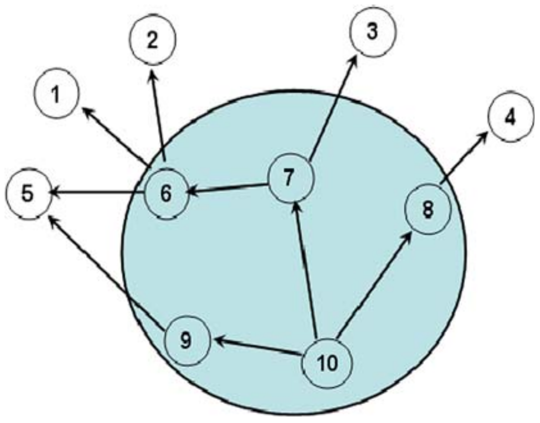

Fig. 3 Network of node s_{10} and the environment around it

5.3.2 *Algorithm 2*

Algorithm 2 is Algorithm 1 in which when a node cannot find a neighboring node for data aggregation (a neighboring node in the same region) such as node s_6 in Fig. 3, it chooses a neighboring node with more residual energy for forwarding its packets. Using this algorithm, a node located at the boundary of a region, forward it packet to the outside of its region using a neighbor with highest residual energy. This means that routing inside a region is performed based on data aggregation ratio between nodes whereas routing between different regions is performed based on residual energy.

In Algorithm 2, upon receiving a packet in node s_i from the neighboring node s_j, $\textit{Enhanced-DAR}_{i,j}$, is computed. If this ratio is more than *AcceptRate*, node s_i sends back a favorable response to LA_j. Otherwise, node s_i considers its residual energy to compute the response to action i of LA_j; if the normalized residual energy (NRE_i is computed using Eq. 14) is more than *AcceptRate*, the response is favorable and is unfavorable otherwise.

$$NRE_i = \frac{EL_i}{MaxEnergyLevel} \tag{14}$$

In the above equation, EL_i is the residual energy level of node s_i and *MaxEnergyLevel* is the residual energy level of a full battery charged node.

6 Experimental results

To evaluate the performance of the proposed method several experiments have been conducted. In the first two experiments, the basic algorithm is compared with three different methods: (i) LEACH algorithm proposed in [3], (ii) method given in [2] with no learning, and (iii) method given in [4] which uses Q-learning. In the third experiment, the performance of the proposed data aggregation algorithm is studied by comparing the number of packets received at the sink node in each data gathering round by the number of packets which needed to be received at the sink at that round. The fourth experiment compares Algorithms 1 and 2 with the Basic Algorithm. Last experiment studies the convergence behavior of learning automata in different nodes of the network.

All simulations have been implemented using NS2 simulator. We have used IEEE 802.11 as the MAC layer protocol, two ray ground as the propagation model and omnidirectional antenna. Nodes are placed randomly on a two-dimensional area of size 100 m × 100 m. Energy model specified in [15] is used for estimating amount of energy consumed for packet transmissions. All packets except for data packets, which are 525 bytes long, are assumed to be 8 bytes long. Environment is modeled by 12 circular climates; each has its own temperature, direction and velocity of movement. Climates are moving using random way point movement strategy with a velocity which is randomly selected from a normal distribution with $\mu = 0.001$ and $\sigma = 0.0001$. Movements of the climates are bounded to the boundaries of the network, and hence if a climate happens to get out of the environment boundaries, it changes its direction by 180° and comes back to the environment of the network. In all experiments, α and β for the proposed algorithm are set to 0.1, and ε (acceptable range of aggregation) is set to 5. Also α and β parameters in Q-learning method are set to 0.1 and 0.01, respectively. These values are selected experimentally. The t which is the time interval between two consecutive sending of data to the sink in each node is equal to 10 s. *AcceptRate* is set to 0.85. Simulations are performed for 50, 100, 200, 300, 400, and 500 nodes. The results are averaged over 50 runs.

6.1 Experiment 1

In this experiment, we study the total number of packets received at the sink node. Figure 4 depicts the results for the proposed method and three existing methods mentioned earlier. This figure shows that the proposed method outperforms methods (i), (ii), and (iii) by 66, 37, and 9%, respectively. Figure 5 compares the results obtained for the proposed method with the results obtained for other three methods for $N = 500$ when μ varies in the range [0, 0.05]. This figure shows that the proposed method has no superiority over other methods when climates have no movement ($\mu = 0$), but it works better as μ increases.

6.2 Experiment 2

In this experiment, we study the mean energy consumption of nodes in the network. Figure 6 shows that the proposed

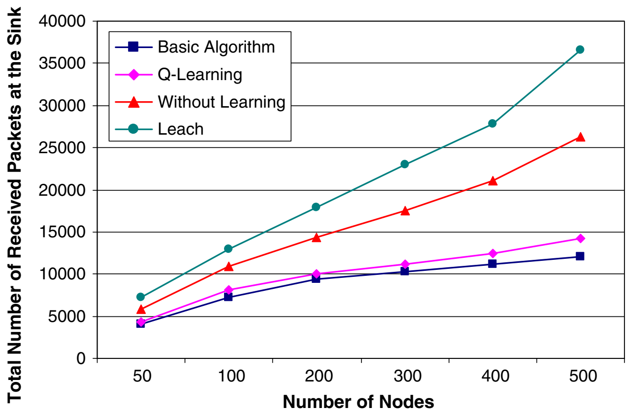

Fig. 4 Total number of received packets at the sink

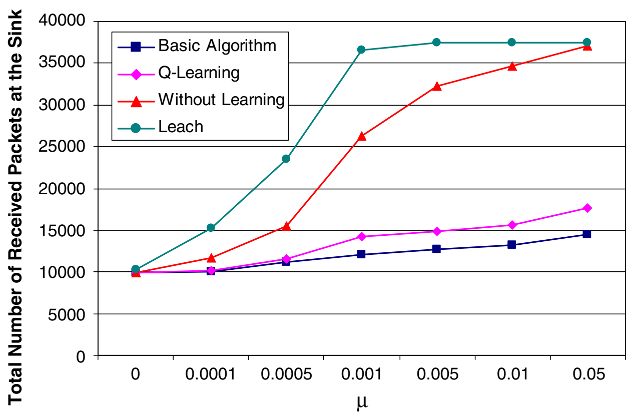

Fig. 5 Changes in total number of received packets at the sink node with respect to changes in μ

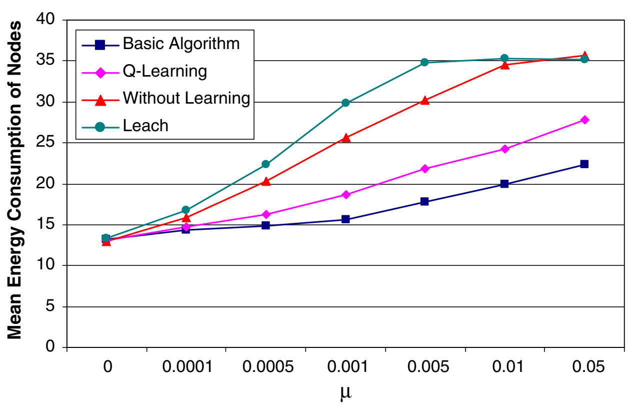

Fig. 7 Changes in mean energy consumption of nodes with respect to changes in μ

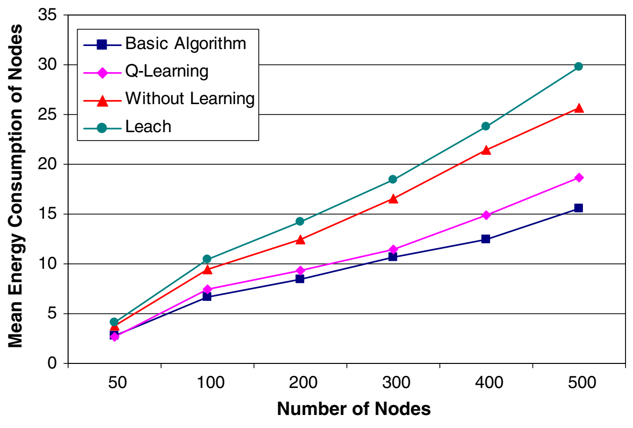

Fig. 6 Mean energy consumption of node

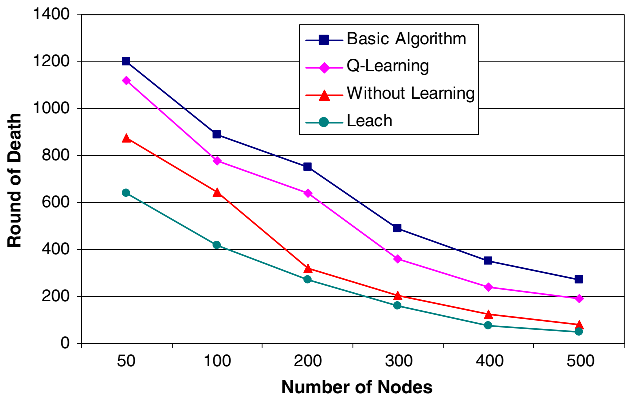

Fig. 8 Network lifetime based on the *Round of Death* metric

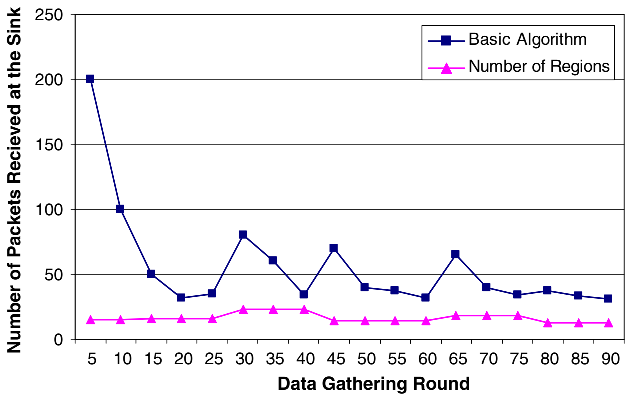

Fig. 9 Number of packets received at the sink in each data gathering round for Basic Algorithm versus the number of packets which must be received at the sink at that round

method outperforms all other three methods in terms of mean energy consumption. It should be noted that in the proposed method, the overhead of the acknowledgement packets has also been taken into consideration for estimating the total energy consumption. We found out that on average, the amount of energy consumed for sending or receiving acknowledgement packets by each node is about 30% of the total energy consumed by that node. However, reduction of the total number of data packets transmitted throughout the network in the proposed method can compensate the overhead of these acknowledgment packets. The proposed method works well when climate zones are moving. To show this, we compared the results obtained for the proposed method with the results obtained for other methods for $N = 500$ for different values of μ changing between 0 and 0.05. The comparison whose result is given in Fig. 7 shows that the proposed method performs better in terms of mean energy consumption as μ increases.

Figure 8 compares the lifetime of the network for the proposed method and other three methods in terms of the *Round of Death* metric. Higher lifetime for the proposed algorithm as shown in Fig. 8 is due to reduction in the number of transmitted packets in the network. As it can be seen from Fig. 8, when the number of nodes in the network increases, *Round of Death* decreases. This is mainly due to the fact that in a denser network, sensor nodes near the sink relay more packets and hence their energy is depleted earlier resulting in a lower *Round of Death.*

6.3 Experiment 3

This experiment is conducted to study the efficiency of the proposed method with respect to the information received at the sink node during a data gathering round. For this purpose, we compare the number of packets actually received at the sink node using the proposed method and the number of packets which needed to be received at the sink node in each data gathering round. Figure 9 which gives this comparison shows that by applying the proposed method, number of packets received at the sink node in each data gathering round gradually approaches to its ideal amount, which is, the number of regions in the environment in that round. Peaks in this figure correspond to

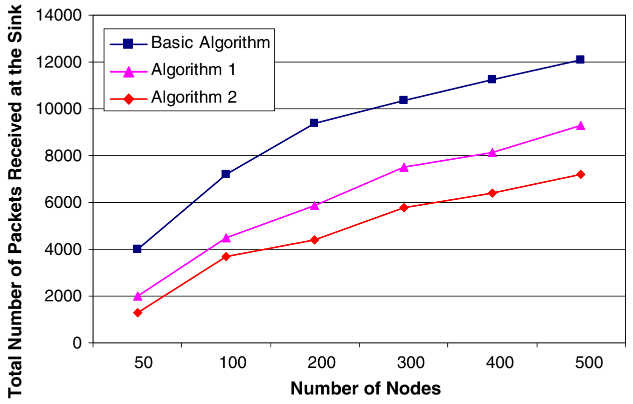

Fig. 10 Total number of packets received at the sink

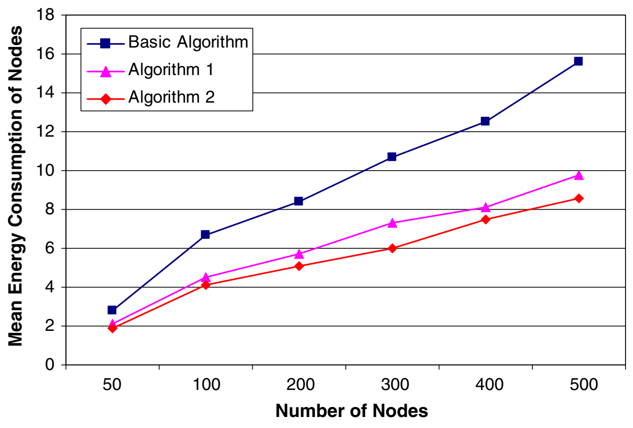

Fig. 11 Mean energy consumption of nodes

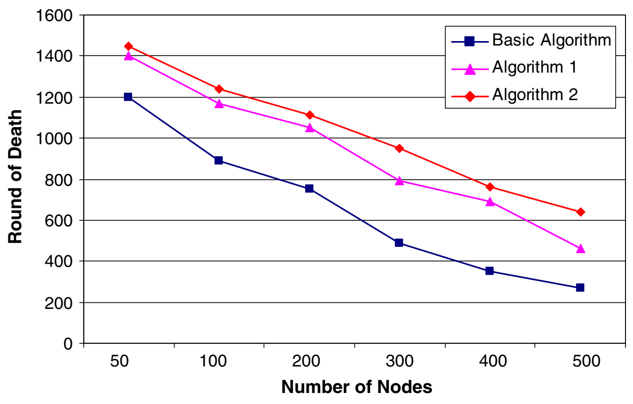

Fig. 12 Network lifetime based on the *Round of Death* metric

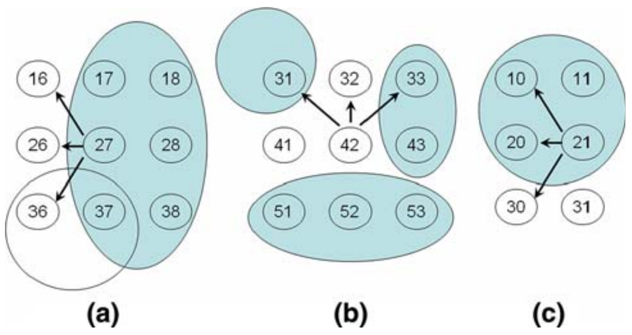

Fig. 13 **a** Environment around node number s_{27} and its neighbors, **b** Environment around node number s_{42} and its neighbors, and **c** Environment around node number s_{21} and its neighbors

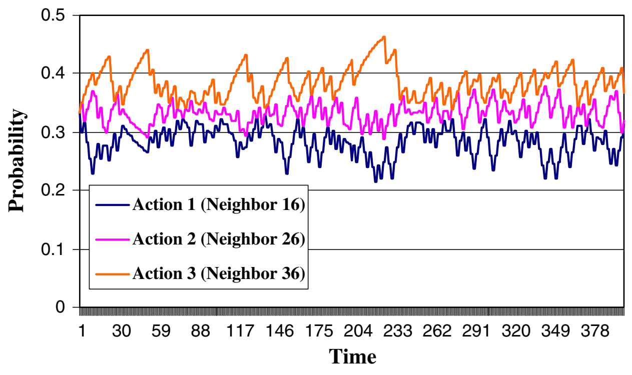

Fig. 14 Action probabilities for node s_{27}

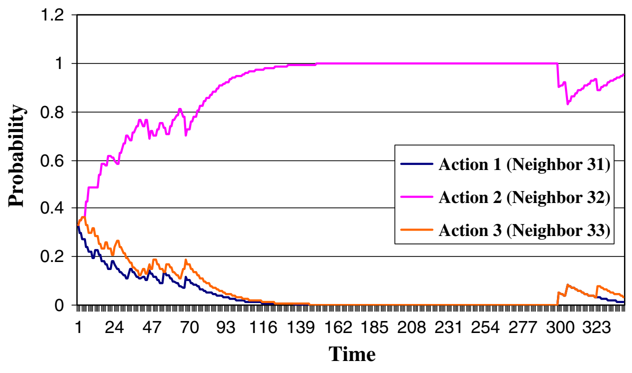

Fig. 15 Action probabilities for node s_{42}

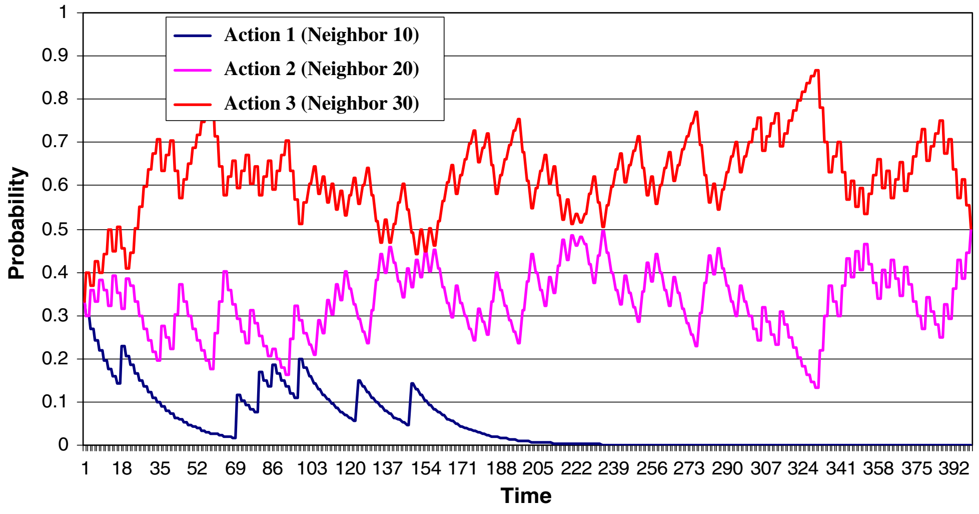

Fig. 16 Action probabilities for node s_{21}

sudden changes in the number of regions due to the movements of climates. It can be seen that the proposed algorithm learns new routing paths and hence adapts the network to these changes.

6.4 Experiment 4

In this experiment, the basic algorithm is compared with Algorithms 1 and 2 with respect to the mean energy consumption of nodes and the total number of packets received at the sink node. As it is shown in Figs. 10 and 11, comparing to the basic algorithm, Algorithms 1 and 2 both results in lesser number of received packets at the sink and lower mean energy consumption in each node. This indicates that higher data aggregation ratio can be obtained using Algorithms 1 or 2. Figure 12 compares basic algorithm, Algorithms 1 and 2 in terms of network lifetime. As before, *Round of Death* metric is used for this purpose. It can be seen that Algorithms 1 and 2 both result in longer lifetime for the network.

6.5 Experiment 5

This experiment is conducted to better understand the convergence behavior of learning automata residing in the nodes of the network. For this purpose we keep track of the action probability vectors of 3 learning automata residing in nodes s_{27}, s$_{42}$, and s_{21} of network of Fig. 13. RL_{27} has three entries corresponding to s_{16}, s_{26} and s_{36} as shown in Fig. 13(a). Thus *learning automaton* LA_{27} has three actions each corresponds to one of these nodes. None of s_{16}, s_{26} and s_{36} is in the same region as node s_{27}; and hence no aggregation can be performed. As a result, the action probability of none of the actions of LA_{27} approaches 1 as it is depicted in Fig. 14. RL_{42} has three entries s_{31}, s_{32} and s_{33} (Fig. 13(b)). Node s_{32} is in the same region as s_{42} and hence s_{32} can aggregate packets received from s_{42}. Therefore, as it shown in Fig. 15, the probability of action corresponding to s_{32} approaches one. Finally, RL_{21} has three entries s_{10}, s_{20} and s_{30} (Fig. 13(c)). Node s_{21} is in the same region as s_{10} and s_{20} and hence both s_{10} and s_{20} can aggregate packets received from s_{21}. As a result, the probability of the two actions corresponding to s_{10} and s_{20} approach 0.5 whereas the probability of the action corresponding to s_{30} approaches 0. This is depicted in Fig. 16.

7 Conclusion

In this paper we proposed a novel method based on learning automata for data aggregation in wireless sensor networks especially when the environment's changes can not be predicted beforehand. In this method each node in the network is equipped with a learning automaton. The learning automaton has a number of actions each of which corresponds to one of the neighbors of the node. The learning automaton for each node helps the node to find the next best hop for forwarding its packets toward the sink with the aim of performing as much as data aggregation as possible. It was shown through simulations that the proposed method outperforms the existing methods in terms of network's lifetime especially when the environment is highly dynamic.

References

1. Shah, R., & Rabaey, J. (2002). Energy aware routing for low energy ad hoc sensor networks. In *Proceedings of the IEEE*

Wireless Communications and Networking Conference (WCNC), Orlando, Florida, March.

2. Beaver, J., Sharaf, M. A., Labrinidis, A., & Chrysanthis, P. K. (2003). Location-aware routing for data aggregation in sensor networks. In *Proceedings of the 2nd Hellenic Data Management Symposium.*
3. Heinzelman, W., Chandrakasan, A., & Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of 33rd Hawaii International Conference on System Science (HICSS '00)*, January.
4. Beyens, P., Peeters, M., Steenhaut, K., & Nowe, A. (2005). Routing with compression in wireless sensor networks: a Q-learning approach. In *Fifth European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS 05)*, Paris, France.
5. Xu, Y., Lee, W. C., Xu, J., & Mitchell, G. (2006). Processing window queries in wireless sensor networks. In *IEEE International Conference on Data Engineering (ICDE'06)*, Atlanta, GA, April.
6. Rosemark, R., & Lee, W. C. (2005). Decentralizing query processing in sensor networks. In *The Second International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous'05)*, San Diego, CA, July (pp. 270–280).
7. Winter, J., Xu, Y., & Lee, W. C. (2005). Energy efficient processing of K nearest neighbor queries in location-aware sensor networks. In *The Second International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous'05)*, San Diego, CA, July (pp. 281–292).
8. Bontempi, G., & Le Borgne, Y. (2005). 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.
9. Liu, C., Wu, K., & Pei, J. (2005). 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.
10. Younis, O., & Fahmy, S. (2005). 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.
11. Virrankoski, R., & Savvides, A. (2005). TASC: Topology adaptive spatial clustering for sensor networks. In *Second IEEE International Conference on Mobile Ad Hoc and Sensor systems*, Washington, DC, November.
12. Soro, S., & Heinzelman, W. (2005). 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.
13. Lotfinezhad, M., & Liang, B. (2004). Effect of partially correlated data on clustering in wireless sensor networks. In *Proceedings of the IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*, Santa Clara, California, October.
14. Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., & Madden, S. (2004). Distributed regression: An efficient framework for modeling sensor network data, *Intel Corporation.*
15. Dasgupta, K., Kalpakis, K., & Namjoshi, P. (2003). An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In *IEEE Wireless Communications and Networking Conference*, March (Vol. 4, No. 1).
16. Radivojac, P., Korad, U., Sivalingam, K. M., & Obradovic, Z. (2003). Learning from class-imbalanced data in wireless sensor networks. In *IEEE Semiannual Vehicular Technology Conference, VTC-Fall 2003*, Orlando, Florida, USA, October (Vol. 5, pp. 3030–3034).
17. Haleem, M., & Chandramouli, R. (2005). 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). doi:10.1109/JSAC.2005.845636.
18. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2006). Exploiting locality of demand to improve the performance of wireless data broadcasting. *IEEE Transactions on Vehicular Technology,55*(4), 1347–1361. doi:10.1109/TVT.2006.877464.
19. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2003). Learning-automata-based polling protocols for wireless LANs. *IEEE Transactions on Communications,51*(3), 453–463. doi:10.1109/TCOMM.2003.809788.
20. Nicopolitidis, P., Papadimitriou, G. I., Obaidat, M. S., & Pomportsis, A. S. (2005). Carrier-sense-assisted adaptive learning MAC protocol for distributed wireless LANs. *International Journal of Communication Systems, Wiley,18*(7), 657–669. doi:10.1002/dac.724.
21. Nicopolitidis, P., Papadimitriou, G. I., & Pomportsis, A. S. (2004). Distributed protocols for ad-hoc wireless LANs: A learning-automata-based approach. *Ad Hoc Networks Journal Elsevier,2*(4), 419–431. doi:10.1016/j.adhoc.2003.09.004.
22. Ramana, B. V., & Murthy, C. S. R. (2005). Learning-TCP: A novel learning automata based congestion window updating mechanism for ad hoc wireless networks. In *Proceedings of 12th IEEE International Conference on High Performance Computing, December* (pp. LNCS 454–464).
23. Beigy, H., & Meybodi, M. R. (2002). A learning automata based dynamic guard channel scheme. In *Lecture Notes on Information and Communication Technology*, Springer Verlag, November (Vol. 2510, pp. 643–650).
24. Beigy, H., & Meybodi, M. R. (2003). An adaptive uniform guard channel algorithm: A learning automata approach. In *Lecture Notes in Intelligent Data Engineering and Automated Learning, Springer Verlag, LANCES 2690,* Berlin, Heidelberg, Germany (pp. 405–409).
25. Beigy, H., & Meybodi, M. R. (2009). Learning Automata based Dynamic Guard Channel Algorithms. *Journal of High Speed Networks* (to appear).
26. Golipour, M., & Meybodi, M. R. (2008). LA-mobicast: A learning automata based mobicast routing protocol for wireless sensor networks. *Sensor Letters,6*(2), 305–311. doi:10.1166/sl.2008.038.
27. Esnaashari, M., & Meybodi, M. R. (2008). A cellular learning automata based clustering algorithm for wireless sensor networks. *Sensor Letters, 6*(5), 723–735. doi:10.1166/sl.2008.m146
28. Esnaashari, M., & Meybodi, M. R. Dynamic point coverage in wireless sensor networks: A learning automata approach. In *Lecture Notes in Computer Science*, Springer Verlag, Kish Island, Iran (pp. 758–762, to appear).
29. Narendra, K. S., & Thathachar, M. A. L. (1989). *Learning Automata: An introduction*. USA: Prentice Hall.
30. Thathachar, M. A. L., & Sastry, P. S. (2002). Varieties of learning automata: An overview. *IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics,32*(6), 711–722. doi:10.1109/TSMCB.2002.1049606.

Author Biographies

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 from1985 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.