A Learning Automata Based Data Aggregation Method for Sensor Networks

M. Esnaashari[1], M. R. Meybodi[2]

[1,2] Soft Computing Laboratory, Computer Engineering and Information Technology Department
Amirkabir University of Technology, Tehran, Iran
(Esnaashari,mmeybodi)@aut.ac.ir

Abstract

One way to reduce energy consumption in wireless sensor networks is to reduce the number of packets being transmitted in the network. Data aggregation technique, in which packets with related information are combined together in intermediate nodes before further forwarding towards their destination, shown to be a good solution for this purpose. It is clear that more related information along the paths each packet traverse results in higher aggregation ratio. If the information each node has, varies from time to time, finding paths along which more nodes with related information exist becomes a complicated task. In this paper, a learning automata based data aggregation method in sensor networks will be proposed. Each node of the sensor network is equipped with a learning automaton which learns the path along which the aggregation ratio is maximum. Simulation results show that the proposed method performs well in situations where nodes having related information vary during the network's lifetime.

Keywords

Sensor Networks, Data Aggregation, Learning Automata

1. Introduction

Sensor networks are composed of several low–cost micro devices with limited power, computation and memory resources. Each node has the ability to sense a specific phenomenon, such as temperature, humidity, pressure, etc. and send its readings for its neighbors. In other words, two most important abilities of a sensor node are sensing and communicating with other nodes. In some cases, these nodes may be connected to each other via a wired network, but most of the time, a sensor network is completely wireless. In these networks, nodes commonly have few or no movements at all.

Sensor nodes are very prone to failure due to their limited resources and abilities. One major reason for this failure is exhausting energy resources and for this reason energy is a vital factor in sensor networks. Usually a central node called sink is the destination of all packets in a sensor network. In some applications, all nodes have the ability to communicate directly with the sink, but this leads to a limited network dimensionality as well as consuming more energy power due to wider range of communication. Therefore, in most scenarios, nodes communicate with the sink through their neighbors; hence they need to know which neighbor can forward their packets to the sink better.

Until now, many routing algorithms have been reported for sensor networks. Ilyas and Mahgoub in [17] give a survey on some of these algorithms such as MCFA[1], LEACH[2], PEGASIS[3], TEEN[4], GAF[5] and SPIN[6]. Some other methods such as EAR[7], Rumor, CADR[8] and ACQUIRE[9] are surveyed in [18] by Akkaya and Younis. In a number of these algorithms, each node may have more than one alternative path to the sink, one of which is selected according to a certain criterion. Different criteria such as distance to the sink, traffic load and consumed energy along the path, can be used for selecting a specific path. As energy is a vital resource, consumed energy along a path would be a good criterion for this purpose. It can be used in two different ways. One way is to compute energy consumption for each path separately, and then choose the path with the least energy consumption [16]. One problem with such a method is that, it needs a way of computing energy consumption along each path. Another way for forwarding data packets is to make use of data aggregation technique. In data aggregation technique, packets with related information are combined together in intermediate nodes to form a single packet before further forwarding to the sink. Using this method, one can select a path along which more related packets can be found to be aggregated with its packet. This way, number of packets transmitted throughout the network will be reduced; hence less energy would be consumed. So, in this method, a path with highest data aggregation ratio is used.

If the nodes with related information don't vary throughout the lifetime of the network, the problem of maximizing aggregation ratio is simple. Each node selects a neighbor with more related information to that of itself as it is done in [12]. On the other hand, if each node's information changes in time, maximizing aggregation ratio becomes a complicated task. In such

dynamic situations, nodes should adapt themselves to the changing information of themselves and their neighbors in order to find a path toward the sink along which aggregation ratio is maximum.

In this paper a new method based on learning automata for data aggregation has been proposed. In the proposed method, each node in the network is equipped with one learning automaton. These learning automata collectively learn the best path of aggregation for each node for transmitting its packets towards the sink. In other words, the learning automaton in each node forces the node to send its data via a path along which aggregation ratio is maximum

To evaluate the performance of the proposed method several experiments have been conducted and the results are compared with the results of three different methods: *i*) without aggregation, *ii*) method given in [12] which performs data aggregation with no learning, and *iii*) method given in [14] which performs data aggregation using Q-learning. The results show that the proposed method outperforms all these methods, especially when the environment is highly dynamic.

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

2. Related Works

Some of the methods reported recently for data aggregation in sensor networks such as [1], [2] and [3], are query based methods. In these methods a query is generated at the sink and then is broadcasted through the network. Each node, upon receiving this query, processes it partially, and then passes it on to its neighbors. After the query is processed completely, its result will be sent back to the sink. In this scenario, some nodes just process the query, and some others, propagate it, receive partial results, aggregate the results, and send them back to the sink.

In some other works, such as [4], [5], [6], [7] and [8], the network is first clustered, and then cluster heads are used for aggregating data packets in each cluster separately. Lotfinezhad, and Liang in [9] try to investigate the effect of partially correlated data on the performance of the clustering methods for data aggregation. They have shown that partially correlated data has a strong effect on the clustering performance, meaning that the lower is the data correlation; the lower is the clustering performance. They have also shown that the amount of energy consumed in a single node is strongly related to its position in the network; a node far from the sink may consume more energy than other nodes. They have also shown that the network lifetime is inversely proportional to total consumed energy.

Some other methods are also reported for data aggregation in sensor networks. Guestrin et. al. in [10] try to approximate the data generated in a single node for a specific period of time with a third order equation, and send the coefficients of this equation to the sink instead of data itself. Dasgupta et. al. in [11] assume that each node in the network has the ability to aggregate data. Based on this assumption, they present a method for maximizing network lifetime. They define the network lifetime as the time elapsed before the first node dies due to energy exhaustion. this method tries to find an aggregation tree rooted at the sink spanning all the sensors and maximizes the lifetime of the network. For this purpose, they first find all possible aggregation trees and then, make use of a heuristic approach to find the one which can maximize the lifetime of the network.

Beaver, and Sharaf in [12] propose an algorithm which tries to find paths along which, sensors belonging to the same group (i.e. sensing the same phenomenon) reside. In this algorithm, sink will initiate a "path construction" packet through the network. Each node, upon receiving this packet for the first time, set its parent to the sender of the packet, and then forwards the packet to its neighbors. If a node receives "path construction" again, it checks whether sender of the packet belongs to the same group as itself or not. If so, this node, changes its parent to this new sender of "path construction". This way, each path to the sink will consist of mostly the nodes belonging to the same group and hence aggregation ratio along the path will be increased. This method performs well in situations where nodes do not change their groups during the lifetime of the network.

Very few efforts in data aggregation make use of learning. Radivojak et. al. in [13] uses machine learning technique to force sensor nodes to send only specific data required by the sink. Learning algorithm used in this method is executed in the sink and its results is propagated throughout the network. Beyens et. al. in [14] uses a Q-learner in each node forcing the node to send its data via a path along which aggregation ratio is maximum

3. Learning Automata

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

Environment can be defined by the triple $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ represents a finite input set, $\beta = \{\beta_1, \beta_2, ..., \beta_r\}$ represents the output set, and $c = \{c_1, c_2, ..., c_r\}$ is a set of penalty probabilities, where each element c_i of c corresponds to one input of action α_i. An environment in which β can take only binary values 0 or 1 is referred to as P-model environment. A further generalization of the environment allows finite output sets with more than two elements that take values in the interval [0, 1]. Such an environment is

referred to as Q-model. Finally, when the output of the environment is a continuous random variable which assumes values in the interval [0, 1], it is referred to as an S-model. Learning automata are classified into fixed-structure stochastic, and variable-structure stochastic. In the following, we consider only variable-structure automata.

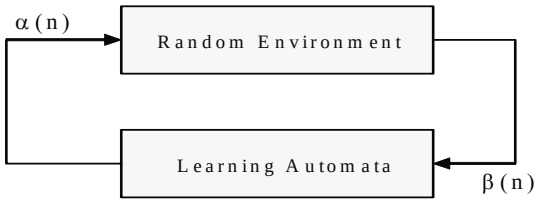

Fig. 1. Relationship between learning automata and its environment

A variable-structure automaton is defined by the quadruple $\{\alpha, \beta, p, T\}$ in which $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_r\}$ represents the action set of the automata, $\beta = \{\beta_1, \beta_2, \cdots, \beta_r\}$ represents the input set, $p = \{p_1, p_2, \cdots, p_r\}$ represents the action probability set, and finally $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ represents the learning algorithm. This automaton operates as follows. Based on the action probability set *p*, automaton randomly selects an action α_i, and performs it on the environment. After receiving the environment's reinforcement signal, automaton updates its action probability set based on equations (1) for favorable responses, and equations (2) for unfavorable ones.

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

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

In these two equations, *a* and *b* are reward and penalty parameters respectively. for *a* = *b*, learning algorithm is called L_{R-P} [10], for *a* << *b*, it is called $L_{R\varepsilon P}$ [11], and for *b* = 0, it is called L_{R-I} [12].

4. Problem Statement

The problem is to find the best path for each node in a sensor network for directing its packets towards the sink. The best path is defined to be the one along which the highest aggregation ratio can be obtained. The assumption that all nodes want to send their data to a sink node is valid in most sensor networks, as these networks are usually used for controlling some parameters such as temperature, humidity, pressure, etc. in a large environment and each node needs to send its data to a control center.

To specify the problem more clearly, without loss of generality, we make use of a specific application in which nodes have the ability to sense the temperature. These nodes are scattered randomly in a large environment, each of them senses the temperature of its surrounding area and sends its readings back to the sink. We assume that several climates exist in the environment each having a different temperature, and moving along a specific direction with a specific velocity. Also we assume that the velocity and direction of movement of each climate may be changed randomly in time. To further simplify the problem, we assume the climates to be of spherical shape. Figure 2 depicts such an environment with 18 sensor nodes, and 5 climates.

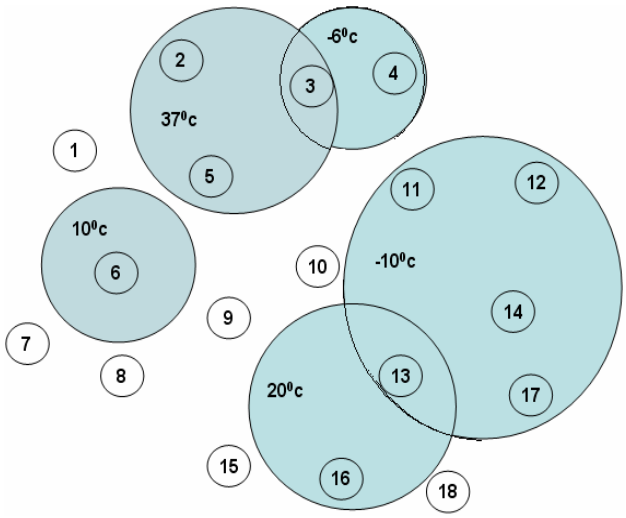

Fig. 2. A schematic of the simulated environment with 18 sensor nodes, and 5 different climates

Each sensor node reports the temperature of the climate in which it resides. If it happens for a node to reside in more than one climate, its sensor will sense the temperature equal to the mean temperature of all climates it resides in. Nodes which reside in no climate, will report 0 as their surrounding area temperature. For example, nodes 2 and 5 in figure 2 will report 37, sensor 4 will report -6, and sensor 3 will report mean of 37 and -6 which is 15.5. In addition nodes 1, 7, 8,9,10, 15 and 18 will report 0 as their reading. Data of nodes which have close reading to each other can be aggregated. For instance, all of the nodes 11, 12, 14 and 17 in figure 2 have their reading as -10, so instead of sending 4 different data packets to the sink, it is suffice to send just a single packet.

As mentioned earlier, having direct communication between nodes and the sink is not a wise assumption because of two reasons. The first reason is that in large networks, some nodes may reside in areas which are too far from the sink, and direct communication is completely impossible. The second reason is that even when direct communication is possible, it is better to make use of intermediate nodes for directing packets towards the sink, because energy consumption for sending data to a far destination would be more than that of a near one. So, we assume that nodes would make use of their neighbors to send data packets to the sink. Upon receiving a packet by a node, it checks to see if it can aggregate its data with

the content of the received packet. This is possible whenever reported temperature in the packet is close enough to the temperature read by the node itself. Aggregation would be performed by averaging on the node's data and content of the received packet.

5. The Proposed Method

In this section we first describe the routing algorithm used and then present the proposed learning automata based data aggregation method, and finally two improvements to the proposed method will be discussed.

5.1. Routing Algorithm

The routing algorithm used is based on the simple flooding method. Pseudo code for this algorithm is given in Figure 3. Sink node initiates a "Path Construction" packet using procedure "MakePathConstructionPacket", and broadcasts it over the network. This packet contains a parameter called "DistanceToSink" which is set to 1 at the sink node. Other nodes wait to receive the "Path Construction" using "ReceivePacket" procedure. Upon receiving this packet, three different cases may be arisen:

(a) A "Path Construction" packet with lower "DistanceToSink" than that of current packet has been previously received at this node. In this case, the newly received packet is ignored using "Drop" method.

(b) The previously received "Path Construction" packets have equal "DistanceToSink" with that of current packet. In this case, sender of the "Path Construction" is added to the routing table as a new alternative path to the sink using "AddRoute" method, "DistanceToSink" is increased by 1, and the packet is broadcast over the network again.

(C) The previously received "Path Construction" packets have greater "DistanceToSink" than that of current packet. This means that a path with fewer numbers of hops to the sink is found. So, all the previously found paths are removed from the routing table using "ClearAllRouts" method. Then sender of the currently received "Path Construction" is added as a path to the sink, "DistanceToSink" is increased by 1, and finally the packet is broadcast over the network again.

The algorithm continues until all nodes in the network find their possible paths to the sink, all having the same number of hops. In addition, for each path, only storing the next hop in the routing table will be sufficient.

5.2. Learning Automata Based Data Aggregation Methods

As it was mentioned earlier, at the end of the routing phase, each node finds a list of all its neighbors using which it can forward its packets toward the sink. We use $RoutingList_i$ (RL_i) to refer to this list for node i. $N(RL_i)$ is the number of entries in this list. Each node i in the network is equipped with a learning automaton with $N(RL)_i$ actions whose probability of selecting each is initially set to $1/N(RL)_i$. Each action of the learning automaton of a node is the selection of one of its neighbors using which it may forward its packets toward the sink. The action selected by an automaton will be rewarded or penalized based on the acknowledgment it receives from the selected neighbor in response to the sent packet. This acknowledgment determines to what extent the information in the sent packet was related with that of the selected neighbor.

```
Procedure Routing
   d = Infinity; //Distance to the sink
   if (IsSinkNode)
      P = MakePathConstructionPacket();
      P.DistanceToSink = 1;
      Broadcast(P);
      Return;
   end if
   Do {
      P = ReceivePacket();
      if (P == "PathConstruction")
         if (d < P.DistanceToSink) //Case 1.
            Drop(P);
         if (d == P.DistanceToSink) //case 2.
            AddRoute(P.Forwarder);
            P.DistanceToSink++;
            P.Forwarder = This.ID;
            BroadCast(P);
         end if
         if (d > P.DistanceToSink) //case 3.
            ClearAllRoutes();
            AddRoute(P.Forwarder);
            P.DistanceToSink++;
            P.Forwarder = This.ID;
            BroadCast(P);
         end if
      end if
   } for a specified amount of time
end
```

Fig. 3. Pseudo code for routing algorithm

Figure 4 presents the main loop each node follows in the proposed method. After the routing phase is finished, each node i starts to sense its surrounding environment, and in predefined intervals, sends out its reading to the sink via one of its alternative paths. For selecting a path, node i uses its automaton to select one of its neighbors j. Node j upon receiving a packet from node i checks if it can aggregate data reported in the packet with that of itself. This procedure is described in figure 5. The data reported in each packet is the temperature of surrounding area of the reporter node which is a real number in centigrade degree. Node j then uses equation (3) to compute the data aggregation ratio ($DAR_{j,i}$), using the temperature reported from node i (T_i) and the temperature sensed by itself (T_j).

$$DAR_{j,i} = U\left(\frac{Max(T_i - a, T_j - a) - Min(T_i + a, T_j + a)}{2a}\right) \quad (3)$$

where

$$U(x) = \begin{cases} x & ; x \geq 0 \\ 0 & ; x < 0 \end{cases} \quad (4)$$

In equation (3), a is a parameter which specifies the acceptable range of aggregation. In other words, aggregation can be performed only if $|T_i - T_j| < 2a$.

If $DAR_{j,i}$ is more than a specified threshold *AcceptRate*, the average of T_j and T_i will be sent to the next hop from node *j*, otherwise, node *j* sends two different packets for T_j and T_i separately. In addition, $DAR_{j,i}$ is sent back to node *i* as environment feedback. Based on the received feedback, the automaton of node *i* penalizes or rewards the selected action (action *k*). If this feedback is greater than *AcceptRate*, action *k* is rewarded and if it is less than *AcceptRate*, action *k* is penalized according to equations (5), and equations (6), respectively.

$$p_k(n+1) = p_k(n) + \alpha.(DAR_{j,i}).(1 - p_k(n))$$
$$p_l(n+1) = p_l(n) - \alpha.(DAR_{j,i}).p_l(n) \qquad \forall l \; l \neq k \quad (5)$$

$$p_k(n+1) = (1 - \beta.(1 - DAR_{j,i}))p_k(n)$$
$$p_l(n+1) = \frac{\beta.(1 - DAR_{j,i})}{r-1} + (1 - \beta.(1 - DAR_{j,i}))p_l(n) \qquad \forall l \; l \neq k \quad (6)$$

In these equations, α is the reward rate, and β is the penalty rate.

```
Procedure Main
   Do {
      T = SenseEnvironment();
      NextHop = Automata.SelectAction();
      SendP = MakePacket(T);
      SendPacketTo(SendP, NextHop);
      RecvP = ReceiveAckFrom(NextHop);
      if (RecvP.DAR > AcceptRate)
      //Using equation (5)
         Automata.RewardAction(RecvP.DAR);
      else
      //Using equation (6)
         Automata.PenalizeAction(RecvP.DAR)
   } while simulation time isn't over
end
```

Fig. 4. Main loop each node follows after the routing phase

```
Procedure RecvPacketFrom(Packet P)
   //Using equation (3)
   DAR = ComputeAggregationRatio (P.T, T);
   if (DAR > AcceptRate)
      T = (T+P.T)/2.0;
   AckP = MakeAckPacket(DAR);
   SendPacketTo(AckP, P.Sender);
end
```

Fig 5. The procedure each node follows upon receiving a packet from one of its neighbors

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.

5.3. Improvements

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

5.3.1. Algorithm 1

In this algorithm unlike the basic algorithm in which a node chooses a neighbor which can better aggregate its data, a node chooses a neighbor which has a neighbor with a data for aggregation. In figure 6, node 10 has three neighbors 7, 8 and 9 for sending its packets towards the sink. The basic algorithm chose one of these neighbors at random. But node 7 has node 6 as its neighbor, which can aggregate its data with that of this node, whereas the other two alternatives do not have such neighbors.

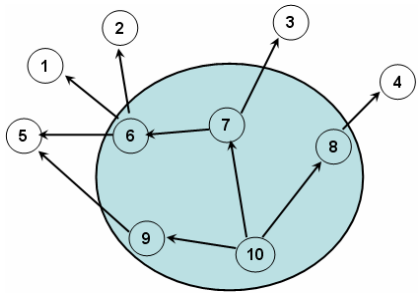

Figure 6. Among nodes 7, 8 and 9, node 10 selects node 7 with higher probability

Algorithm 1 performs the following. Each node *j* upon receiving a packet from its neighbor *i*, first computes $DAR_{j,i}$ using equations (3) and (4), and then makes use of equation (7) for computing *Enhanced-*$DAR_{j,i}$. Once *Enhanced-*$DAR_{j,i}$ is computed, this will be sent back to node *i* as the response of node *j*.

$$Enhanced - DAR_{j,i} = Max(DAR_{j,i}, MRR_j) \quad (7)$$

In this equation MRR_j is for Maximum Received Reward and is defined as the most favorable response received at node *j* from its outgoing neighbors.

Using this equation, in network of figure 6 the computed Enhanced-$DAR_{j,i}$ at node 7 is more than that of nodes 8 and 9. This is because $DAR_{j,i}$ in all three nodes are equal, but MRR at node 7 is more than that in nodes 8 and 9 which results in higher probability of selecting node 7 by node 10.

5.3.2. Algorithm 2

Algorithm 2 is algorithm 1 in which if a node cannot find a neighbor to aggregate its data with that of the node (such as node 6 in figure 6), it chooses a neighbor with more residual energy for forwarding its packet considering the residual energy for neighbor selection in such a situation, a path with both higher aggregation ratio and higher residual energy is produced. That is a more energy safe path can be found. Figure 7 is the pseudo code for algorithm 2. In this algorithm, upon receiving a packet in node *j* from a neighbor *i Enhanced-DAR*, is computed. If this ratio is more than *AcceptRate*, node j sends back a favorable response to the automaton of node i. Otherwise, node j considers its residual energy to compute the response of taking action of the automatan residing in node i; if the residual energy is more than *AcceptRate*, the response is favorable and the response is unfavorable otherwise.

```
Procedure Alg2-RecvPacketFrom (P)
  //Using equation (3)
  Penalizing = false;
  DAR = ComputeAggregationRatio (P.T, T);
  Enhanced-DAR = Max (DAR, MRR);
  if (Enhanced-DAR > AcceptRate)
    T = (T+P.T)/2.0;
  else
    Enhanced-DAR = Max(Enhanced-DAR
                       ,ResidualEnergy/MAX_ENERGY)
    if (Enhanced-DAR < AcceptRate)
      Penalizing = true;
  end
  AckP = MakeAckPacket(Enhanced-DAR,Penalizing);

  SendPacketTo(AckP, P.Sender);
end
```

Fig. 1. Pseudo code of Algorithm2

6. Experimental Results

To evaluate the performance of the proposed method several experiments have been conducted. In the first two experiments, the basic aAlgorithm is compared with three different methods: *i*) without aggregation, *ii*) method given in [12] with no learning, and *iii*) method given in [14] which uses Q-learning. In the third experiment, we study the convergence behavior of learning automata in different nodes of the network. Remaining experiments compare algorithms 1 and 2 with the Basic Algorithm.

All simulations have been implemented using NS2 simulator. Environment is modeled by a number of spherical climates; each has its own temperature, direction and velocity of movement. Nodes of the network are placed regularly on a 2 dimensional grid. This placement is selected for simplicity of tracking the simulation results, but in general, nodes can be placed in any position in the environment. In all experiments, α and β for the proposed algorithm are set to .1, and a (acceptable range of aggregation) is set to 5. Also α and β parameters in Q-learning method are set to .1 and .01 respectively. 12 different climates are used in the environment, parameters of which are provided in table (1). In simulations, if a climate happens to get out of the environment boundary, it changes its direction by 180 degree and comes back to the environment of the network. Time interval between two consecutive sending of data to the sink in each node is equal to 10 minutes, and the overall time of the simulation is 3 hours and 20 minutes (120,000 seconds). Simulations are performed for 4, 8, 16, 20, 35, 50, 70, 84 and 100 nodes. The results are averaged over three runs.

6.1. Experiment 1

In this experiment, we study the total number of received packets at the sink node. Figure 8 depicts the results for the proposed method and three other methods mentioned earlier. This figure shows that the proposed method outperforms methods i, ii, and iii by 62%, 27%, and 9%, respectively.

6.2. Experiment 2

In this experiment, we study the total energy consumed by the network. For estimating the energy consumption of each node, we assume that .05 milliWatt energy will be consumed upon sending or receiving a single packet. Although usually receiving a packet consumes less energy than sending it, but for simplicity we do not make any difference between these two cases. In addition, energy consumption during inactive periods is not taken into account for further simplicity. Figure 9 compares the results obtained for the proposed method with the other 3 methods. The experiments show that the proposed method outperforms all the other three methods in terms of energy consumption.

6.3. Experiment 3

This experiment is conducted to better understand the convergence behavior of learning automata residing in the nodes of the network. For this propose we study the action probabilities of two automata residing in nodes 27 and 42. The automaton for node 27 has neighbors 16, 26 and 36 as shown in figure 10(a). None of these neighbors has related data to that of node 27; and hence no aggregation can be performed and as a result the action probability of none of the actions of automaton of node 27 converges to 1 as it is depicted in figure 11. The automaton for node 42 (figure 10(b)), has three neighbors 31, 32 and 33. For this node neighbor 32 has related data to that of 42 and hence aggregation in this node is possible, and as a result the action probability of the corresponding action converges to 1 as it is depicted in figure 12.

6.4. Experiment 4

In this experiment, the basic algorithm is compared with algorithms 1 and 2 in terms of total energy consumption and total number of packets received at the sink node. As it is shown in figures 13 and 14, using algorithms 1 and 2 leads to higher data aggregation ratio.

7. Conclusion

In this paper we proposed a novel method based on learning automata for data aggregation in wireless sensor networks. Each node in this method is equipped with a learning automaton. This automaton has a number of actions, each corresponds to one of the neighbors of the node through which can forward node's packets to the sink. Each node through its learning automata learns which neighbor is the best for forwarding its packets according to the aggregation ratio criterion. It has been shown through simulations that in situations where nodes having related information vary during the network's lifetime, the proposed method outperforms other similar methods.

Table 1. Climates information: X and Y denote center position of the climate, R denotes its radius, T is its temperature, V stands for its velocity, and D denotes its direction in radian

	1	2	3	4	5	6	7	8	9	10	11	12
X (m)	385	85	460	685	985	1210	310	1210	1585	1585	835	535
Y (m)	85	385	535	385	535	310	910	835	535	835	1135	1285
R (m)	110	110	110	110	110	220	220	110	110	110	110	110
T (Centigrade)	70	-7	14	-14	-40	20	50	-20	-37	47	89	-30
V (m/s)	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001	.001
D (Radian)	.5	.3	1.1	1.0	.6	.3	.5	.5	.2	0.0	.4	.9

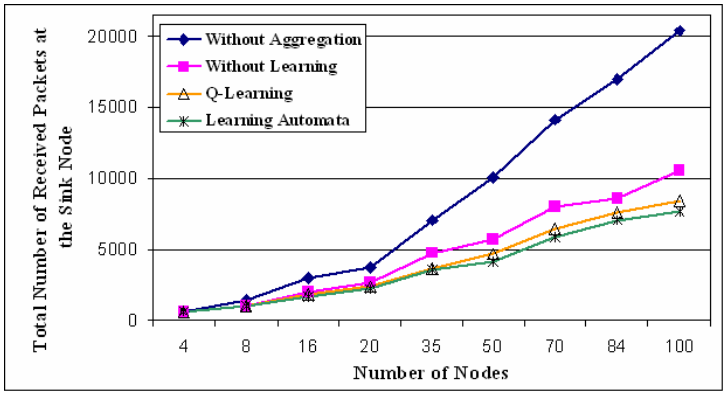

Figure 8. Total number of received packets at the sink node for the proposed method and other methods

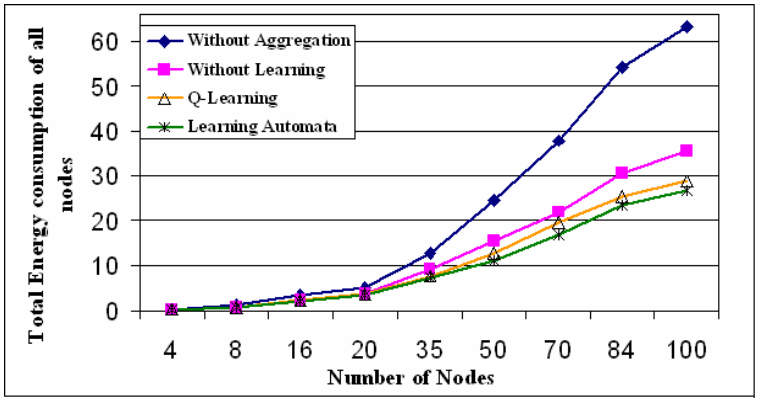

Figure 9. Total energy consumption of all nodes for the proposed method and other methods

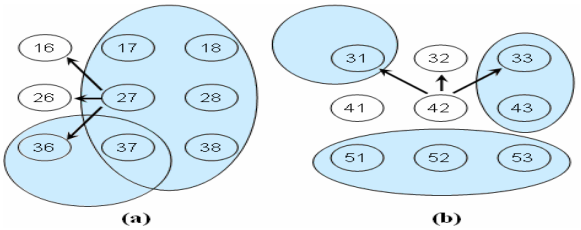

Fig. 2. (a) Environment around node number 27 and its neighbors, (b) Environment around node number 42 and its neighbors

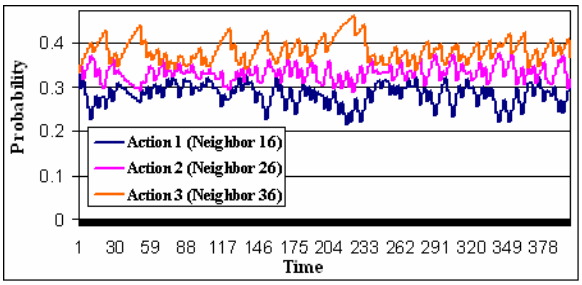

Fig. 3. Action probabilities of the learning automaton in node 27 during the simulation

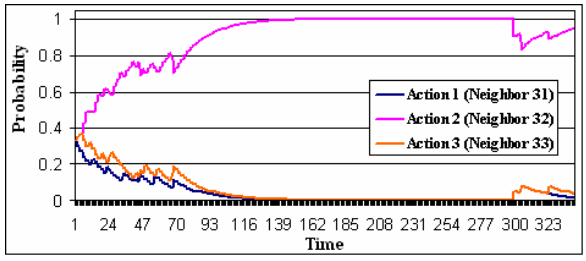

Fig. 4. Action probabilities of automaton in node 42 during the simulation

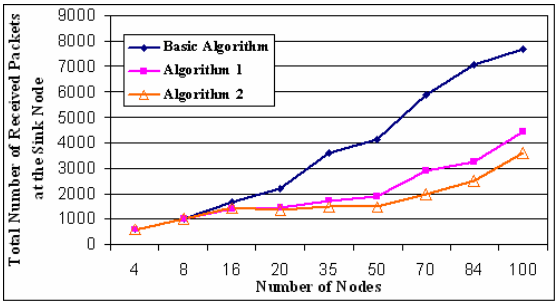

Fig. 5. Total number of received packets at sink node for Basic Algorithm and Algorithm 1

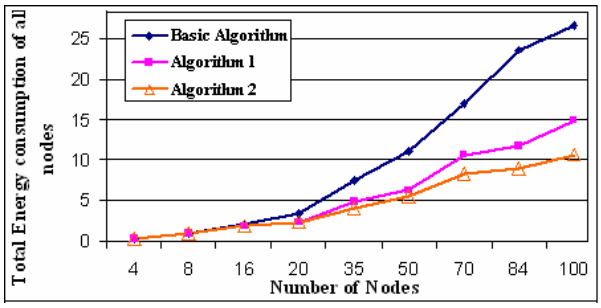

Fig. 6. Total energy consumption of all nodes for the Basic Algorithm and Algorithm 2

References

[1] Y. Xu, W. C. Lee, J. Xu, and G. Mitchell, "Processing Window Queries in Wireless Sensor Networks", *IEEE International Conference on Data Engineering (ICDE'06), Atlanta, GA, April 2006.*

[2] R. Rosemark and W. C. Lee, "Decentralizing Query Processing in Sensor Networks", *the Second International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous'05), San Diego, CA, July, 2005, pp. 270-280.*

[3] J. Winter, Y. Xu, and W. C. Lee, "Energy Efficient Processing of K Nearest Neighbor Queries in Location-aware Sensor Networks", *the Second International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous'05), San Diego, CA, July, 2005, pp. 281-292.*

[4] G. Bontempi and Y. Le Borgne, "An adaptive modular approach to the mining of sensor network data", *Workshop on Data Mining in Sensor Networks, SIAM SDM, Newport Beach, CA, USA, April 2005.*

[5] C. Liu, K. Wu, and J. Pei, "A Dynamic Clustering and Scheduling Approach to Energy Saving in Data Collection from Wireless Sensor Networks", *In Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'05), Santa Clara, California, USA, September, 2005.*

[6] O. Younis and S, Fahmy, "An Experimental Study of Routing and Data Aggregation in Sensor Networks", *In Proceedings of the International Workshop on Localized Communication and Topology Protocols for Ad hoc Networks (LOCAN), held in conjunction with The 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS-2005), November 2005.*

[7] R. Virrankoski and A. Savvides, "TASC : Topology Adaptive Spatial Clustering for Sensor Networks", *Second IEEE Intl. Conf. on Mobile Ad Hoc and Sensor systems", Washington, DC, November, 2005.*

[8] S. Soro and W. Heinzelman, "Prolonging the Lifetime of Wireless Sensor Networks via Unequal Clustering," *Proceedings of the 5th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (IEEE WMAN '05), April 2005.*

[9] M. Lotfinezhad and B. Liang, "Effect of partially correlated data on clustering in wireless sensor networks," *in Proceedings of the IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, California, October 2004.*

[10] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin and S. Madden, "Distributed Regression : An Efficient Framework for Modeling Sensor Network Data", *Intel corporation, 2004.*

[11] K. Dasgupta, K. Kalpakis and P. Namjoshi, "An Efficient Clustering-based Heuristic for Data Gathering and Aggregation in Sensor Networks", *IEEE Wireless Communications and Networking Conference, Vol.4, No. 1, March, 2003.*

[12] J. Beaver, M. A. Sharaf, A. Labrinidis, and P. K. Chrysanthis, "Location-Aware Routing for Data Aggregation in Sensor Networks", *Proc. of the 2nd Hellenic Data Management Symposium, 2003.*

[13] P. Radivojac, U. Korad, K. M. Sivalingam and Z. Obradovic, "Learning from Class-Imbalanced Data in Wireless Sensor Networks," *IEEE Semiannual Vehicular Technology Conference, VTC-Fall 2003*, Vol. 5, pp. 3030-3034, Orlando, Florida, U.S.A., October 2003.

[14] P. Beyens, M. Peeters, K. Steenhaut and A. Nowe, "Routing with Compression in Wireless Sensor Networks: a Q-learning Approach", *In "Fifth European Workshop on Adaptive Agents and Multi-Agent Systems (AAMAS 05), Paris, France.", 2005.*

[15] K. S. Narendra and M. A. L. Thathachar, Learning Automata: An introduction, *Prentice Hall, 1989.*

[16] R. Shah and J. Rabaey, "Energy Aware Routing for Low Energy Ad Hoc Sensor Networks", *in Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*, Orlando, Florida, March 2002.

[17] M. Ilyas, I. Mahgoub, "Handbook of Sensor Networks : Compact Wireless and Wired Sensing Systems", *CRC Press*, London, Washington, D.C., 2005.

[18] K. Akkaya, M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", *Elsevier Ad Hoc Network Journal*, pp. 325-349, 2005.

[1] Minimum Cost Forwarding Algorithm
[2] Low Energy Adaptive Clustering Hierarchy
[3] Power-Efficient Gathering in Sensor Information Systems
[4] Threshold-Sensitive Energy-Efficient Protocol
[5] Geographic Adaptive Fidelity
[6] Sensor Protocols for Information via Negotiation
[7] Energy-Aware Routing
[8] Constrained Anisotropic Diffusion Routing
[9] ACtive QUery forwarding In sensoR nEtworks
[10] Linear Reward-Penalty
[11] Linear Reward epsilon Penalty
[12] Linear Reward Inaction