A new Scheduling Mechanism Inspired of Artificial Immune System Algorithm for Wireless Sensor Networks

Arash Nikdel[1], Amir Massoud Bidgoli[2], Mohammad Hossein Yektaie[1]

[1]*Department of Computer, Science and Research Branch, Islamic Azad University, Khuzestan, Iran*

[2]*Department of Computer Engineering, Islamic Azad University, Tehran North Branch, MIEEE, Ph.D. Manchester University, Tehran, Iran*

a.nikdel@khouzestan.srbiau.ac.ir, drbidgoli@gmail.com, mh.yektaie@gmail.com

Abstract

In hierarchical routing in Wireless Sensor Networks, the nodes are divided into clusters. In each cluster, one node is selected as cluster head and other nodes are cluster members. Till now, different mechanisms have been proposed for communication between cluster members and cluster head in hierarchical protocols. Most of them determine a same time for each cluster member to communicate with cluster head without regarding the nodes conditions. In this paper, we propose a novel scheduling mechanism inspired of Artificial Immune System algorithm called CHSM. In this mechanism, the nodes with more information have a better chance for communicating with cluster head. Then, we propose QoS-CHSM mechanism and improve the proper nodes distribution in each cluster. In this method, we try to apply a more proper distribution of nodes in the cluster, by changing cluster to the virtual sub clusters and applying the CHSM for each virtual cluster separately. In fact, the CHSM is a special manner of QoS-CHSM in which the number of virtual clusters equals one. We have simulated LEACH protocol and used proposed scheduling mechanisms in it and then compared it with original LEACH protocol which uses TDMA scheduling mechanism. The results of simulation show the effectiveness of the proposed mechanisms.

Keywords: *Wireless Sensor Network, Scheduling Mechanism, LEACH Protocol, Artificial Immune System algorithm*

1. Introduction

Recent technological advances have led to the emergence of small, low-power devices that integrate sensors with limited processing and wireless communication capabilities [1]. The wireless sensor network (WSN) has emerged as a promising tool for monitoring the physical world and has a wide variety of potential applications in many fields [2] [3]. Sensors can be deployed rapidly and cheaply, thereby enabling large-scale, on-demands monitoring and tracking [2]. Wireless sensor networks open a wide range of applications, including environment monitor, disaster prediction, military surveillance and vehicle tracking [1] [2]. Different applications have different requirements, so they adopt the different network models and design assumptions, such as: network structure, sensor deployment strategy, sensing model, transmission range, failure model, time synchronization, and location information [1]. Furthermore, applications differ in their requirements; therefore the underlying sensor networks usually have different design objectives or have different priorities among the

design objectives [1]. Designing protocols and algorithms for WSNs is more challenging due to limited resources, lack of centralized control, unreliable wireless channel conditions, and various application-specific demands [4].

One of the important issues on the field of sensor networks is routing. In [5], the routing protocols have been classified in four general categories such as Data-centric, hierarchical, location-awar and Quality of Service (QoS) Aware. In hierarchical routing, the nodes are divided into clusters. In each cluster, one node is considered as cluster head and other nodes are cluster members. Cluster members receive information from the environment and then send them to the cluster head. After that, the cluster head send the information to the sink node. Most of the hierarchical routing protocols consist of two steps. The first step is selecting the cluster head and the second one is routing. Hierarchical routing is a useful method in order to decrease the number of messages sending to the base stations and also to increase the lifetime of the network.

Different methods are used in hierarchical protocols in order to avoid interference between the data of cluster members. The most used method is Time-Division Multiple Access (TDMA) that is used in many protocols [6] [7] [8] [9] [10] [11] [12] [13] [14]. TDMA method doesn't distinguishes between cluster member with low data and cluster member with high data and gives both of them the same chance to communicate with the cluster head. While a member of cluster maybe located in a dynamic area and should have more time to send information to it in comparison with another members.

In this paper, we propose a novel scheduling mechanism inspired of Artificial Immune System called Case History Scheduling Mechanism (CHSM) for communication between a cluster members and cluster head. The remaining of this paper is organized as follow: original LEACH protocol is explained in section 2. Artificial Immune System algorithm as a basic learning strategy used in the proposed protocol is discussed in section 3. Proposed mechanism is explained in section 4. Improved version of our mechanism is presented in section 5. Simulation results are shown in section 6. Section 7 is the conclusion.

2. Leach Protocol

LEACH [9] is an important well known hierarchical routing protocol. The operation of LEACH is broken up into rounds, where each round begins with a set-up phase, when the clusters are organized, followed by a steady-state phase, when data transfers to the base station occur. Initially, each node chooses a random number between 0 and 1. If the number is less than a threshold which is obtained by (1), the node becomes a cluster head for the current round. Cluster head for the current round broadcasts an advertisement message to the rest of the nodes. The non-cluster head nodes must keep their receivers on during this phase of set-up to hear the advertisements of all the cluster head nodes and decides the cluster to which it will belong for this round. The cluster-head node creates a TDMA schedule telling each node when it can transmit and broadcast back to the nodes in the cluster. Once the clusters are created and the TDMA schedules fixed, data transmission can begin.

$$\begin{aligned} &\textit{If } n \in G \\ &\qquad T(n) = P / (1 - P \times (r \bmod (1/P))) \\ &\textit{Otherwise} \\ &\qquad T(n) = 0 \end{aligned} \tag{1}$$

The time slices are assigned to cluster members based on TDMA mechanism. At each time slice, cluster head communicates with one of its members and receives the information from it. After some time slices, the cluster head sends the received information to the sink

node. For distributing the load among different nodes, at the end of each round the cluster heads are changed based on mentioned mechanism. The round and time slice in LEACH protocol is shown in Fig. 1.

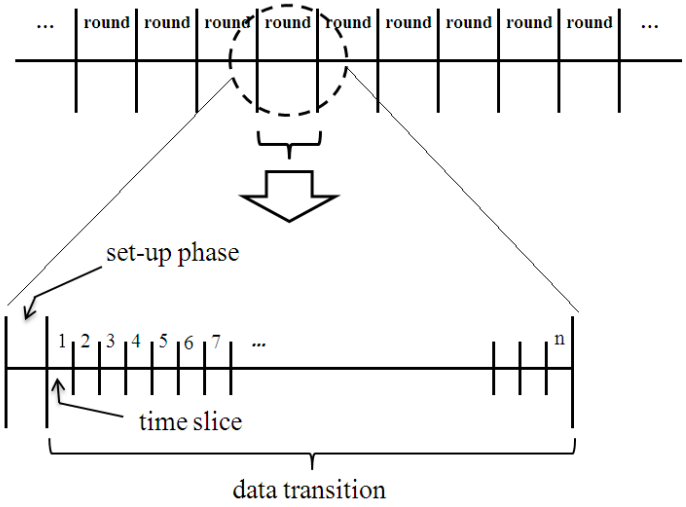


Figure 1. The Round and Time Slice in LEACH Protocol

3. Artificial Immune System Models

Artificial Immune System (AIS) are distributed adaptive systems for problem solving using models and principles derived from the Human Immune System. The Immune System is the defense system of our body, which can produce and secrete antibodies used to protect us against infection through an antigen recognition process. The capabilities of the Natural Immune System (NIS) is mainly the inner working and cooperation between the mature T-Cells and B-Cells that is responsible for the secretion of antibodies as an immune response to antigens. The various theories with regards to the functioning and organizational behavior of the NIS, is discussed. These theories inspired the modeling of the NIS into the AIS for application in non-biological environments. Many different AIS algorithm models have been built, including bone marrow models, thymus models, Clonal selection algorithms, Positive Selection, and immune network models [15] [16]. The pseudo code for the basic AIS algorithm includes eight steps as follows [16]:

```
Initialize a set of ALCs as population C;
Determine the antigen patterns as training set D_T;
While some stopping condition(s) not true do
    For each antigen pattern z_P ∈ D_T do
        Select a subset of ALCs for exposure to z_P , as population S ⊆ C;
        For each ALC x_i∈S do
            Calculate the antigen affinity between z_P and x_i;
        End for
        Select a subset of ALCs with the highest calculated antigen affinity as population H ⊆ S;
        Adapt the ALCs in H with some selection method, based on the calculated antigen affinity and/or the
                                                                    network affinity among ALCs in H;
        Update the stimulation level of each ALC in H;
    End for
End while
```

Each of the algorithm's parts is briefly explained next:

1. *Initializing C and determining D_T:* The population C can either be populated with randomly generated ALCs or with ALCs that are initialized with a cross section of the data set to be learned.
2. *Stopping condition for the while-loop:* In most of the discussed AIS models, the stopping condition is based on convergence of the ALC population or a preset number of iterations.
3. *Selecting a subset, S, of ALCs:* The selected subset S can be the entire set P or a number of randomly selected ALCs from P.
4. *Calculating the antigen affinity:* The antigen affinity is the measurement of similarity or dissimilarity between an ALC and an antigen pattern. The most commonly used measures of affinity in existing AIS models are the Euclidean distance, r-continuous matching rule, hamming distance and cosine similarity.
5. *Selecting a subset, H, of ALCs:* In some of the AIS models, the selection of highest affinity ALCs is based on a preset affinity threshold. Thus, the selected subset H can be the entire set S, depending on the preset affinity threshold.
6. *Calculating the network affinity:* This is the measurement of affinity between two ALCs. The different measures of network affinity are the same as those for antigen affinity. A preset network affinity threshold determines whether two or more ALCs are linked to form a network.
7. *Adapting the ALCs in subset H:* Adaptation of ALCs can be seen as the maturation process of the ALC, supervised or unsupervised. Some of the selection methods that can be used are negative selection (or positive selection), clonal selection and/or some evolutionary technique with mutation operators. ALCs that form a network can influence each other to adapt to an antigen.
8. *Updating the stimulation of an ALC:* The stimulation level is calculated in different ways in existing AIS models. In some AIS models, the stimulation level is seen as the summation of antigen affinities, which determines the resource level of an ALC. The stimulation level can also be used to determine a selection of ALCs as the memory set. The memory set contains the ALCs that most frequently match an antigen pattern, thus memory status is given to these ALCs.

Artificial immune systems have been successfully applied to many problem domains. Some of these domains range from network intrusion and anomaly detection, data classification models, virus detection, concept learning, data clustering, robotics, pattern recognition and data mining [16].

4. Proposed Scheduling Mechanism

In this section, we describe the proposed scheduling mechanism called CHSM. This mechanism is employed after set-up phase in order that cluster heads collect information from their members. In this mechanism, each cluster head selects the members to communicate with them using AIS algorithm. In the proposed mechanism, we suppose that if one cluster member has a packer to send at the moment *t*, it will have another one to send very likely at the moment *t+1* and vice versa. For this reason, if the selected member has data to send, it will be more probable to select it in the next time slices and otherwise, selection probability

will be decreased. The proposed algorithm keeps the *B* bit history of communications with each node. We present this history by sequence of bits and called it case history. In case history, 0 means that node doesn't have any data to send and 1 means that node has data to send. At first, after performing the set-up phase, all case histories are considered equal to 0 as shown in Fig. 2.

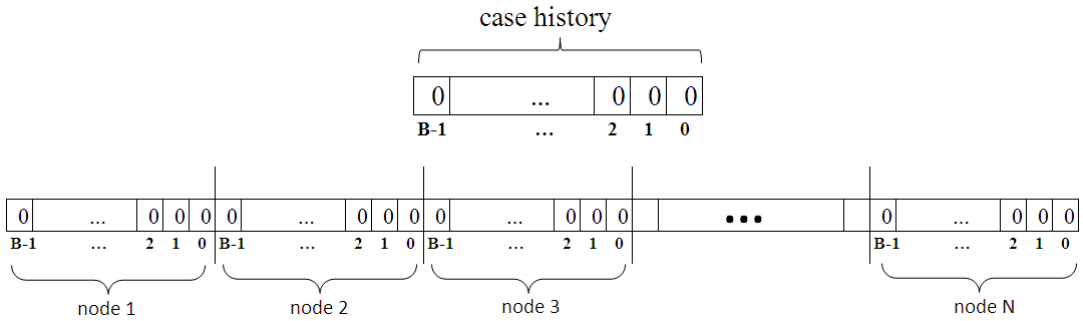


Figure 2. Cluster head keeps the B bit history of communications with each node. At first, all case histories are considered equal to 0.

Whenever the cluster head communicates with a node, it updates the case history of that node. For this purpose, all bits are shifted to the right and the least significant bit is removed and the value of the most significant bit is calculated by f_r function. The amount of function f_r is calculated according to (2). This process is shown in Fig. 3.

If the node doesn't have any data to send
$f_r = 0$
Else (2)
$f_r = 1$

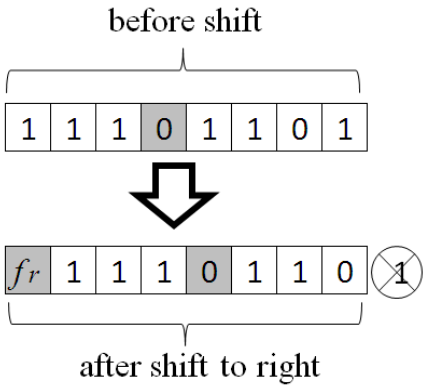


Figure 3. Process of updating the case history

In each time slice of the proposed algorithm, the cluster head selects one of the cluster members. The probability of selecting each node is obtained by (3):

$$P_i = \text{affinity}_{\text{node}} / \Sigma_{j=1 \text{ to } N} \text{affinity}_{\text{node-j}} \quad (3)$$

In (3), the affinity value of node is calculated using case history of node according to (4):

$$\text{affinity}_{\text{node}} = \lambda + \Sigma_{b=0 \text{ to } B-1} (2^{b-1} a^{b-1}) \quad (4)$$

In (4), it is supposed that the node case history is B bit (a_{B-1}, … , a_2 ,a_1,a_0). Also λ is a constant number in order that the node affinity value does not exceed a determined level. In order that the affinity of a node with the most affinity value in maximum case becomes ζ times more than the node with minimum affinity value, the parameter λ is determined according to (5) to (8) as follows:

$$\text{affinity}_{\text{minimum}} = \lambda \quad (5)$$

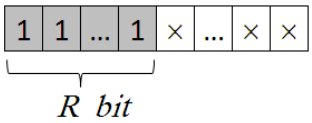
$$\text{affinity}_{\text{maximum}} = \lambda + 2^{B-1} \quad (6)$$

$$\zeta = \text{affinity}_{\text{maximum}} / \text{affinity}_{\text{minimum}} = (\lambda + 2^{B-1}) / \lambda \quad (7)$$

$$\lambda = (2^{B-1}) / (\zeta - 1) \quad (8)$$

After selecting one node, the cluster head requests the selected member to send data packet. If the cluster member has packet to send, it will send it to the cluster head and otherwise, it will not respond. Then, the cluster head regarding that node has had data to send or not, updates the case history of mentioned node. The process of updating the case history was discussed before. In this way, the probability of selecting node in future becomes more or less.

In the proposed mechanism, the nodes with affinity value more than threshold will be selected as memory cells and have some quota. It means that after the specified number of time slices (η), the cluster head will communicate with them surely. In fact, they are given a chance to resend data. The affinity threshold is calculated according to (9):



$$\text{affinity}_{\text{threshold}} = \lambda + \Sigma_{b=R \text{ to } B-1} (2^{b-1}) \quad (9)$$

In order to give quota to the nodes selected as memory cells, the cluster head in some constant time rounds (after η time slice), determines μ time slice to communicate with the nodes selected as memory cells. The amount of μ is calculated by (10):

$$\mu = |M| \quad (10)$$

Where |M| represents the number of nodes selected as memory cells and μ is the number of time slices determined to communicate with the nodes selected as memory cells.

The only operator in AIS algorithm is mutation. We suppose that the mutation rate of each node (τ_{node}) has reverse impact on its affinity value ($\text{affinity}_{\text{node}}$). τ_{node} is calculated for each node according to (11):

$$\tau_{node} = \psi / (\text{affinity}_{\text{node}} + \varepsilon) \quad (11)$$

Where ψ is a constant number that is calculated in the way that mutation value doesn't become less than the predetermined value. Also ε is a constant number that should be selected properly in order that mutation value doesn't exceed the predetermined value.

In the proposed mechanism, the cluster head after some time slices (σ), selects some of nodes randomly (∂ percent of nodes) and mutates the case history of nodes regarding their mutation rate. Fig. 4 shows three binary numbers with four different mutation rates:

Regarding the binary values before and after mutation, we observe that this operation makes smaller number bigger and big number smaller. When we have more mutation rate, then the change will be greater. Note that the nodes with less affinity have more mutation and vice versa.

In the proposed mechanism, the only way to give more chance to those nodes which had not any data to send in the past time slices and with low affinity value, is mutation. While it is possible that such nodes aren't given any chance to communicate with cluster head in some successive time slices.

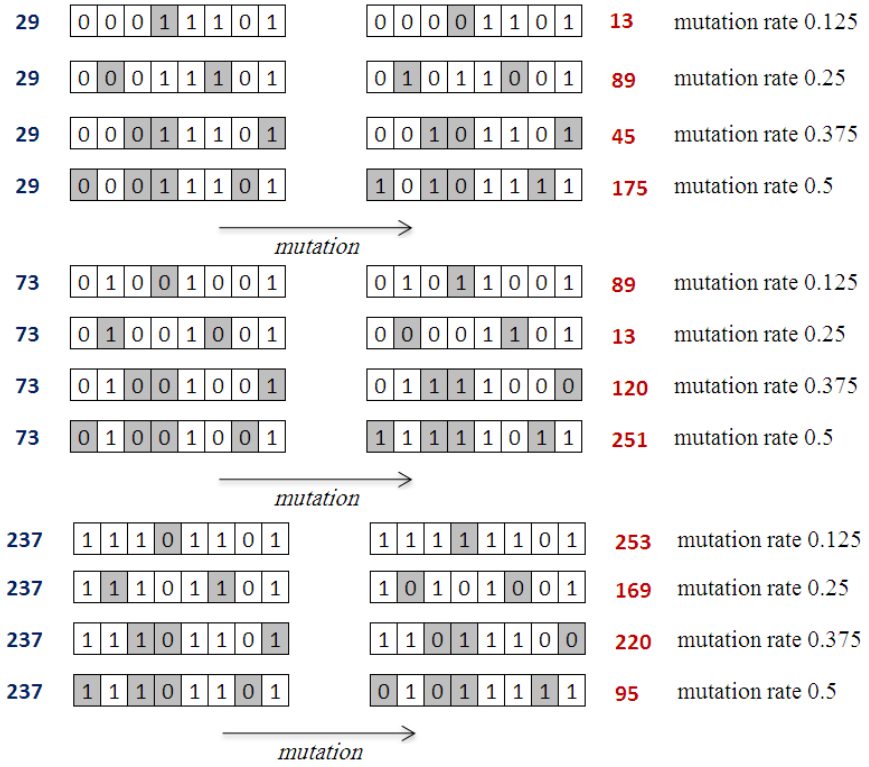
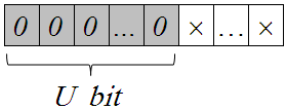


Figure 4. Three binary numbers with four different mutation rates

In order to overcome this challenge, after each μ+η time slice, the δ number of time slices spent on communicating with α percent of nodes that their affinity are less than the affinity $_{\text{threshold-bellow}}$. The value of affinity threshold-bellow is calculated according to (12):



$$\text{affinity}_{\text{threshold-below}} = \lambda + \Sigma_{b=0 \text{ to } B-U} \left(2^{b-1} \right) \qquad (12)$$

Also the value of δ is calculated according to (13). Note that η and μ time slices is for communicating with member nodes and also with the nodes selected as memory cells.

$$\delta = \alpha \times |C| \quad (13)$$

Where δ represents the number of time slices are allocated to communicate with such nodes, |C| represents number of nodes with affinity value less than affinity $_{\text{threshold-below}}$. Also, α is percent of nodes which are communicated and are given chance to send data. It is obvious that α will be between 0 and 1.The cluster head repeats these tasks after *n* time slices. The value of *n* is calculated According to (14):

$$\eta + \mu + \delta = n \quad (14)$$

This sequence continues until the new round of clustering. After the number of time slices, the cluster head sends received information of its cluster members to the sink node. These repetitions are shown in Fig. 5. The pseudo code for the CHSM algorithm includes six steps as follows:

```
// 1.initialize parameter
Initialize parameter
For each node n_x do
   Case-history_node-x='00000000'

// 2.main loop of CHSM mechanism
Count =0
Repeat:
   // 3.the cluster head communicates with a node; Note that probability of selecting each node is proportionate with its affinity
   For i = 1 to μ do
     Select a random node as n_x
     Send a request date packet for node n_x
     If node n_x have data then f_r = 1
     Else f_r = 0 //node n_x not have any data
     Shift to right Case-history of node n_x with f_r bit
     Count++
   End For

   //4.Communicating with the member nodes in memory cell set
   M = φ //initializing the memory cell
   For each node n_x do
     Calculate affinity of node n_x
     If affinity node-x > affinity threshold then
        Add the node n_x to the memory cell set, M
     End if
   η=|M|
   For i= 1 to η do
      n_x = i^th node in memory set, M
      Send a request date packet from node n_x
      If node n_x have data then f_r = 1
      Else f_r = 0 // node n_x not have any data
      Shift to right Case-history of node n_x with f_r bit
      Count++
   End for

  //5.Communicating with α percent of members with low affinity
   C= all node that their affinity is less than affinity threshold-below
```

```
δ=α*|C|
For i= 1 to δ do
   n_x = select one node in memory set C randomly
   Send a request date packet from node n_x
   If node n_x have data then f_r = 1
   Else f_r = 0 //node nx not have any data
   Shift to right Case-history of node n_x with f_r bit
   Count++
End//for

//6.mutation process
For each node n_x do
   Calculate the mutation rate τ for node n_x
   Mutate Case-history of node n_x with τ rate
End For

While (Count <= n)
```

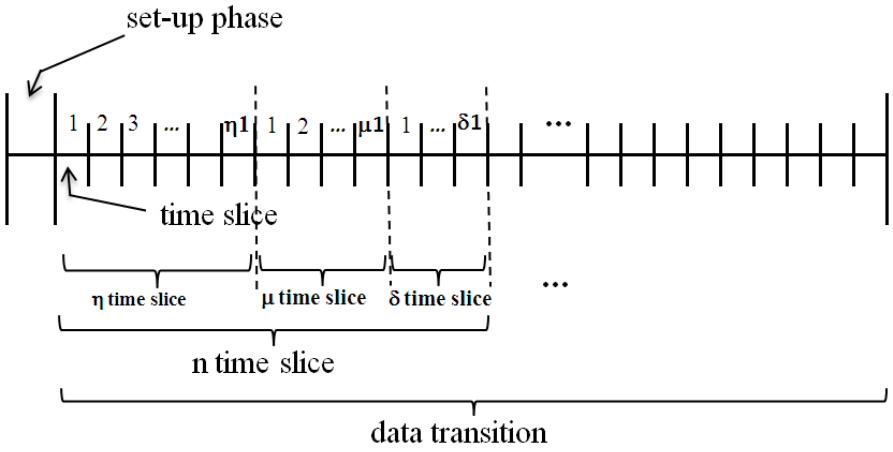


Figure 5. The repetition of cluster head tasks

5. Improving the proposed mechanism using virtual clustering concept

In this section, we describe an improved CHSM mechanism, called QoS-aware Case History Scheduling Mechanism (QoS-CHSM). In the CHSM mechanism, there is no attention to the proper distribution of the selected nodes in the cluster. The nodes are selected only based on their affinity value. In order to improve the distribution of the selected nodes in cluster area, we divide the cluster to the several sub clusters (virtual cluster) and apply the CHSM mechanism to each sub cluster separately. We assume that the nodes are static once deployed, and each one knows its own location which can be achieved by using some location system [17].

After performing the set-up phase, the cluster head sends the special packet ("please-register" packet) to all nodes. This packet consists of geographic location of cluster head and also the number of virtual clusters. After receiving this message, each member calculates its virtual cluster number based on (15) to (18):

$$\theta = \mathrm{Sin}^{-1}\left((x - x_{node}) / d\right) \qquad (15)$$

$$d_{node} = ((x - x_{node})^2 + (y - y_{node})^2)^{1/2} \quad (16)$$

$$\Omega = 2\Pi / r \quad (17)$$

$$VirtualCluster_{node} = \lceil \theta / \Omega \rceil \quad (18)$$

Where (x_{node}, y_{node}) is coordinates of the member node, (x, y) is coordinates of the cluster head, r is the number of sub clusters (virtual clusters), $VirtualCluster_{node}$ is sub cluster number of member node. In Fig. 8, one cluster is divided to six virtual clusters, so r =6. For example, if we suppose that θ=150, Ω=60, then the member node will be the member of third sub cluster (i.e. $VirtualCluster_{node}$=3). This process is shown in Fig. 6.

At the end, cluster head communicates with each of their members separately and collects the "registerMe" packets from their members. This packet contains ID and Virtual Cluster number of member node.

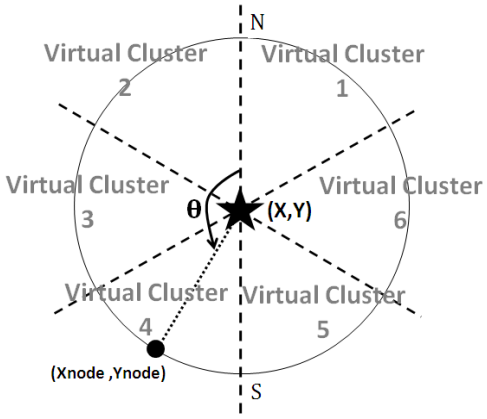


Figure 6. Dividing a cluster to the virtual sub clusters

After receiving all "registerMe" packets, the cluster head will be informed of the number of all nodes in cluster and the number of nodes in each virtual cluster. After determining the number of member nodes in each virtual cluster, the cluster head applies the CHSM algorithm to each virtual cluster separately and determines the number of necessary time slices in order to guarantee the quality of service. The cluster head calculates parameters for each virtual cluster separately according to (19) to (26):

$$N_1 + N_2 + ... + N_r = \sum_{i=1 \text{ to } r} (N_i) = N \quad (19)$$

$$\eta_1 + \eta_2 + ... + \eta_r = \sum_{i=1 \text{ to } r} (\eta_i) = \eta \quad (20)$$

$$\eta_i = (N_i / N) / \eta \quad (21)$$

$$\mu_i = \lceil M_i \rceil \quad (22)$$

$$\mu_1 + \mu_2 + \ldots + \mu_r = \sum_{i=1 \text{ to } r} (\mu_i) = \mu = |M| \quad (23)$$

$$C_1 + C_2 + \ldots + C_r = \sum_{i=1 \text{ to } r} (C_i) = C \quad (24)$$

$$\delta_1 + \delta_2 + \ldots + \delta_r = \sum_{i=1 \text{ to } r} (\delta_i) = \delta \quad (25)$$

$$\alpha_i = (C_i / C) / \alpha \quad (26)$$

Where r represents the number of virtual clusters, N represents the number of all nodes and N_i is the number of member nodes in the i^{th} virtual cluster. η shows the number of all time slices allocated before giving quotas and η_i represents the number of time slices allocated to communicate for the i^{th} virtual cluster. μ is the number of all nodes selected as memory cells, μ_i is the number of member nodes in the i^{th} virtual cluster selected as memory cells. M represents the number of member nodes in memory cells set and M_i is number of nodes belonging to the i^{th} virtual cluster which also are in the memory cells set. Fig.7 shows the time slices of this mechanism.

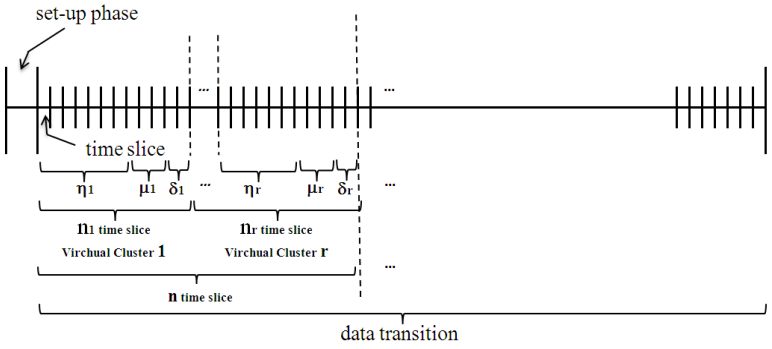


Figure 7. Using the virtual clustering to improve the node's distribution in CHSM

The virtual clustering effect on improving the quality of service is shown in Fig.8.

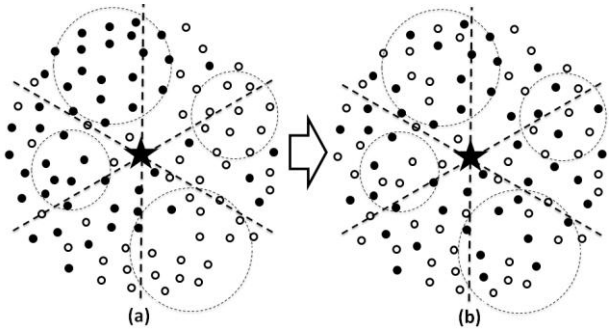


Figure 8. The virtual clustering effect on improving the quality of service

In Fig. 8.a , it is shown that the cluster head communicates with half of nodes. But due to improper distribution of nodes at different locations, performing the CHSM method will not get the favorite result and both unchecked and checked are observed at some points. But this problem is solved by applying QoS-CHSM method as shown in Fig. 8.b.

6. Simulation Results

In this section, our proposed mechanism is simulated using NS2 simulator [18]. We simulated LEACH protocol and used proposed scheduling mechanism in it ($LEACH_{CHSM}$ and $LEACH_{QoS\text{-}CHSM}$) and then compared it with original LEACH protocol which uses TDMA scheduling mechanism ($LEACH_{TDMA}$). In order to perform simulation, 100 sensor nodes are distributed in an area equal to 100×100 square meters. The sink node is selected randomly among 100 nodes. The nodes take the temperature of environment and inform the cluster head of its changes [19]. The number of virtual clusters in our simulation is considered equal to 4 (r = 4). Other assumptions are as follow: B=8, η=50, α=0.05, λ=128, ζ=3, ψ=128, σ=50 and ∂=50%. Each node that is selected as a cluster head will be responsible in 500 time slices. The cluster head puts the received data in a packet after 5 time slices and sends it to the sink node. The network is tested for loads (the average number of packets sent by each node in each time slice) 0.0125, 0.0250, 0.0375, 0.0500, 0.1000, 0.1500, 0.2000, 0.2500 and 0.3000. The simulation duration is 600 seconds and during simulation, the clustering step is performed several times.

Table 1. The Parameters Values for Simulation

The assumptions									
Parameter	B	η	α	λ	ζ	ψ	σ	∂	*r*
value	8	50	0.05	128	3	128	50	50%	4

6.1 The First Experiment

In this experiment, we evaluate the performance of proposed mechanism. For this purpose, LEACH protocol with P=0.077 is employed using two scheduling mechanism. After clustering step based on LEACH protocol, the cluster heads begin to collect the data packets from their own cluster members using CHSM, QoS-CHSM or TDMA mechanism. In this experiment, load parameter (i.e. the average number of packet sent by a node in each time slice) is considered 0.1.

The results of experiment are shown in Table 2 and Fig. 9. As can be seen in Table 2 and Fig. 9, the average of received data for CHSM and QoS-CHSM is greater than TDMA mechanism. As mentioned before, each node that is selected as a cluster head will be responsible in 500 time slices. In proposed method, after some time, the cluster head communicates with cluster members in more time slice and so, cluster head knows its members conditions better than before. Thus cluster head communicates with a member which has more information to send. Therefore, it receives more data packets from its members. It means that the CHSM method determines some members of cluster with more sending information. Thus, cluster head give them more chance to communicate with cluster heads. While in TDMA mechanism, the cluster heads do not distinguish between their members.

Table 2. The average of received data

Method	Time slice									
	10	**20**	**30**	**50**	**70**	**100**	**200**	**300**	**400**	**500**
LEACH with TDMA	0.28	0.33	0.37	0.42	0.40	0.43	0.44	0.42	0.46	0.43
LEACH with CHSM	0.27	0.35	0.47	0.50	0.53	0.60	0.62	0.68	0.72	0.73
LEACH with QoS-CHSM (r=4)	0.30	0.37	0.45	0.45	0.50	0.56	0.56	0.62	0.60	0.63

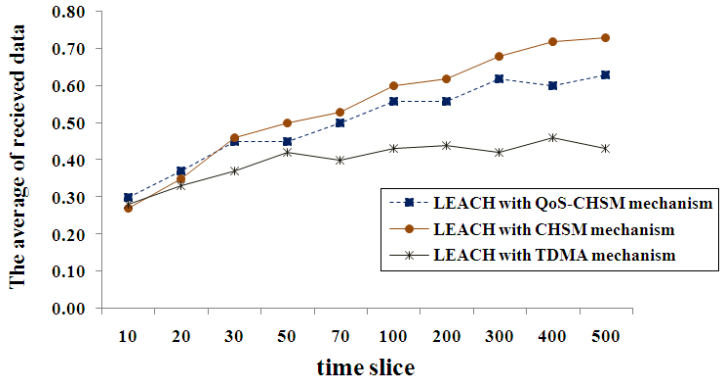


Figure 9. Average data received in different time slices

6.2 The Second Experiment

In this experiment, the average of received data for different traffic loads with P=0.077 are evaluated. Fig. 10 and Table 3 show the results of this experiment. As can be seen in Fig. 10, the average of received data in each time slice in proposed mechanisms is higher than TDMA. This is because in proposed mechanisms, the cluster head knows its active members and so, these members are allocated more time to send information.

Table 3. The average of received data

Method	Average traffic load								
	0.0125	**0.0250**	**0.0375**	**0.0500**	**0.1000**	**0.1500**	**0.2000**	**0.2500**	**0.3000**
LEACH with TDMA	0.12	0.21	0.32	0.35	0.48	0.55	0.57	0.73	0.83
LEACH with CHSM	0.15	0.33	0.42	0.60	0.70	0.77	0.80	0.90	0.92
LEACH with QoS-CHSM (r = 4)	0.14	0.36	0.41	0.60	0.62	0.70	0.69	0.87	0.85

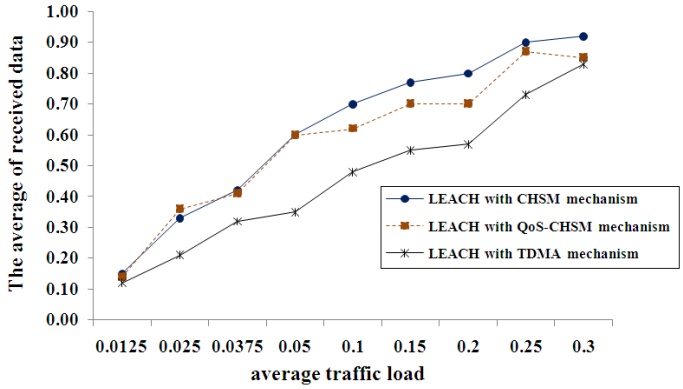


Figure 10. Average of received data in different traffic loads

6.3 The Third Experiment

The purpose of this experiment is to evaluate the average time that each package waits in queue. For this purpose, CHSM, QoS-CHSM and TDMA mechanisms with P=0.077 for different loads are employed and the average of waiting time in queue for each package is calculated. Fig. 11 and Table 4 show the results of this experiment.

Table 4. The average waiting time in queue for each packet in different traffic loads

Method	Average traffic load								
	0.0125	**0.0250**	**0.0375**	**0.0500**	**0.1000**	**0.1500**	**0.2000**	**0.2500**	**0.3000**
LEACH with TDMA	0.33	0.46	0.45	0.50	0.51	0.53	0.60	0.63	0.65
LEACH with CHSM	0.25	0.31	0.32	0.40	0.38	0.38	0.40	0.41	0.41
LEACH with QoS-CHSM	0.21	0.30	0.33	0.40	0.43	0.45	0.49	0.46	0.51

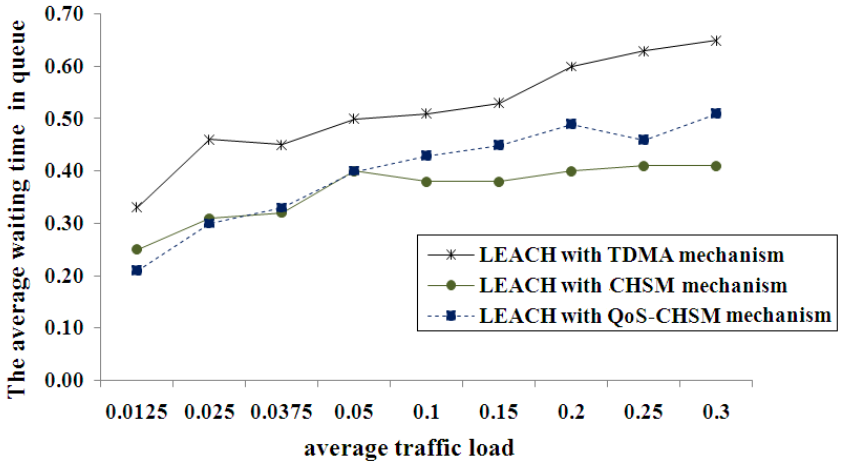


Figure 11. The average waiting time for each packet in different traffic loads

The experiment results show that the performance of CHSM (and QoS-CHSM) is better in comparison with TDMA. Regarding that in CHSM, the active cluster members have more time to send information, they can send their data at a higher rate, therefore, the average of waiting time in queue for their packets is decreased.

5. Conclusions

In this paper, we proposed a new scheduling mechanism inspired of Artificial Immune System algorithm for communication between cluster heads and their members. Then, we improved the QoS in CHSM and proposed QoS-CHSM scheduling mechanism. In order to evaluate our mechanism, we simulated LEACH protocol and used proposed scheduling mechanisms in it ($LEACH_{CHSM}$ and $LEACH_{QoS\text{-}CHSM}$) and then compared it with original LEACH protocol which uses TDMA scheduling mechanism ($LEACH_{TDMA}$). The results of simulation show the effectiveness of the proposed mechanism.

References

[1] X. Li, Y. Mao, and Y. Liang, "A Survey on Topology Control in Wireless Sensor Networks". In: 10th Intl. Conf. on Control, Automation, Robotics and Vision, ICARCV, Hanoi, Vietnam, 17–20 December 2008.

[2] J. Jia, J. Chena, G. Chang, and Z. Tan, "Energy efficient coverage control in wireless sensor networks based on multiobjective genetic algorithm", in: Computers and Mathematics with Applications journal (Elsevier) , 2009, pp. 1756_1766.

[3] X. Zhang, X. Ding, S. Lu, and G. Chen, "Principles for Energy-Efficient Topology Control in Wireless Sensor Networks", WiCom '09. 5th International Conference on Wireless Communications, Networking and Mobile Computing, IEEE, 24-26 Sept 2009.

[4] H. J. Choe, P. Ghosh, and S. K. Das, "QoS-aware data reporting control in cluster-based wireless sensor networks", in: Computer Communications journal (Elsevier), 2010, Published by Elsevier B.V.

[5] K. Akkaya, and M. Younis,"A survey on routing protocols for wireless sensor networks", in: Ad Hoc Network Journal (Elsevier), 2005, pp. 325-349.

[6] M. Younis, M. Youssef, and K. Arisha, "Energy-Aware management in cluster-based sensor Networks", in: The International Journal on Computer Networks, Vol. 43, No. 5, December 2003, pp. 649-668.

[7] I. S. Misra, S. Dolui, and A. Das, "Enhanced-Efficient adaptive clustering protocol for distributed sensor networks", ICON 2005.

[8] T. Voig, A. Dunkels, J. Alonso, H. Ritter, and J. Schiller, "Solaraware clustering in wireless sensor networks", in: Computers and Communications, ISCC 2004, Ninth International Symposium, Vol. 1, 28 June-1 July 2004.

[9] T. T. Huynh, and C. S. Hong, "Prolonging network lifetime via intracluster routing in wireless sensor networks", in: Proceedings of ICMU2005, April. 2005, pp. 162-167.

[10] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energyefficient communication protocol for wireless microsensor network", in: Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS '00), January 2000.

[11] R. Musunuri, and J. A. Cobb, "Hierarchical-battery aware routing in wireless sensor networks", in: Vehicular Technology Conference, 2005.

[12] A. Manjeshwar, and D. Agrawal, "TEEN: A routing protocol for enhanced efficiency in wireless sensor networks", in: 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, 2001.

[13] A. Manjeshwar, and D. Argrawal, "Apteen: A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks". IPDPS, 2002.

[14] M. Younis, M. Youssef, and K. Arisha, "Energy-Aware routing in cluster-based sensor networks", in: Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002), Fort Worth, Texas, October 2002.

[15] A. M. Bidgoli , A. K. Javanmardi, and A. M. Rahmani, "Application of AIS algorithm for optimization of TORA protocol in ad hoc network", in: WORLDCOMP, The International Conference on Wireless Networks (ICWN 2010), Las Vegas, USA, 2010.

[16] A. P. Engelbrecht, "Handbook of Computational Intelligence, An Introduction", wiley, second edition 2007, pages 426-428 and 448.

[17] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low-cost outdoor localization for very small devices", in: IEEE Personal Communications 7 (2000), pp. 28_34.

[18] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[19] D. Janakiram, R. Venkateswarlu, and S. Nitin, "A survey on programming languages, middleware and applications in wireless sensor networks", IITM-CSE-DOS-2005-04, 2005.

Authors

Arash Nikdel received the B.S. degree in Computer Engineering from the Islamic Azad University and the M.S. degrees in Computer Engineering from Ahvaz Science and Research Branch in Iran, in 2007 and 2011, respectively. His research interests include computer networks and evolutionary algorithms.

Amir Massoud Bidgoli is a professor of Computer Engineering in the Department of Computer Engineering, Islamic Azad University, Tehran North Branch. He received his B.Sc. degree in Electronics Engineering from Lancaster University (UK) and the M.Sc. degree in Computer & Control Engineering from Salford University (UK), in 1987 and 1989, respectively. He received his Ph.D. in Computer Science from Salford & Manchester University (UK), in 1996. He is a member of IEEE and is indexed in IEEE Explorer. His research interests include computer networks & evolutionary algorithms, computer architecture, networking, distributed systems and image processing. He has over fifty publications in journals and ISI conferences.

Mohammad Hossein Yektaie has been graduated in BSc of Mathematics and its application in computer science from Isphahan University, Isphahan, Iran and Master Degree in Software Engineering from La Rochelle University, La Rochelle , France. He has been obtained his Doctorate or PH.D. in Computer Science from La Rochelle University. He has taught several courses graduate and undergraduate in the field of computer engineering and IT. He has published a lot of articles about his work. His research interests are AI, Pattern recognition, Cloud Computing and NLP. Recently he is working on a Machine translation program.