Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 1 (2012) 77 – 84

Procedia Technology

INSODE 2011

Applications of learning automata in wireless sensor networks

Jalil Jabari Lotf[a], Mehran Hosseinzadeh[b],seyed hossein hosseini nazhad ghazani[c], Rasim M. Alguliev[d]*

[a,c,d]*Institute of Information Technology of Azerbaijan National Academy of Science, Baku, Azerbaijan*
[b]*Mehran Hosseinzadeh,Department of Computer Engineering,University College of Nabi Akram,Tabriz, IRAN*

Abstract

We are dealing with a host of challenging issues in wireless sensor networks like: limited energy supply of a sensor, which is a threat for network life time, QOS needs and energy efficient routing. Researchers have proposed a variety of approaches to deal with these issues, but when it comes to dynamic, unpredictable environments, some of these solutions lose their effectiveness. The question is whether there is an adaptive approach, which could adapt itself with the changing nature of the environment? "Learning Automaton" is the answer to this question. If we equip the sensors with some intelligent automata, our network earns the ability of adapting itself with the environment when it changes. In other words, the WSN uses the power of learning and becomes intelligent as time passes. In this paper we introduce several proposed algorithms which use the learning automata to deal with the issues mentioned above, intelligently.

Keywords: Learning Automata;wireless sensor networks;routing protocol

1. Introduction

A sensor network is composed of many sensor nodes which have been distributed randomly in the area in order to gather special information from the environment, process the gathered information and finally send it to main nodes of information collectors [1]. A learning automaton is a self-operating machine or in other words, an abstract model which consists of a finite set of actions, internal states, a probability vector corresponding to its possible actions, and an output function. LA is connected in a feedback loop to the environment as shown in Fig.1:

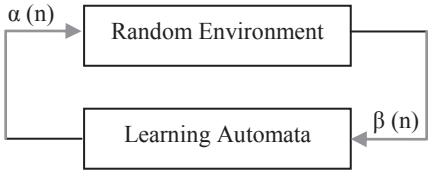

Fig .1. The relation between learning automata and the environment

At first, the learning automaton selects an action from its action set and performs it on its surrounding environment. Then the environment evaluates this action and responses to the automaton with a reinforcement signal. Based on the selected action, and received signal, the automaton updates its internal state and selects its next action. Thus we

* Jalil Jabari Lotf. Tel.: +98-914-410-7116; fax: +98-411-477-1095.
E-mail address: jalil.jabari@gmail.com.

doi:10.1016/j.protcy.2012.02.015

can say that LA is decision makers. The word "learning" is defined as any permanent change in behavior as a result of past experience, and a learning automaton should therefore have the ability to improve its behavior as time goes by, toward a final goal, which is choosing the best action in specific situations.
One main advantage of learning automaton is that it needs no knowledge of the environment in which it operates or any analytical knowledge of the function to be optimized. It's all because of the learning ability which makes LA an adaptive system which can adapt itself with the random environment and choose the best action. When the automaton first starts its job, naturally it has no knowledge about the environment which it is operating in. So it considers the probability of all actions to be equal. Say the number of possible actions in the action set is **r**, as a result, the probability corresponding to each of these actions equals to **1/r**. In other words the action probability vector *p(n)=Probability {α(n) = α i}* is given by :

$$P_i(n) = 1/r \quad (i=1,2,\ldots,r) \tag{1}$$

After taking responses from the environment, the automaton updates its probability vector. This means that it increases or decreases the probabilities of some actions according to the feedback received from the environment, or in other words the automaton learns which action is better than the others. The environment can be defined by the triple **E={α,β,c}** where **α={ α1, α2,…, αr }** is the action vector which is the finite input set of the environment, **β={ β1, β2,…, βr }** is the output set, and finally **c={c1, c2,…, cr}** is a set of penalty probabilities where each element $\mathbf{c_i}$ corresponds to one input of action $\boldsymbol{\alpha_i}$.
The output **β(n)** from the environment is an element of the set **β** and can take one of the values **β1** and **β2**. In the simplest case, the values of **βi** are chosen to be 0 or 1, where 1 is associated with failure/penalty response. The elements of **c** are defined as:

$$Probability\ \{ \beta(n)=1 \mid \alpha(n)=\alpha i \} = ci \qquad (i=1,2,\ldots,r) \tag{2}$$

According to the above equation, **ci** is the probability that the action **αi** will result in a penalty input from the environment. 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 **α ={ α1,…, αn }** represents the action set of the automata, **β={ β1,…, βn}** represents the input set, **P={ P1,…, Pn}** represents the action probability set, and finally:

$$P(n+1) = T[\alpha(n),\ \beta(n),\ p(n)] \tag{3}$$

Represents the learning algorithm. This automaton operates as follows. Based on the action probability set **p**, automaton selects an action **αi** , and performs it on the environment. After receiving the environment's reinforcement signal, automaton updates its action probability set based on equations (4) for favorable responses, and equations (5) for unfavorable ones:

If the environment response is favorable

$$Pi(n+1)=pi(n)+a[1-pi(n)] \tag{4}$$
$$Pj(n+1)=(1-a)pj(n) \qquad for\ all\ j,\ j\neq i$$

Else If the environment response is unfavourable

$$Pi(n+1)=(1-a)pi(n) \tag{5}$$
$$Pj(n+1)=b/(r-1)+(1-b)pj(n) \quad for\ all\ j,\ j\neq i$$

In these two equations, **a** and **b** are reward and penalty parameters respectively. **Pi(n)** means the probability of selecting action **αi** in $\mathbf{n^{th}}$ selection. For example if the response of the environment is favorable, action **αi** is rewarded, and other actions are panelized by equation (4). Meaning that the probability of selecting action **αi** in $\mathbf{(n+1)^{th}}$ selection increases.

2. Applications Of Learning Automata

2.1. A Learning Automata Based Data Aggregation Method

The WSNs are very sensitive to the amount of energy consumption, meaning the more energy consumption, the less network life time. One possible way to reduce energy consumption is to reduce the number of data packets being transmitted in the network. There is a very famous technique named data aggregation. 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. It is obvious that more related information along the path in which each packet travels will cause in higher aggregation ratio. So we must find a path with the most related information in order to reach the maximum data aggregation ratio. This is a very simple task when the nodes with related information do not vary in whole network life time. But if these information change during time, it is a complicated task. In situations like this, when we are dealing with a dynamic environment, all nodes must adapt themselves to these information changes in both, themselves and their neighbors, in order to find the optimal path in terms of maximum data aggregation ratio. The learning automata are used for this purpose. Each node is equipped with a learning automaton and after performing the routing algorithm mentioned in [1], each node (**node (i)**) has a list of its neighbors which is going to be used to forward the packets to the sink. Earlier we said that there is a finite set of actions in a learning automaton, here the action is to select one of the node's neighbors in order to forward the packets to the sink. The action selected by the automaton will be rewarded or penalized based on the acknowledgment it receives from the selected neighbor (**neighbor (k)**) in response to the sent packet. This acknowledgement tells the automaton that to what extent the information in the sent packet was related to the information in that neighbor (k). This ACK is the data aggregation ratio or in abbreviation DAR, and based on that the automaton of node (i) penalizes or rewards the selected action-k(selecting K^{th} neighbor). If DAR is greater than a specified threshold the selected action-k is rewarded, and if it is not, action-k is penalized. And by this procedure the probability vector of actions set is updated. So for the next neighbor selection, node(i) will choose a neighbor with the highest probability from the neighbor list. Using the proposed method, each node gradually learns the best neighbor (the neighbor with the most related data) for forwarding its packets toward the sink. The main goal of this procedure is to reduce the amount of energy consumption. In [1] these four methods are compared in terms of total energy consumption. i) without aggregation, ii) data aggregation with no learning, iii) data aggregation using Q-learning and iiii) proposed method in[1]. The results show that the method mentioned above outperforms all other three methods, especially when the environment is highly dynamic [1]:

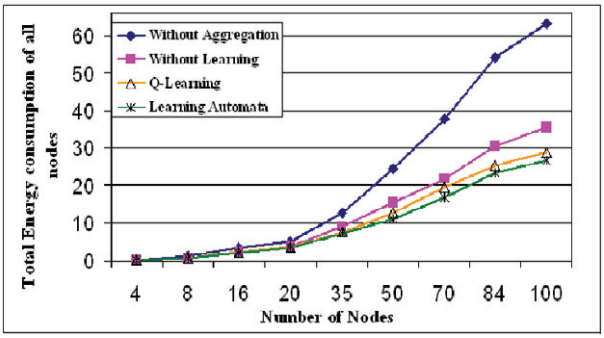

Fig. 2. Comparison among four methods

2.2. A learning Automata Based Power Management

Since the nodes need to be unobtrusive, they have a small form-factor and therefore can carry only a small battery. As a result they have a limited energy supply and power management tool is A must in the design of WSNs. This could be done by designing a robust power control algorithm which prolongs both battery life time and as a result network life time. Clearly, this approach could lead to maintaining network connectivity. By this powerful algorithm, implementing large WSNs could become more reliable, practical and robust [2]. However, according to [2], implementing transmission power management is not a simple task and it contains some considerable risks as follows:

1) As the transmission power is reduced, the communication range is reduced and may risk losing network connectivity.
2) As the communication range is reduced, the number of hops per packet may also increase, and consequently, may increase the system latency and decrease its throughput.
3) As the transmission power is being increased or decreased, more collisions may occur due to incorrect assumptions about the usage of the channel.
In number 2, there could be another possibility, meaning that if the communication range is reduced, the network loses its connectivity but since some nodes are not in the transmission range of others, they are more comfortable to send their data to each other. Because the less network connectivity is, the less collision probability will be, and this could possibly result in network throughput increase. So, we can achieve more throughput in cost of losing network connectivity, which is not desirable at all.
The transmission power management algorithm which is going to deal with this uncertainty cannot be deterministic, because designing such an algorithm is very complicated, if at all possible. So, we need an adaptive solution which uses learning automata in the algorithm. The probabilistic nature of Learning Automata makes them a useful choice for this issue. So, depending on network conditions and nodes density, the learning automaton must control the transmission power level of the node. The proposed method in [2] uses stochastic learning automata (SLA). The term stochastic is because of the dynamic nature of this kind of automaton, meaning not only the probability of selecting each action by the automaton will not be constant and unchangeable, but also will change during the loop shown in Fig.1 in order to find the optimum action which is increasing or decreasing the transmission level of a node.
The operation of the SLA is as follows [2]. At first, according to equation (1), the probability of selecting each action from the actions set of the SLA is equal to 1/r. Since we have only 2 actions here, the probability for both of them is 1/2. So an action, i.e., increasing or decreasing transmission power, is selected randomly.
If the action results in a reward, its probability distribution is increased and the probability distributions of the other actions are decreased. On the other hand if the randomly selected action results in a penalty, its probability distribution is decreased and the probability distributions of the other actions are increased based on Equations 4 and 5. The penalty or reward is assigned based on the number of packet collisions. If the collision level is low, a reward is given, otherwise the action is penalized.
In Fig.3 we can see the effect of not using power control and using SLA based power control, on the number of collisions that may occur at node0.

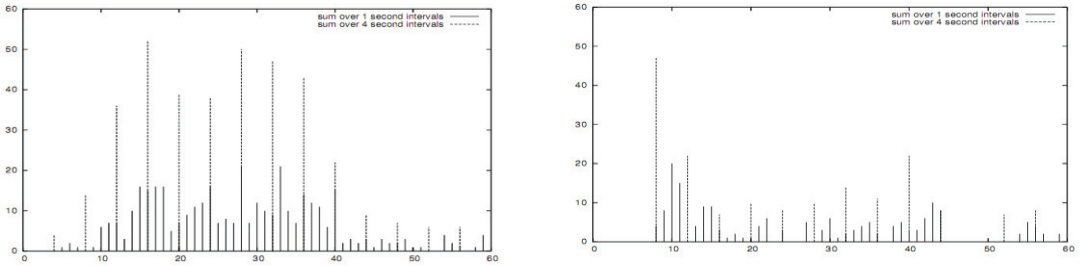

Fig. 3. (a) Number of collisions at node0 without any power control; (b) Number of collisions at node0 with using SLA based power control

As seen in Fig.3, SLA reduces the collision rate which provides more bandwidth in the network.

2.3. Adaptive Learning Solution for Congestion Avoidance in Wireless Sensor Networks

One of the main challenges in WSN is to restrain the congestion in the network traffic. Congestion will result in packet dropping at the intermediate nodes or formation of queues in them which in turn will cause packet delay. The main idea is that if the following equation is true, as a result the occurrence of congestion in the nodes is smoothly avoided.

Packet Service Rate *(Transmitting Rate)* **= Data Packet Arrival** *(Processing Rate)*

The name of proposed algorithm in [3] is Learning Automata-Based Congestion Avoidance Algorithm in Sensor Networks (LACAS) which deals with the congestion problem in WSNs effectively. Generally there are two approaches which algorithms use to deal with congestion in networks; namely congestion avoidance, and congestion

control. LACAS uses congestion avoidance method and by using learning automata which are placed in all of the nodes, attempts to reduce the probability of congestion occurrence in intermediate nodes, while it saves network resources during network life time.

The LA in LACAS algorithm are capable of controlling the rate of flow of data at the intermediate nodes based on probabilistically how many packets are likely to get dropped if a particular flow rate is maintained. These automata learn during the loop shown in Fig.1 and based on the past behavior they choose a better flow rate which is likely to avoid congestion occurrence in the network. It has been seen in previous approaches that congestion avoidance in intermediate nodes is met. Here is their procedure: When the packet flow of sending node is more than the packet flow of the receiving node, congestion is going to happen at the receiving node. Thus the receiver tells the sender to control its sending flow rate to avoid congestion in the receiver. In LACAS this feedback is sent to the receiver by the receiver itself, meaning the intermediate nodes control their own data flow rate according to the output of the learning automaton stationed in them. This output tells the node which flow rate should be chosen. Compared to other methods, by omitting the redundant feedback to the sender, LACAS saves the precious resources of the network, like residual energy at the nodes and the link capacities. The most optimum action (data flow rate) selected by the automaton in a node is completely dependent on the number of dropped packets in that node. In other words, the automaton recommends using a data flow rate by which the number of dropped packets is minimum. This rate will be penalized or rewarded by the environment and by this, the learning automaton updates its probability vector. According to the new probability vector the automaton selects the next action with the highest probability in the list, and this action will be penalized or rewarded by the environment. This loop continues until the most optimal action is selected. Say the chosen optimal data flow rate is $\mathbf{x_i}$, then $Lim\ p(\boldsymbol{x_i})=1$ when $i\rightarrow\infty$. This means that the probability of the most desirable action tends to unity when i leans to infinity. So, after choosing the flow rate $\mathbf{x_i}$, the node transmits its data with x_i flow rate. Since this is the most optimal action selected by the automaton, by choosing this rate, both the probability of the node to get congested and the number of dropped packets in that node are reduced. According to [3] the simulation result for energy consumption in all nodes is presented in Fig.4. This diagram compares energy consumption among 3 different modes: 1) congestion mode 2) normal mode and 3) using LACAS.

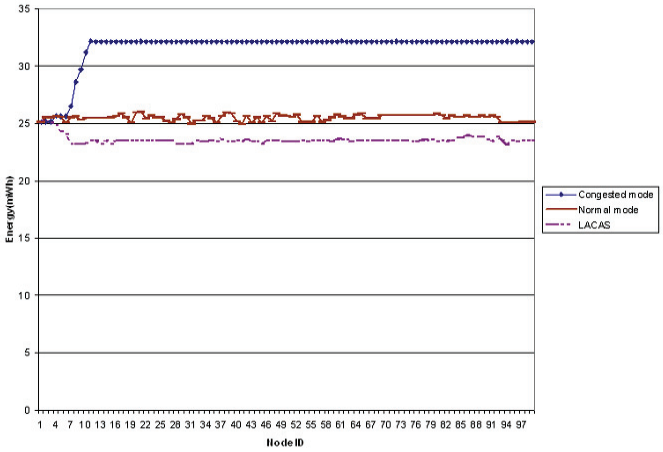

Fig. 4. Energy consumption comparison among 3 modes

It is clear that LACAS outperforms other methods in terms of energy consumption by controlling the flow rate.

2.4. EEMLA: Energy Efficient Monitoring of Wireless Sensor Networks with Learning Automata

The main goal of deploying sensor nodes in a field is to monitor the field of interest and report the occurrence of subject of interest (temperature, humidity, quake,…) to the base station. Usually the sensors are redundantly deployed in order to manage the unexpected failures in the network. There are some applications like environment monitoring which need the sensors to be deployed in high density. Redundancy means that some parts of the field are covered by more than one sensor at the same time, so that if one sensor dies the area is still covered by the others. This redundancy makes the network almost invulnerable to probable risks. The questions that need to be answered are these: Should all the nodes be active at any given time for a monitoring task? Is it possible to dynamically calculate the minimum number of active nodes needed to cover the maximum area of the environment

at any given time? The answers are “definitely not” and “yes”, respectively. This is important, because if all the nodes stay in active mode in whole network life time, the residual energy in the nodes descends rapidly, thus network life time decreases. So, the main goal is to monitor the maximum area possible at any given time with the minimum number of active nodes possible, to reach the maximum network life time possible. In [4] an algorithm named EEMLA (Energy Efficient Monitoring of wireless Sensor Network with Learning Automata) is proposed. In this scheme each node is equipped with a learning automaton which has exactly 2 actions in its action set, namely ON and OFF. This learning automaton learns during time, when the most proper time to switch the node to ACTIVE mode (by selecting ON action) or ASLEEP mode (by selecting OFF action) is. Using this scheme, the network energy resource is used efficiently. The EEMLA scheme works as follows:
Since at the beginning of the EEMLA algorithm the automaton has no knowledge about the network, it sets equal probability 0.5 for selecting each of its actions. This means that the first selection of either action must be randomly, but in next rounds, based on the information gained previously from the environment, the automaton changes these probabilities. In EEMLA, according to [4], it is assumed that the nodes are location-aware so that they can specify the areas they can cover. Then, all of the nodes broadcast the information of their respective covered areas attached by their ID to their local neighbors. Now it is time for every automaton residing in each node to select one of the actions, then they must broadcast their selected actions to their neighbors. After this, not only all of the nodes know the exact covered areas by each of their neighbors, but also all of them know the selected action in their respective neighbors. After gaining these valuable information, **node(i)** performs the following algorithm :

*If the selected action of **LAi** was **ACTIVE** then:*
1. If all of the regions under the coverage of the node(i) are covered by those neighbors whose selected actions are ACTIVE then node(i) penalizes its selected ACTIVE action.
2. Otherwise, node(i) rewards ACTIVE action.

*Else if the selected action of **LAi** was **ASLEEP** then:*
1. If all of the regions under the coverage of the node(i) are covered by those neighbors whose selected actions are ACTIVE then node(i) rewards its ASLEEP action.
2. Otherwise, node(i) penalizes the selected ASLEEP action.

In [4] there have been introduced some other monitoring algorithms like LUC-I, LUC-P and TIAN. In Fig.5 there is a comparison among these protocols and the proposed EEMLA in terms of number of active nodes.

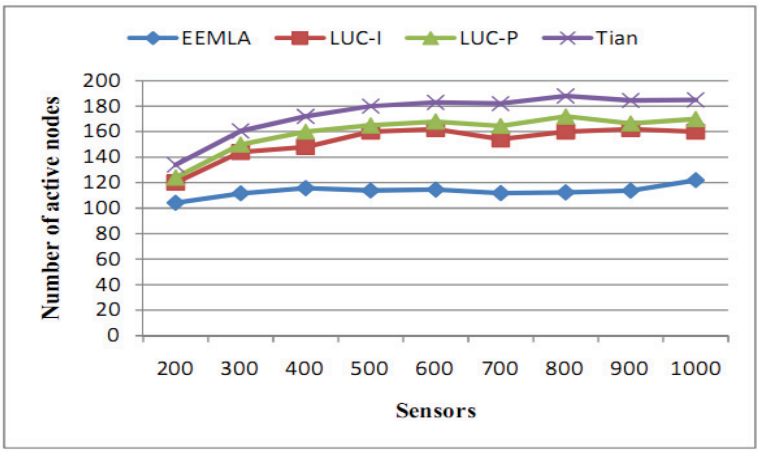

Fig. 5. The comparison among four methods in terms of number of active nodes

As Fig.5 implies, the number of active nodes in EEMLA is always smaller than the other proposed algorithms. This is all because of the learning automata which help the nodes to realize their redundant neighbors in their sensing range and by this, the minimum number of active nodes for covering the maximal area of the entire field is determined.

2.5. ICLEAR:Energy Aware Routing Protocol for WSN Using Irregular Cellular Learning Automata

ICLEAR has two phases: First the BS sends a neighbor packet which has 3 fields named node Id, energy level(always pertaining to packet sender) and hop count to its neighbors, and sets hop count to 1. Each of these neighbors upon receiving this packet builds their routing table. For example suppose that nodei receives a neighbor packet from nodej. It checks the packet and puts an entry pertaining to nodej in its routing table, and then assigns it with a probability that is calculated as follows:

$$p(j) = \frac{Energy\ Level}{\sum_{i=1}^{all} Energy\ Level} \qquad (6)$$

Energy Level j is the energy level of the j^{th} neighbour node, and "all" is the total number of neighbour nodes. p(j) is the probability of choosing nodej as the next hop for transmitting data to the sink. After calculating the p(j), nodei increments the hop count by 1 and then forwards its packet to its immediate neighbour. After building the routing table, the second phase starts. When a node senses an event, it generates an AGENT packet and then selects the neighbour with the highest probability from its routing table, and sends the AGENT packet to it; this packet travels the network till it finds the BS. Each node that receives the AGENT packet creates an ACK packet with two fields namely Node Id and Energy Level and sends it to the sender node. When the sender receives the ACK, it compares the Energy Level in the packet to the Energy Level in its routing table. If the value read from the packet is higher than a threshold named φ, this action is rewarded; otherwise it is penalized by the learning automaton residing in the sender node.

This ACK helps the LA learn to choose the optimal neighbor at any given time. The performance results of ICLEAR completely depend on the value of φ. As Fig.6 shows, the amount of energy consumed differs depending on φ. Energy Consumption in ICLEAR with a=b=0.1 for different values of φ in random network with 20 events is shown in Fig.6 where **a** and **b** are reward and penalty parameters, respectively.

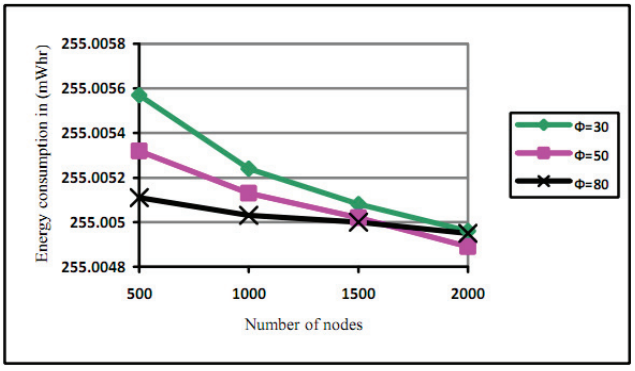

Fig .6. Energy Consumption in ICLEAR with a=b=0.1 for different values of φ in random network with 20 events

3. Conclusion

Wireless sensor networks are increasingly being used in military, environmental, health and commercial applications. These networks suffer from severe resource constraint like limited battery supply, QOS needs and energy efficient routing issues. Any algorithm, which is going to be proposed to solve a specific problem in the field of WSN, must consider the dynamic environment, which surrounds the network. This means that a deterministic and static solution could not work efficiently when we are dealing with dynamic environments, because when an action

seems to be the best in a moment, could be the worst if the environment changes. In this paper we have tried to explore some Learning Automata-based algorithms which have the ability to deal with the changes of environment. The surveyed approaches in this paper have covered QOS needs, the limited network life time issues and energy efficient routing problems in the case of WSNs.

References

1. M. Esnaashari, M. R. Meybodi,"A Learning Automata Based Data Aggregation Method for Sensor Networks", 12th International CSI Computer Conference, Tehran, Iran, 20-22 February 2007.
2. El-Osery, A.I. Baird, D. Abd-Almageed, W,"A Learning Automata Based Power Management for Ad-Hoc Networks", International Conference on Systems, Man and Cybernetics,Waikoloa, Hawaii island, 10-12 Oct. 2005
3. Misra, S. Tiwari, V. Obaidat, M.S. ,"Adaptive Learning Solution for Congestion Avoidance in Wireless Sensor Networks", International Conference on Computer Systems and Applications, Rabat, Morocco, 10-13 May 2009
4. Mostafaei, H. Meybodi, M.R. Esnaashari, M,"EEMLA: Energy Efficient Monitoring of wireless Sensor Network with Learning Automata", International Conference on Signal Acquisition and Processing, Bangalore, India, February 09-February 2010.
5. Navid, A.H.F, Javadi, H.H.S, "ICLEAR: Energy Aware Routing Protocol for WSN Using Irregular Cellular Learning Automata", IEEE Symposium on Industrial Electronics and Applications, Kuala Lumpur, Malaysia, October 4-6, 2009.