A Novel Algorithm Based on Weakly Connected Dominating Set and Learning Automata and Its Application to Clustering and Routing the Sensor Networks

Nayereh Abtahi Naseri
Department of Computer Engineering
Islamic Azad University
Arak, Iran
n.abtahi@iau-saveh.ac.ir

Mohammd Reza Meybodi
Department of Computer Engineering and IT
Amirkabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir

Javad Akbari Torkestani
Department of Computer Engineering
Islamic Azad University, Science and Research Branch
Tehran, Iran
j-akbari@iau-arak.ac.ir

Abstract—**A wireless sensor network consisting of a large number of small sensors with low-power transceivers can be an effective tool for gathering data in a variety of environments. Prolonged network lifetime and scalability are important requirements for many sensor network applications. Clustering is an effective topology control approach in wireless sensor networks, which can increase network scalability and lifetime. Clustering sensors into groups so that sensors communicate information only to cluster heads and then the cluster heads communicate the aggregated information to the processing center, may save energy. The weakly connected dominating set (WCDS) is very suitable for cluster formation. Then we find a suitable route for sending packets with a new routing algorithm with the help of the created cluster. In this paper, we propose a distributed, algorithm for WCDS construction in wireless sensor networks based on distributed learning automata. To evaluate the performance of the proposed algorithm several experiments have been conducted.**

Keywords-sensor network*; *clustering; distributed learning automata; weakly connected dominating set; routing

I. INTRODUCTION

A wireless sensor network consists of many sensor nodes. The sensors which are randomly deployed in the environment of a phenomenon play the role of gathering specific data from the environment, processing and finally sending it to the base station. Sensor networks have critical applications in the scientific, medical, commercial, and military domains [1]. One of the advantages of wireless sensors networks is their ability to operate unattended in harsh environments where contemporary human-in-the-loop monitoring schemes are risky, inefficient and sometimes infeasible. Given the vast area to be covered, the short lifespan of the battery-operated sensors and the possibility of having damaged nodes during deployment, large population of sensors are expected in most WSN applications. It is envisioned that hundreds or even thousands of sensor nodes will be involved [2]. Network lifetime can be defined as the time elapsed until the first node or the last node in the network depletes its energy (dies). Grouping sensor nodes into clusters has been widely pursued by the research community. Clustering the network efficiently is an important way of managing large number of sensor nodes. Clustering mechanisms being especially effective in increasing network scalability and reducing data latency, have been extensively exploited. In addition to supporting network scalability, clustering has numerous advantages. It can localize the route set up within the cluster and thus reduce the size of the routing table stored at the individual node [3]. Clustering can also conserve communication bandwidth since it limits the scope of inter-cluster interactions to CHs and avoids redundant exchange of messages among sensor nodes [4]. Moreover, clustering can stabilize the network topology at the level of sensors and thus cuts on topology maintenance overhead. Every cluster would have a leader, often referred to as the cluster-head. The rest of the paper is organized as follows. In the next section, overview on clustering algorithms. In Section III, preliminaries on dominating sets and weakly connected dominating set problem. In section IV learning automata and some of its variations are presented. Section V describes the assumptions and prime goals of the study. In section VI details of the proposed clustering algorithm are presented. Section VII describes details of the proposed routing algorithm. In section VIII, performance of the proposed algorithm is evaluated via simulations and then compared with some other clustering protocols. Finally, section IX concludes the paper and gives the directions of future work.

978-1-4577-0989-0/11/$26.00 ©2011 IEEE

II. Related work

In this section, we will briefly overview some of the existing clustering algorithms for wireless sensor networks and ad-hoc network. One of the most famous clustering algorithms introduced for sensor networks [5] is the low energy adaptive clustering hierarchy (LEACH) algorithm. In LEACH a predetermined fraction of nodes p, elect themselves as heads by comparing a chosen random number with a predefined threshold. After the heads are elected, they broadcast an advertisement message to the rest of the nodes in the network that are the new heads. Upon receiving this advertisement, all the non-cluster head nodes decide on the cluster to which they want to belong, based on the signal strength of the advertisement. HEED [6] is a protocol which periodically selects cluster heads according to a hybrid of the node residual energy and a secondary parameter through constant time iterations. It uses the primary parameter, i.e., residual energy to select an initial set of cluster heads and the secondary parameter *AMRP*, $AMRP = \frac{\sum_{i=1}^{M} MinPw\, r_i}{M}$ to "break ties" among them, where $MinPwr_i$ denotes the minimum power level required by a node $v_i, 1 \leq i \leq M$, to communicate with its cluster head, and M is the number of nodes within the cluster range. HEED achieves fairly uniform distribution of the cluster heads across the network. In hierarchical routing algorithms, at first a clustering scheme is applied to form a kind of hierarchy and afterward, the process of routing is divided into inter and intra-cluster phases. Some examples of these algorithms are "Fixed size cluster routing, TEEN, and APTEEN, and AIMRP[7]. ACE [8] is an emergent algorithm that uses just three rounds of feedback to form an efficient cover of clusters across the network. It uses the node degree as the main parameter to elect cluster heads.

The WCDS was first proposed for clustering adhoc networks by Chen and Liestman [9]. The proposed distributed approximation algorithm is also inspired by Guha and Khuller's centralized algorithm [10] for a CDS formation. However, this distributed algorithm is not localized because it uses some central coordinators to direct the algorithm execution. To address the problem of non-localized computation, Chen and Liestman also proposed a zonal algorithm [11], hereafter referred as Zonal. In this approach, they divided the graph into regions, constructed a WCDS for each region, and made adjustments along the borders of the regions to produce a WCDS for the whole graph. Their algorithm for the partitioning phase is partly based on a Minimum Spanning Tree (MST) algorithm of Gallageret et al. [12]. Akbari and Meybodi in [13] promoted the idea of clustering the Ad-Hoc network based on the weakly connected dominating sets with learning automata.

III. Preliminaries dominating sets and weakly connected dominating set

This section presents a brief review of the existing literatures for WCDS and CDS construction [13]. Wireless Ad-Hoc networks can be modeled as a unit disk graph $G(V, E)$, with the hosts representing the individual hosts and an edge connecting two hosts where the corresponding hosts are within transmission range of each other. Furthermore, a dominating set of the graph $G(V, E)$is a host subset $S \subseteq V$, in a way that every host $v \subseteq V$ is either in S or adjacent to a host of S. A minimum connected dominating set is also a connected dominating set with the minimum cardinality. An minimum connected dominating set forms a virtual backbone in the graph through which the routing overhead can be significantly reduced as the number of hosts responsible for routing can be reduced to the number of hosts in the backbone [14, 15]. A dominating set S is called a weakly connected dominating set of a graph G, provided that the graph $< S >_W = (N[S], E \cap (N[S] \times S)$ is a connected sub-graph of G.

IV. Learning automata , Distributed learning automata, A variable action-set learning automaton

A. Learning automata

Learning automatons are adaptive decision-making units. Through repeated interactions with a random environment, a learning automaton improves its performance by learning how to select the optimal action from a finite set of actions allowed. Based on a probability distribution kept over the action-set, the action is chosen randomly and at each instant the given action is used as the input to the random environment. The environment in turn, responds to the taken action with a reinforcement signal. Based on the reinforcement feedback from the environment, the action probability vector is then updated[16, 17]. The relationship between the learning automaton and its random environment has been shown in Fig.1.

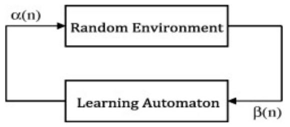

Figure 1. The relationship between the learning automaton and its random environment

There are two main families of learning automata: fixed structure learning automata and variable structure learning automata [16-18]. Variable structure learning automata are shown by a triple$< \beta, \alpha, T >$, where β is the set of inputs, α is the set of actions, and T is the learning algorithm which is a recurrence relation used to modify the action probability vector. Assume that $\alpha(K)$ and $P(K)$ denote the action chosen at instant K and the action probability vector on which the chosen action is based, respectively. The action probability vector P is updated by a linear learning algorithm shown in recurrence equation (1) and (2). Let $\alpha_i(K)$ be the action chosen by the automaton at instant K.

$$P_j(n+1) = \begin{cases} P_j(n) + a[1 - P_j(n)] & j = i \\ (1-a)P_j(n) \quad \forall j & j \neq i \end{cases} \qquad (1)$$

When the taken action is rewarded by the environment (i.e.$\beta(n) = 0$) and

$$P_j(n+1) = \begin{cases} (1-b)P_j(n) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)P_j(n) & \forall j \quad j \neq i \end{cases} \tag{2}$$

B. Distributed learning automata

Distributed learning automata is defined as a network of the learning automata that collectively cooperate to solve a particular problem [17]. This network is formally defined by a quadruple$< A, E, T, A_0 >$, where $A = \{A_1, \dots, A_n\}$is the set of learning automata, $E \subset A \times A$ is the set of the edges where edge $e_{(i,j)}$ corresponds to the action α_{ij} of the automaton A_I, the set of learning schemes with which the learning automata update their action probability vectors is represented by T, and finally $A_0$is the root automaton of Distributed learning automata from which the automaton activation is started .

C. Variable action-set learning automaton

In a variable action-set learning automaton, the number of actions available at each instant changes with time. In this learning automaton there is a finite set of n actions, $\alpha = \{\alpha_1, \dots, \alpha_n\}$. $A = \{A_1, \dots, A_m\}$ represents the set of action subsets and $A(n) \subseteq \alpha$ is the subset of all the actions that can be chosen by the learning automaton, at each instant n . Each action subsets is randomly selected by an external agency according to the probability

$P_i(n) = prob[\alpha(n) = \alpha_i | \,]$

The probability of choosing action α_i, conditioned on the event that the action subset $V(n)$ has already been selected and also $\alpha_i \in V(n)$ and $V(n)$ is set of active actions. The scaled probability $P_i(n)$ is defined as:

$$P_i(n) = \frac{p_i(n)}{K(n)} \tag{3}$$

Where $K(n) = \sum_{\alpha_i \in V(n)} p_i(n)$ is the sum of the probabilities of the actions in subset $V(n)$, and $P_i(n) = prob[\alpha(n) = \alpha_i]$.

A variable action-set learning automaton selects an action and updates the action probabilities through the following procedure. Suppose that $V(n)$ is an action subset selected at instant n. At first, probabilities of all the actions in the selected subset are scaled before selecting an action. This is done as defined in (3). Then based on the scaled action probability vector$P_i(n)$, the automaton randomly chooses one of its possible actions. The learning automaton then updates its scaled action probability vector depending on the response received from the environment[19].Upon receiving a rewarding of environment, updates its action probability vector as

$$P_i(n+1) = P_i(n) + a[1 - P_i(n)] \quad \alpha(n) = \alpha_i \tag{4}$$

$$P_i(n+1) = P_j(n) + a.P_i(n) \qquad \alpha(n) = \alpha_i \,, \forall\, k \quad k \neq j$$

Upon receiving a penalizing of environment, updates its action probability vector as

$$P_i(n+1) = (1-b)P_i(n) \quad \alpha(n) = \alpha_i \tag{5}$$

$$P_i(n+1) = P_j(n) + a.P_i(n) \quad \alpha(n) = \alpha_i \quad \forall\, k \quad k \neq j$$

Then, probability vector of the actions of the chosen subset is rescaled as

$$P_j(n+1) = P_j(n+1).K(n) \qquad for\ all\ j, \alpha_j \in V(n) \tag{6}$$

$$P_j(n+1) = P_j(n) \qquad for\ all\ j, \alpha_j \notin V(n)$$

V. Describes the Assumptions and Criteria Used and Our Prime Goals

In this algorithm, a network of learning automata isomorphic to the input unit disk graph $\mathrm{G(V, E)}$ is initially formed by assigning to each vertex (e.g., v_i) of the graph a learning automaton (e.g., A_i). The resulting network of the learning automata can be described by a duple$< A, \alpha >$, each vertex (or learning automaton) can be in one of two states *active* and *passive*, and is initially set to the passive state. In this method domination set is called the cluster-head set. The dominator nodes play the role of the cluster-heads and their one-hop neighbors assume the role of the cluster members. Each host requires the following data structures to participate in the cluster formation process: *T_energy*, Average predetermined threshold energy network required for termination the cluster formation process as the probability of choosing the cluster-head set. *Min_energy*, The minimum energy of all nodes for clustering. *max_repeat*, A stopping condition for the algorithm as a maximum number of iterations. *Collection_cluster_head*, A set of the chosen cluster-heads at each iteration. *Collection_memmber*, A set of hosts in which each member is a one-hop neighbor of at least one host in the *Collection_cluster_head. prob_vector*, A vector of the probability of choosing the members of *Collection_cluster_head. min_size*, A dynamic threshold contains the cardinality of the smallest *Collection_cluster_head* which has been selected yet. *iteration_num*, A counter which keeps the number of constructed *Collection_cluster_head.*

VI. The Proposed Clustering Algorithm

In this section, we propose a distributed approximation algorithm based on distributed learning automata clustering the wireless sensor networks. The proposed clustering algorithm consists of three phases.

A. Phase I

The first phase deals with setting up the network and recognizing neighborhood. After dispersing the sensors, the first step in cluster formation is recognition of the neighbors. Each host in the in the network (H_i) is assigned to a learning automaton (A_i). A HELLO – PACKET massage is then sent to start the procedure. Any sensor that receives the message, i.e., is within the receiving range of the node, would be considered a neighbor of the sending node. Then through sending a message which comprises the node's ID fields and energy level field, the message is replied. Upon receiving a reply from each neighborhood node, the transmitter of HELLO-PACKET message determines the number of neighbors and the energy level of the nodes.

B. *Phase II*

In the second phase, weakly connected dominating set and clusters are formed. This phase consists of a number of iterations and each iteration a weakly connected dominating set is constructed by randomly activating a number of automata. In the proposed algorithm, to form the action-set of learning automaton A_i, its corresponding host (i.e., host H_i) sends a message locally to its one-hop neighbors. The hosts within the transmission range of the sender reply the message once they receive it. In fact the host forms its action-set on the basis of the received replies. Each host by which the message is replied is associated with an action. Let $\alpha_i = \{\alpha_{i,j} | H_j$ is a neighbor of H_i denotes the action-set of learning automaton A_i. Action $\alpha_{i,j}$ corresponds to the selection of host H_j as a cluster-head by host H_i .As a result of this action-set formation method, some learning automata will have the same actions. A problem of these common actions is selecting the redundant dominators (the dominators by which no more dominate nodes are spanned) and also the same dominators by different automata. The proposed algorithm solves these problems by dealing with the learning automaton with changing number of actions [19]. After the automata action sets are determined, $H_i$which has the highest energy level in the network nodes, activates its corresponding automata, i.e., A_i . This automata then randomly selects one of its actions(selecting the first dominating set).The probability with which learning automaton A_i chooses this action is added to the *prob_vector*. Host H_i then sends an *activation* message to the host (new cluster-head) corresponding to the chosen action.When a given host H_i receives an *activation* message,the state of a given host changes to the active state.It then inserts its ID number as a new cluster-head into the *Collection_cluster_head*, provided that at least one of its one-hop neighbors is not in the *Collection_memmber*. This prevents selecting redundant cluster-heads. To update the *Collection_memmber* it adds its one-hop neighbors' ID to this list. Disabling the actions associated with the selected cluster-heads, updates the action-set of learning automaton A_i. As mentioned before, this avoids selecting the same cluster-heads by different automata, and as a result improves the convergence speed of algorithm. In case there are moreactions that can be taken by learning automaton $A_i$and the *Collection_memmber* does not include all the hosts, and averageenergy ofthe nodesin thenetworkis more than the pre-specified*min-energy*,currently activated automaton A_i chooses one of its actions as a new cluster-head, updates *prob_vector* by adding the choice probability of this action, and sends an *activation* message to the chosen cluster-head. Otherwise, if the size of the *Collection_memmber* equals to the network size and the size of the *Collection_cluster_head* is less than or equal to dynamic threshold *min_size*, the dynamic threshold is set to the cardinality of the selected *Collection_cluster_head* and all the chosen actions of the activated automata are rewarded by sending back a *rewarding* message, otherwise they (i.e., the chosen actions of the activated automata) are penalized by sending back a *penalizing* message.At the end of each iteration, after the activated automata is rewarded or penalized, the action probability vector is updated once again (or rescaled). This is done by enabling all the disabled actions according to the rescaling method described in the variable action-set learning automata. Also at this stage, it should be verified if stopping condition of the cluster formation (or cluster maintenance) process is met. In other words, it should be checked if the choice probability of the *Collection_cluster_head* is less than the certain threshold *T_energy* or if *iteration_num*exceeds the per-specified threshold *max_repeeat*. The choice probability of the *Collection_cluster_head* is calculated as the product of the choice probabilities of the selected cluster-heads based on the information contained in *prob_vector*. If the stopping condition is true, currently active automaton A_i initiates a new iteration, randomly chooses a new cluster-head, and sends an *activation* message to it. Otherwise, it generates a *clustering* message including the last selected *Collection_cluster_head* and broadcasts it within the network.Upon receiving a *rewarding* message, the activated host H_i updates its action probability vector by rewarding chosen action $\alpha_{i,j}$ as

$$P_{i,j}(n+1) = P_{i,j}(n) + a[1 - P_{i,j}(n)], \qquad (7)$$

where $P_{i,j}$ is the probability with which host H_i chooses host H_j as a cluster-head, and penalizing the other actions $\alpha_{i,k}$, for all $k \neq j$, as

$$P_{i,k}(n+1) = (1-a)P_{i,k}(n) \qquad \forall k \;\; k \neq j\,. \qquad (8)$$

When activated host H_i receives a *penalizing* message, it updates its action probability vector by rewarding chosen action $\alpha_{i,j}$ as

$$P_{i,j}(n+1) = (1-b)P_{i,j}(n), \qquad (9)$$

where $P_{i,j}$ is the probability with which host H_i chooses host H_j as a cluster-head, and penalizing the other actions $\alpha_{i,k}$, for all $k \neq j$, as

$$P_{i,k}(n+1) = \left(\frac{b}{r-1}\right) + (1-b)P_{i,k}(n) \; \forall k \;\; k \neq j. \qquad (10)$$

C. *Phase III*

The third phase is that of re-clustering. Re-clustering phase is initiated when cluster heads do not have the necessary energy to send the data. In other words, the energy level is lower than the predetermined threshold level and actually degrades to a specified percent

(*RECLUSTERING_PERCENT*) below the energy level at the time it was selected as a head.

The re-clustering process is similar to the initial clustering. There are differences between ad hoc and sensor networks, e.g., the sensors are not fast enough. Therefore, the re-clustering phase is not carried out as many times as that in the ad hoc networks. Within the same ranges weakly connected dominating set are only established when the average energy level in the network is reduced to a certain level. During the initial clustering, the choice probability of each cluster-head set candidate grows proportional to its optimality among the other candidates and as a result the overhead of the re-clustering phase is significantly smaller as compared with the initial clustering phase. Comparing with the initial clustering phase, re-clustering phase requires a significantly smaller number of iterations for finding the new optimal cluster-head set.

VII. The Proposed Routing Algorithm

Initially, learning automata will be assigned to each node cluster head in the clustering. On the other hand, each of cluster head node in time of clustering knows all of its ominated set and the energy level of each node.Therefore, the initial probability vector of the learning automata action in cluster head node is calculated based on the number of nodes in the dominate set any cluster head (i.e. single-hop neighbors) and the energy level of nodes as follows:

$$P_i = \frac{EnergyLevel_i}{\sum_{j=1}^{N} EnergyLevel_j}$$

P_i : the probability of selecting the *ith* neighbor node; $EnergyLevel_i$: the energy level of the *ith* neighbor node; N: the same number of nodes in each cluster (dominate set per cluster)Probability assigned to nodes, is a initial probability that changes during the life of the network and goes towards selection of higher reliability nodes (higher energy); In fact using learning automata, nodes learn to transmit information from the path that has a higher energy level. Each host requires the following data structures to participate in the routing formation process:

Max-it: A counter which keeps the number of sending to cluster head. Each time it sends the package node to its cluster head this counter increases. *Th-energy*: A threshold of the average predetermined energy related to cluster head

num-cluster: the number of constructed cluster heads of clustering phase. This algorithm in an iterative process including the following steps will send the data. However, until cluster head node has the required energy. In other words, until cluster head energy is not less than pre-determined threshold Th-energy. As soon as the above condition will be violated, or the phase should re-cluster or do WCDS construction again.

First step: first the node that is selected as source, sends a message to the cluster head node for sending data packet. As soon as cluster head receives packet, puts its ID node in the routing table. When a node sends a packet to its cluster head, cluster head will compare energy level node with the average energy nodes cluster and based on the node energy level that the packet it received from, updates its action probability vector according to rules learning. Here we have two states: If the current node energy level is higher than the average energy of other nodes in the cluster

$$EnergyLevel_i > \frac{\sum_{j=1}^{N} EnergyLevel_j}{N}$$

In this case, the mentioned action is rewarded and the probability of its selection is updated according to the following learning rules.

$$P_i(n+1) = P_i(n) + a[1 - P_i(n)] \quad (11)$$
$$P_j(n+1) = (1-a)P_j(n) \quad \forall j \;\; j \neq i$$

Otherwise, if ($EnergyLevel_i \leq \frac{\sum_{j=1}^{N} EnergyLeve\ l_j}{N}$)

The mentioned action is penalized and the probability of its selection is updated according to the following learning rules.

$$P_i(n+1) = (1-b)P_j(n) \quad (12)$$
$$P_j(n+1) = \left(\frac{b}{r-1}\right) + (1-b)P_j(n) \quad \forall j \;\; j \neq i$$

Learning automata of the cluster head whose ID is cluster member nodes will send the packet to that node if the data packet destination is a member of its cluster. And the mentioned node will send the ACK message and data packet is sent to destination according to routing table. Otherwise it goes to the next step. Third stage: cluster head the of the member nodes that belong to other clusters is according to aweak connection on the first phase will be selected randomly. Step Four: Now this node select the next cluster heads and cluster head adds ID node to routing table. and *Max-it* variables increases. If *Max-it* will be greater than *num-cluster,* the algorithm starts a new data iteration to send data and goes to the first stage. Otherwise it goes to the second stage. In subsequent sent the data packets with change in the destination of the packets in each send, data packets a node will be selected to send data packets, which its probability is higher than other nodes in the desired cluster. Therefore, energy consumption reduced.

VIII. Experimental results

In this section, we have conducted several simulations to study the performance of the proposed algorithms. All simulations have been implemented using glomosim simulator. Nodes are randomly deployed in a 1000(m)× 1000(m), and initial energy level of nodes is selected uniformly and 227 mwh. Simulations are performed for 100, 200, … , 1000 nodes. The results are averaged over 25 runs. In all simulations, the learning parameter is fixed at a=b=0.2.

Experiment 1

There exists an optimal value for *a* and *b* learning automata parameters which the proposed algorithm gives the minimum in term of energy consumption. As seen in Fig. 2 for value of *a=b=0.2*, energy consumption is optimal.

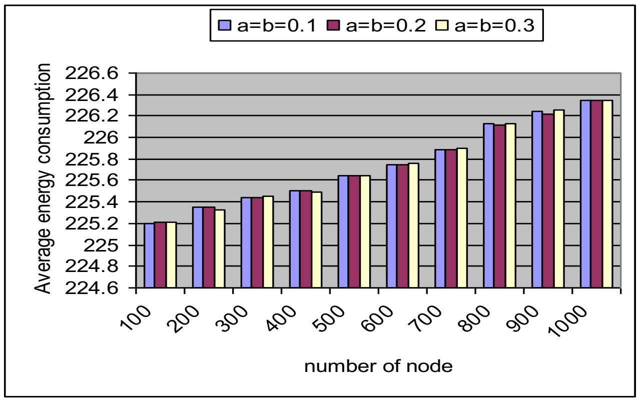

Figure 2. Average Energy Consumption Proposed Algorithm withdifferent values for *a and b*

EXPERIMENT 2

In this experiment, Fig. 3 we compare the Average total network energy consumption with LEACH, HEED

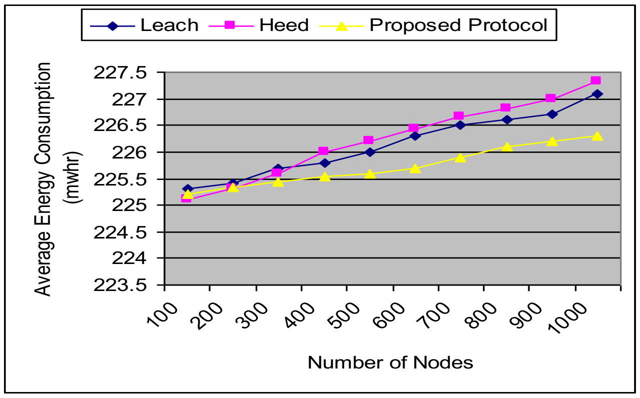

Figure 3. Comparison of clustering methods with average energy consumed

EXPERIMENT 3

In this experiment, we compares of individual nodes remaining energy level changes with the number of nodes in the network. This parameter can be specified by the balance in the network. Fig. 4 gives the result of this comparison.

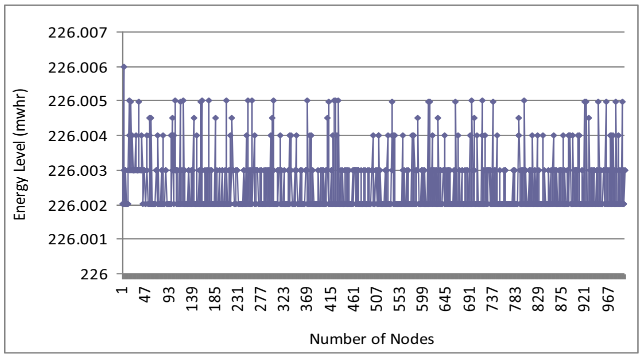

Figure 4. Comparison of changes in energy level of individual nodes

EXPERIMENT 4

In this experiment we compare the proposed clustering algorithm in terms of average remaining energy by existing algorithms. Fig. 5 gives the result of this comparison.

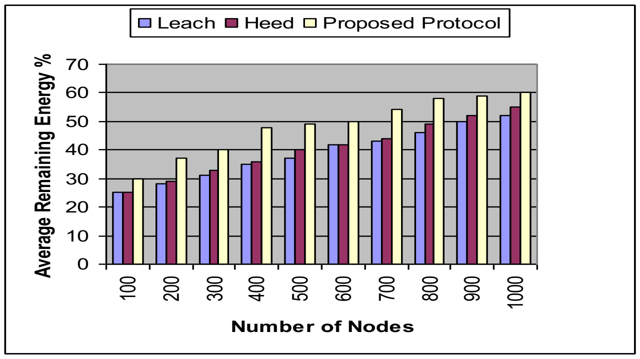

Figure 5. Comparison the average total energy used in different network

EXPERIMENT 5

This experiment whose result is given in Fig. 6 compares the network lifetime for the proposed algorithm with LEACH, HEED algorithms.

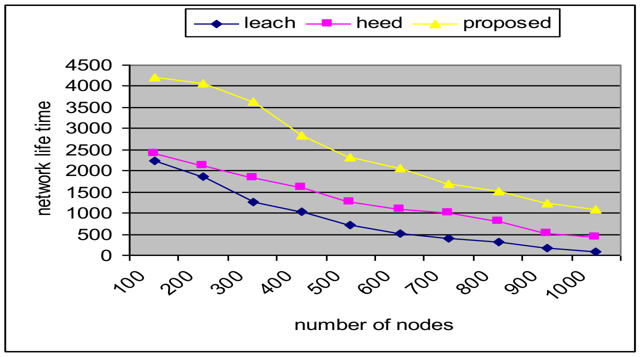

Figure 6. Comparison of clustering methods with respect to network lifetime.

EXPERIMENT 6

In this experiment, we compares the number of generated clusters using the proposed method with LEACH, HEED. Fig. 7 gives the result of this comparison.

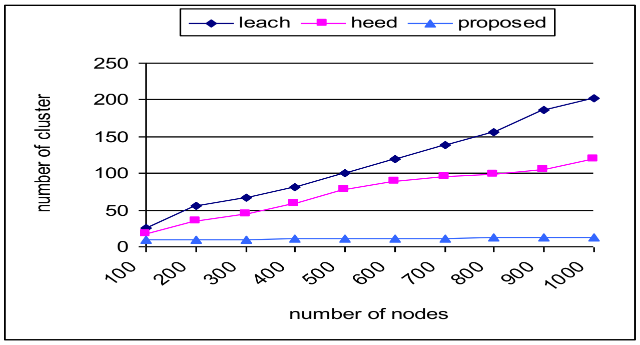

Figure 7. Comparison of the number of clusters resulted from the clustering methods

EXPERIMENT 7

In this experiment, we compares the average End-to-end delay of generated routing using the proposed method and proposed method, LEACH, HEED in terms of average End-to-end delay.Figure 8 gives the result of this comparison.

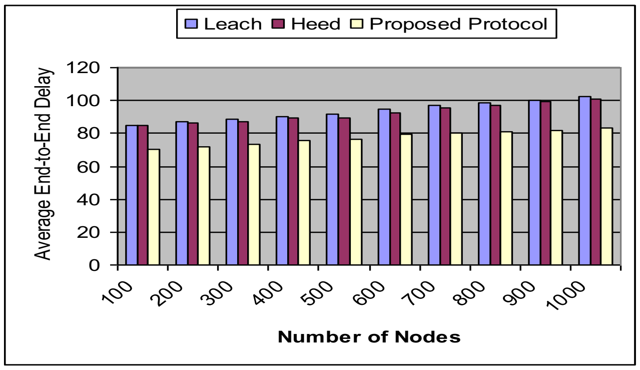

Figure 8. Comparison of the average End-to-end delay

IX. CONCLUSION

In this paper, a new clustering approximation algorithm for sensor networks based on learning automata was proposed. In the proposed algorithm, the dominators assume the role of the cluster-heads and their one-hop neighbors play the role of the cluster members. Then regarding created clusters, by a routing algorithm sending packets from source to destination is performed. To evaluate the performance of the method, several experiments were and the results obtained were compared with the results obtained for LEACH, the basic HEED in terms of mean total network energy consumption, remaining energy level of individual nodes, mean remaining energy, network lifetime, number of cluster, average End-to-End delay. Experiments showed that the proposed clustering algorithm outperforms the existing clustering algorithms.

REFERENCES

[1] I. F. Akyildiz, W. Su. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks (Elsevier) Journal*, vol. 38, pp. 393-422, 2002.

[2] R. Burne, et. al., "A self-organizing, cooperative UGS network for target tracking, in: Proceedings of the SPIE Conference on Unattended Ground Sensor Technologies and Applications II," Orlando Florida, April 2000.

[3] K. Akkaya, M. Younis, "A survey on routing protocols for wireless sensor networks," Elsevier Journal of Ad Hoc Networks, vol 3,no 3, pp. 325–349, 2005.

[4] K. Akkaya, M. Younis, "A survey on routing protocols for wireless sensor networks," Elsevier Journal of Ad Hoc Networks vol 3,no 3, pp. 325–349, 2005.

[5] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks,"*Proceedings of the 33rd International Conference on System Sciences (HICSS'00)*,(2000).

[6] O. Younis, S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Network," *IEEE Transactions on Mobile Computing*, 2004, vol 3, no 4, pp. 660–669.

[7] S.Kulkarni, A.Iyer , and C.Rosenberg, "An Address-light, Integrated MAC and Routing Protocol for Wireless Sensor Networks,"in *IEEE/ACM Transactions on Networking*," Vol 14, Issue 4, Aug. 2006, pp. 793 - 806.

[8] H. Chan, A. Perrig, "ACE: An Emergent Algorithm for Highly UniformCluster Formation,"*Proceedings of the First European Workshop on Sensor Networks(EWSN)*.

[9] Y.Z. Chen, A.L. Liestman, "Approximating minimum size weaklyconnecteddominating sets for clustering mobile ad hoc networks," inProceedings of the Third ACM International Symposium on MobileAd Hoc Networking and Computing (MobiHoc'2002), June 2002, pp.157–164.

[10] S. Guha, S. Khuller, "Approximation algorithms for connected dominatingsets," Algorithmica , vol 20, no 4, (1998),pp. 374–387.

[11] Y.Z. Chen, A.L. Liestman, "A zonal algorithm for clustering adhocnetworks,"*International Journal of Foundations of Computer Scienc*, Vol 14, no 2, (2003), pp. 305–322.

[12] R.G. Gallager, P.A. Humblet, P.M. Spira, "A distributed algorithm forminimum-weight spanning tree," ACM Trans. Programming Languages Syst, vol 5, no 1, (1983), pp. 66–77.

[13] J. Akbari Torkestani, M. R. Meybodi, "Clustering the wireless Ad Hoc networks: A distributed learning automata approach", *Journal of Parallel and Distributed Computing*, April. 2010, Vol. 70, no. 4, pp. 394-405

[14] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit Disk Graphs," *Discrete Mathematics*," Vol. 86, pp. 165-177, 1990.

[15] M.V. Marathe, H. Breu, H.B. Hunt III, S.S. Ravi, D.J. Rosenkrantz, "Simple Heuristics for Unit Disk Graphs," *Networks* Vol. 25, pp. 59–68, 1995.

[16] K. S. Narendra and K. S. Thathachar, "Learning Automata: An Introduction," New York, *Printice Hall*, 1989.

[17] H. Beigy, M. R. Meybodi, "Utilizing Distributed Learning Automata to Solve Stochastic Shortest Path Problems," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 14, pp. 591-615, 2006.

[18] K. S. Narendra, and M. A. L. Thathachar, "On the Behavior of a Learning Automaton in a Changing Environment with Application to Telephone Traffic Routing," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-l0, No. 5, pp. 262-269, 1980.

[19] M. A. L. Thathachar, R. H. Bhaskar, "Learning Automata with Changing Number of Actions", *IEEE Transactions on Systems, Man and Cybernetics*, 1987, Vol. 17, No. 6.