A Rule Learning Approach to Energy Efficient Clustering in Wireless Sensor Networks

Suan Khai Chong*, Mohamed Medhat Gaber*, Shonali Krishnaswamy*, Seng Wai Loke†

*Caulfield School of IT
900 Dandenong Rd, Monash University, Caulfield Campus, Australia.
†215 Franklin Street, Melbourne, VIC 3000,
Latrobe University, Australia.

Abstract

Physical clustering in wireless sensor networks results in the nomination of 'cluster heads'. The cluster head acts as a hub for the cluster. It is a specific node which has superior energy capabilities when compared with the other members of the same cluster. The nomination of cluster head is performed periodically or iteratively. This process is termed as re-clustering. Reclustering is energy-consuming due to the exchange/broadcast of numerous messages. Thus, this paper uses a rule-learning framework, ARTS (Adaptive Rule Triggers on Sensors), to prolong the intervals betweeen reclustering and thus, reduce the number of messages exchanged. The aim is to conserve the energy of cluster heads by using rules obtained from learning/analysis in clustering processes. To demonstrate, we have used the state-of-the-art clustering protocol HEED (Hybrid Energy Efficient Distributed Clustering) due to its high energy-efficiency in selecting cluster heads. From our experiments using ARTS with HEED, we show that the re-clustering process in any physical clustering algorithm can be performed in a more energy-efficient manner.

I. Introduction

Physical clustering in wireless sensor networks allows data processing load to be balanced among sensor nodes in a network of sensors with similar battery capacities. During the reclustering process in physical clustering, sensor nodes are periodically elected as *cluster heads* to further process data coming from cluster members. The current trend in current clustering algorithms such as LEACH[4] and HEED[9] is that the reclustering process is typically controlled by physical parameters such as residual node energy and node proximity degree. The selection of cluster heads in this manner gives preference to nodes that are favourable to extending sensor lifetime. However, for these clustering algorithms, the reclustering process is energy consuming due to the cost of radio communication among nodes to reelect cluster heads. Our premise is that if cluster heads are used to drive sensing operations within correponding clusters efficiently, the re-clustering frequency would be reduced and thus, overall network lifetime would be increased. This assertation comes from the observation that cluster heads can further analyse data coming from other sensors in their clusters.

Thus, we propose a rule-learning and triggering framework, Adaptive Rule Triggers on Sensors or ARTS for WSNs, to prolong the lifetime of sensor nodes running any physical clustering algorithms such as LEACH or HEED. We have experimentally demonstrated that the ARTS framework, when used in conjunction with HEED, increases the overall network lifetime of sensor nodes deployed. The results show that up to 15% energy is saved with only 0.06% overhead. The experiments also show that more cluster head nodes tend to consume less energy when ARTS is used. These experiments are performed on the tinyOS simulator TOSSIM[6] to obtain results that show savings for greater number of nodes (the tinyOS code for TOSSIM can be directly compiled to work on sensor hardware). The rest of this paper is organised in the following way. In section 2, we present an overview of past and present research in relation to our work. Section 3 details the ARTS framework, broken down into the data model arriving sensor streams, followed by the algorithm used for the components. The modifications to the HEED algorithm are given in section 4. To validate our methods, we run TOSSIM for using HEED vs. HEED with ARTS on sensors and record our observations in section 5. We conclude this paper in section 6.

II. Background

In this section, we detail related work involving data processing on WSNs to develop incremental algorithms to work on resource-constrained sensors. The development of algorithms suited for processing on sensor nodes is important because traditional centralised data mining algorithms are computationally infeasible to be directly implemented on sensors. An example of a traditional rule-mining algorithm that is widely known is APRIORI [1], which is aimed at discovering qualitative rules that describe associations between sets of items.

Several attempts have been made at the forefront of performing data mining on sensor nodes to reduce network data transmission. In [5], while utilising a hybrid system comprising sensor hardware such as mica2 and Stargate as higher resource devices, the authors are experimenting with integer-only FFT algorithm on micas, whereas currently processing is done on Stargate. A more recent study is in [8] where the authors investigated correlations that can be formed when sensors in loading truck experience similar vibrations when the trucks send out the same load. The correlation information of the sensor nodes then allowed them to group trucks carrying out the same load. The unique contribution in their work lies in the incremental calculation of the correlation matrix.

Existing inference-based approaches include [3] in which probabilistic models are used to predict missing values and identify outliers in traditional database system. Nevertheless, current association rule mining techniques have not been demonstrated to work in WSNs. The main challenge of using conventional rule learning algorithm on sensors is the generation of k-itemsets that is computationally expensive for sensor devices. More generally, unique challenges facing WSNs that have not been addressed are the handling of multidimensional data from sensors efficiently to generate the rules, and the usefulness of the generated rules in sensor networks.

III. Adaptive Rule Triggers on Sensors: ARTS Framework

Shown in Figure 1, The ARTS framework can be divided into two core components, namely the (i) Mining Component and the (ii) Triggering Component.

Mining Component The mining component is designed to parse and learn from sensor data arriving at cluster heads. This is a node centralised model in which we assume that cluster heads elected would have higher resource capabilities than their cluster members. The output of this component is a list of rules pruned based on user-set confidence threshold.

Triggering Component Further described in Algorithm 2, the triggering component is designed to use the generated rules from the mining component to control sensor nodes in clusters. Using the obtained rules, the component sends messages to the sensors that can be power-saved and regulate their actions depending on whether the rule is valid over time.

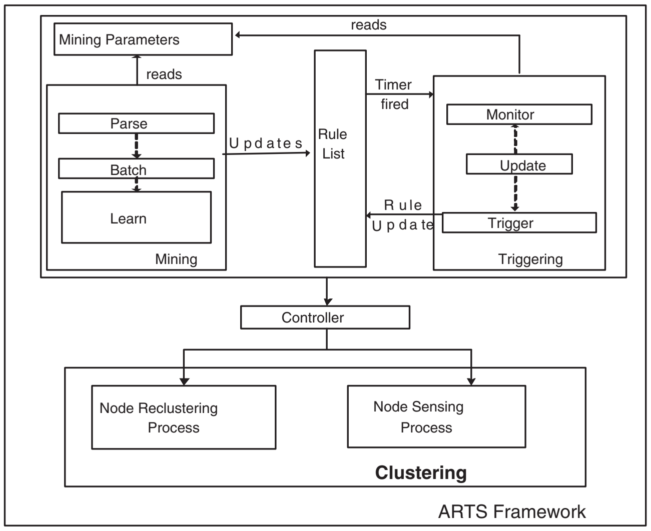

Fig. 1. ARTS Framework

A. Data Representation in ARTS

In [7], the authors analyse the representation of a sensor data stream for mining and propose an interval-list representation. In comparison, our model differs from theirs in representation as we consider mining for rules more efficiently on sensor nodes with multivariate data, while their model is optimised for univariate data. Let us assume we have sensors S_1, S_2, S_3 collecting sound s, temperature *t* and light *l* samples. Typically, data sampled such as t_1 or l_1 is continuous data and would arrive in a random manner. To facilitate generation of only significant rules, we choose to consider only discretised sensor values. Since the arrival time of sensor data value sets is normally random, we model data transactions as data value sets $\{tid; S_n, t_n, s_n, l_n, count\}$, where count is the frequency in time units (e.g. seconds) in which values t_n, s_n or l_n of sensor S_n has remain unchanged.

To reduce the number of rules generated, we count the number of transactions that are frequent, omitting the generation of any k itemsets as in APRIORI and concentrate on only highly correlated rules. This is contrary to the weighted transformation method as used in [7] where itemsets are generated. We assume that transactions are processed in batches $b_1, b_2, \ldots, b_x$ where $1 < x < k$ and $k <$ *number of transactions in* b_x, and that x must also be sufficiently large as transactions cannot occur with equal probabilities (i.e. each transaction of support 1). For instance, given batch b_x, the *support* of any transaction n, with all elements considered is:

$support(n) :=$
total number of transactions with n in b_x
/total number of transactions in b_x.

Subsequently, the *confidence* of a rule, for instance, $(S_1 temperature \rightarrow S_1 light)$
i.e., $a_n \rightarrow a_{n-1}$, generated from a transaction over a user-defined support is given by

$confidence(a_n, a_{n-1}) :=$
$support(a_n, a_{n-1})/support(a_n)$

The confidence measure allows us to trigger rules as long as the premises hold. For instance, if we have a high confidence for a rule stating that a_n implies a_{n-1}, the rule is extracted and we send only reading a_n to the base-station/central node. Upon receiving the reading a_n and utilising knowledge of the rule, the reading of a_{n-1} can be inferred.

B. ARTS Algorithms for Mining and Triggering

In this section, we describe our algorithm, ARTS, for mining rules from sensor data packets arriving at cluster heads. The rules discovered are then used for cluster heads to infer readings of their neighbours and control cluster members' operations.

1) Rule Mining: In the following discussions, ARTS is divided into the mining component in Algorithm 1 and the triggering component in Algorithm 2. We first explain each step of the rule mining process in further detail correponding to the steps in Algorithm 1:

Step 1 S is made up of sensors in the cluster group, as belonging to the cluster head. Transaction batches are collected at that sensor node. Each sensor in S has one or more sensor attributes in A.

Step 2 Each sensor in S has a finite amount of energy that can be obtained from sensor voltage readings. The energy level, e, is in the attribute set A. At each iteration of the algorithm, the energy list is automatically update.

Step 3 For the transaction batch b_n, we generate the coefficient matrix C_{matrix}, for the attributes of sensors in their cluster. The correlation coefficient is calculated based on the current numerical values of the sensor attribute values. The coefficient between any two attributes x_i and

Algorithm 1 ARTS Algorithm: Miner

```
Input: Transaction batch $b_n$
Output: Rule r
1: Let $S = s_1, s_2, ..., s_n$ be the set of sensors operating in a group, $A = a_1, a_2, ..., a_k$ be the set of sensor attributes, and sensor attribute pairs $SA = s_1a_1, s_1a_2, ..., s_na_k$.
2: Obtain energy levels of sensors in $S$ and sort them in ascending order of energy levels, sorted energy lists $Energys_S = e_1, e_2, ..., e_n$.
3: Generate covariance matrix, $C_{matrix}$.
4: Using a bitmap, initialise two sensors in $S$ with greatest probability measure from $C_{matrix}$ and $Energys_S$.
5: Transpose continuous transaction values in $b_n$ to discrete values.
6: Set $frequentItems$ as transaction and $frequentItemsCount$ as the transaction count.
7: for $i = 1$ to $t_h$ do
8:     $currentSupport = transCount_i/h$
9:     if $currentSupport > maxSupport$ then
10:        $maxSupport = currentSupport$
11:    end if
12: end for
13: if $highestSupport >= thresholdSupport$ then
14:    Get most frequent transaction in list
15:    $GenerateRules(frequentTransaction)$
16: else if Number of bits set $> 2$ then
17:    if all bits set then
18:        Reset all bits to 0
19:    else
20:        Remove one bit reflecting current highest correlation in matrix
21:    end if
22: end if
23: Add generated rule with unique rule id to ruleQueue.
```

y_j, and x_i,y_j in SA is given by:

$$C_{matrix}(x_i, y_j) = (h \times (x_i \times y_j) - \sum_{i=0}^{h} x_i \times \sum_{j=0}^{h} y_j) / (\sqrt{h \times \sum_{i=0}^{h} x_i^2 - (\sum_{i=0}^{h} x_i)^2}) \times \sqrt{h \times \sum_{j=0}^{h} y_j^2 - (\sum_{j=0}^{h} y_j)^2})$$

where matrix size, h, = $(n \cdot k)^2$
n = total number of sensors in S
k = total number of attributes in A

Step 4 A binary bitmap is used for the algorithm to consider attribute combinations that contain sensors with high correlation values in C_{matrix} and sensors with the biggest variance in their energy levels in $Energys_S$.

Probability of being selected,

prob$_{s_i}$ $= |Energy_{Si_{max}} - Energy_{S_i} \times 2)/Energy_{Si_{max}}| \times getHighestCoeff(C_{matrix}(x_i, y_j))$

where $C_{matrix}(x_i, y_j) > 0$, and $i >= 0, j >= 0$

The idea is to choose a transaction combination with the biggest contrast in energy levels i.e. to obtain the rule with the lowest number of high-energy sensors and highest number of low energy sensors, to conserve the greatest amount of energy.

Step 5 Numerical values of the sensor attributes are required at the pre-processing step of the algorithm. Following the pre-processing, the numerical values for individual sensors are transposed to discrete values to generate rules and to reduce processing complexity.

Step 6 The $frequentItems$ list stores the most frequent transactions in order and their corresponding counts in $frequentItemsCount$. The sizes of $frequentItems$ list and $frequentItemsCounts$ list are user-defined. $frequentItems$ can contain a user-defined number of items but with transaction size, s, $2 =< s >= k$.

Steps 7-12 We obtain the highest support from transactions already in list.

Steps 13-22 We check if the current transaction in the batch has a support greater than the threshold. If the support within the batch is greater than the threshold, the algorithm generates the rules from the current transaction. The bitmap is then updated by setting an additional bit if probability threshold is met. The rationale is that rules that will be generated next will involve more sensors in the same grouping that has met the threshold to conserve more energy. Otherwise, if the threshold is not met, reduce number of bits set by one.

Step 23 After a rule is generated, it is added to a ruleQueue, given that the user-defined threshold confidence is achieved. The ruleQueue is served periodically per user-defined intervals and ranked on basis of the rule confidence. Out of the rules, a hashtable is then created for the list of sensors to monitor with their expected trigger values.

2) Rule Triggering: Rules in the ruleQueue are processed in the following way:

Algorithm 2 ARTS Algorithm: Trigger

```
Input: Rule r
Output: Rule Trigger
1: Divide rule into two parts with same rule id, if rule is valid.
2: Update monitorList with antecedent sensors.
3: Update triggerList with consequent sensors.
4: Set rule to ACTIVE state for trigger on next timer fire.
```

Steps 1-4 The rule that is obtained from the mining algorithm is made up of two integral parts: rule antecedents and rule consequents. The monitorList and triggerList store values of antecedent and consequent sensors respectively, with shared rule ids. While values of antecedent sensors in monitorList remain true during sensing, we assume the consequent sensors have the implied values in triggerList and activate triggers accordingly (i.e. send trigger instructions and activate a state). In our current implementation, we utilise the trigger message to command sensors to choose the data type of future messages. This is formed by different permutations of the default message type. For instance, if a sensor would send only light and temperature readings, the variations of this are: (i) Send only light reading (ii) Send only temperature reading (iii) Send none (iv) Send all.

IV. Rule-based Cluster Head Selection

The re-clustering interval of the clustering algorithm can be adjusted based on the rule output. Assuming that ARTS is implemented on each sensor node that is also running the clustering algorithm, after the algorithm has run for a period of time on the selected cluster head nodes, rules pruned to the user-specified threshold would be discovered. From the rules discovered, we would then obtain the number of sensors that would be controlled by the cluster head (total number of consequents in the rules). The reclustering interval would be based on this value. The basic idea is that, if cluster heads are able to trigger more sensors in their respective clusters, they should continue to be cluster heads for a longer period of time and vice-versa. The code modification can be localised to the reclustering step of a clustering algorithm. With reference to the HEED algorithm[9], the modifications are required in two areas:

Step 3 in HEED, the initial reclustering probability:
$CH_{prob} \leftarrow (clusterSize - numConsequents)$
$/clusterSize$
, and step 15 in HEED, the repeat reclustering probability:
$If((S_{CH} \leftarrow v : v\_is\_a\_cluster\_head) \neq \emptyset)$
    $CH_{prob} \leftarrow (clusterSize - numConsequents)$
$/clusterSize$
$Else$
    $CH_{prob} \leftarrow min(CH_{prob} \cdot 2, 1)$

V. ARTS Implementation and Results

ARTS has been implemented in TinyOS code, where iHEED [10] cluster nodes are modified to direct messages to ARTS when they are elected as cluster heads. The ARTS algorithm occupies an additional 9446 bytes in mica2 ROM when pre-programmed into all nodes that also run iHEED. By default, the sensing information that iHEED nodes send is light reading. Consequently, the trigger message that is used in the following TOSSIM simulations commands a node not to send its light information when the related rule is true. By using the rules discovered, the aggregate information for the consequent sensors are replaced by an approximate value for the discrete state. In this instance, the light value states low, medium, high translates to approximate values of 250, 500, 750 respectively. The unit of measurement in these experiments is the Credit-Point System (CREP) discussed in [10] to determine the total network energy residue of sensor nodes at runtime.

A. Performance Evaluation

Figure 2 shows a run of ARTS with the usage of rules pruned to different confidence thresholds ranging from 0.5 to 0.9, with 50 nodes. At time=140, we note that an energy savings of around 15% is achieved through using ARTS. However, given the trend observed in Figure 2 i.e. the graph showing the use of HEED and the graph with ARTS diverges as time passes, we expect that for a long enough time, further significant savings can be achieved. From observation, the results show that on any of the confidence thresholds, HEED cluster heads running ARTS outperforms the nodes that run with just using HEED.

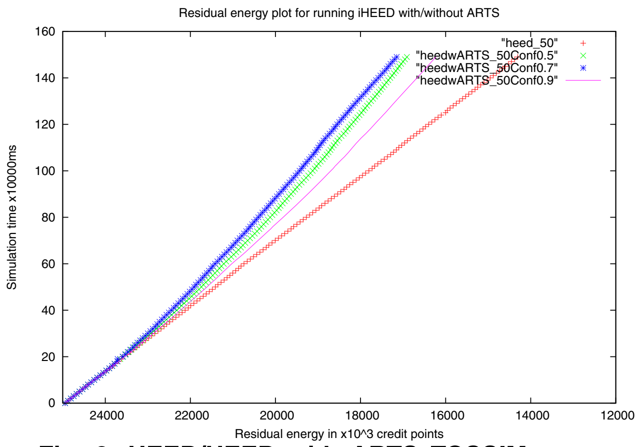

Fig. 2. HEED/HEED with ARTS TOSSIM run with different Confidence levels

B. Overhead

To measure the amount of overhead in using ARTS, we ran a similar experiment with 50 nodes, using a 0.75 confidence threshold for ARTS. Figure 3 shows that the overhead while using ARTS increases only marginally over the default iHEED (at time=140, the overhead is 0.06%), considering the amount of energy saved in Figure 2 when we used a 0.7 confidence threshold.

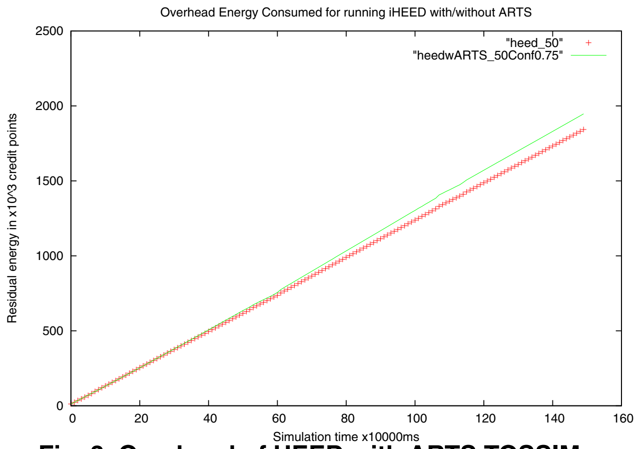

Fig. 3. Overhead of HEED with ARTS TOSSIM run, 50 nodes

C. Scalability

Figure 4 shows the results of the simulation when the algorithm is run on a higher number of nodes, with the experimental run using a confidence threshold of 0.75. In this figure, we observe that as we increase the number of nodes that we use, we see a linear increase in the amount of energy conserved.

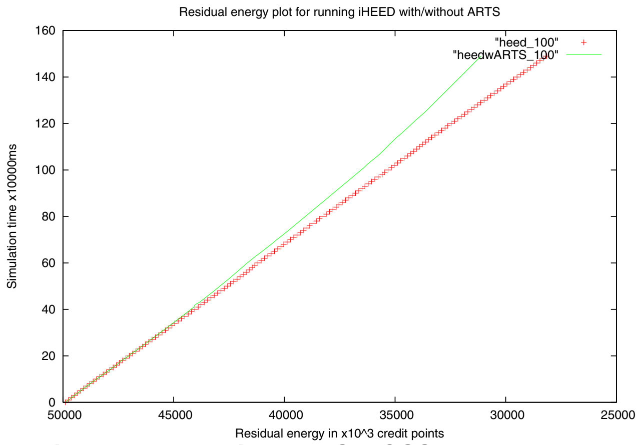

Fig. 4. HEED with ARTS TOSSIM run, 100 nodes

D. Accuracy

We refer to our previous study for the accuracy rates of the rules generated. In [2], we performed a PC simulation run of the ARTS algorithm, assuming that we have a node M that collects light, temperature and microphone readings from three other sensor streams coming from sensors S0, S1, and S2. The synthetic data that we have generated has the attribute that: (1) S0 light readings and S1 light readings have a positive correlation of 0.8 +/- 0.04 (2) S1 light readings and S1 temperature readings have a positive correlation of 0.8 +/- 0.04 (3) S2 light readings and S2 temperature readings have a negative correlation of -0.8 +/- 0.04. Table 1 shows the rules obtained when ARTS run on this dataset for 8 minutes. The column on success rate shows the percentage of packets that have been correctly predicted, knowing the rule value of the rule antecedent.

TABLE I. Table of Rules Discovered in PC Simulation Run

Discovered Rules				
Time	ID	Rules	Conf.	SuccessRate%
1	R1	$S1L[H] \rightarrow S0L[H]$	0.83	82.9% (34/41)
1	R2	$S0L[H] \rightarrow S1L[H]$	1.0	100% (34/34)
3	R3	$S1L[H] \rightarrow S0L[H]$	0.83	82.9%
3	R4	$S0L[H] \rightarrow S1L[H]$	1.0	100%
3	R5	$S0M[Y] \rightarrow S0L[L]$	0.5	45.3% (24/53)
3	R6	$S0L[L] \rightarrow S0M[Y]$	1.0	85.7% (24/28)
5	R7	$S1T[L] \rightarrow S0L[H]$	1.0	100% (20/20)
5	R8	$S0L[H] \rightarrow S1T[L]$	0.73	58.9% (20/34)
7	R9	$S1T[L] \rightarrow S0L[H]$	1.0	100%
7	R10	$S0L[H] \rightarrow S1T[L]$	0.64	58.9%

E. Cluster Heads Energy Consumption

In this section, we detail the results obtained from implementation of rule-based cluster head selection that we have discussed in section 4. Specifically, figure 5 shows the energy consumed by all cluster heads during a 25 minutes simulation run using different seed values (124755, 20000, 25000, 30000 respectively) in TOSSIM. For all experiments using ARTS, the confidence threshold of 0.7 has been used for 50 number of nodes. From the results, we note that by using rule-based cluster head selection with ARTS, the overall cluster-head energy improved by up to 30% (in run 2) when compared to the run with ARTS(sensor node control only). This is a consequence of the on-demand nature of reclustering that ARTS imposes to further extend network lifetime.

VI. Conclusion

We have presented in this paper, our rule-learning framework ARTS, that can complement existing in-network clustering algorithms to prolong sensor network lifetime with minimal overhead. Integrating an in-network clustering algorithm with ARTS allows cluster heads to benefit from rule-based monitoring and controlling of clus-

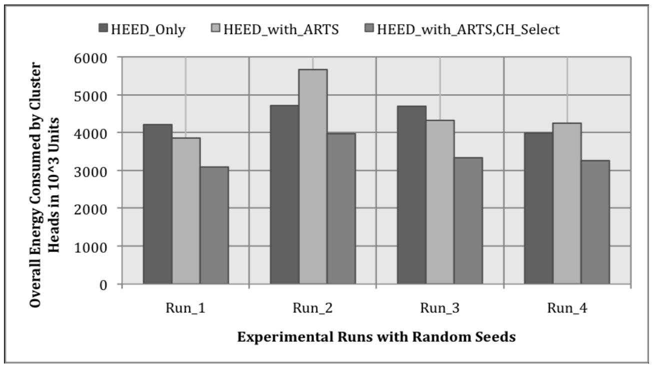

Fig. 5. Overall Cluster-heads Energy Consumption Comparison

ter members, thus, preserving network energy. In addition, ARTS implementation is independent from the clustering protocol used and can also be integrated easily to the cluster head selection process to achieve further energy savings. It provides a low overhead rule learning technique for significant energy conservation. Furthermore, in these experiments, it is likely that the improvements shown with HEED would increase when ARTS is used other clustering protocols such as LEACH [4] that only uses random seeds for reclustering.

References

[1] R. Agrawal, and R. Srikant, Fast Algorithms for Mining Association Rules, *Proceedings of the 20th VLDB*, pp. 487–499, 1994.

[2] S.K. Chong, S. Krishnaswamy, S.W. Loke, and Mohamed Gaber, Using Association Rules for Energy Conservation in Wireless Sensor Networks, *Proceedings of the 23rd ACM Symposium on Applied Computing*, Brazil, 2008.

[3] A. Deshpande, C. Guestrin, and S.R. Madden, Using Probabilistic Models for Data Management in Acquisitional Environments, *Proceedings of the 2005 CIDR Conference*, 2005.

[4] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An Application-Specific Protocol Architecture for Wireless Microsensor Networks, *IEEE Transactions on Wireless Communications*, Vol. 1, pp. 600-670, 2002.

[5] W. Hu, V.N. Tran, N. Bulusu, C.T. Chou, S. Jha, and A. Taylor. The Design and Evaluation of a Hybrid Sensor Network for Cane-toad Monitoring, *Proceedings of Information Processing in Sensor Networks (IPSN 2005/SPOTS 2005)*, Los Angeles, 2005.

[6] P. Levis, N. Lee, M. Welsh, and D. Culler, TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, *SenSys 2003*, 2003.

[7] K.K. Loo, I. Tong, B. Kao, and D. Cheung, Online Algorithms for Mining Inter-Stream Associations From Large Sensor Networks, *PAKDD*, pp. 143-149, 2005.

[8] R. Marin-Perianu, M. Marin-Perianu, and P. Havinga, Movement-based Group Awareness with Wireless Sensor Networks, *Proceedings of Pervasive 2007*, 2007.

[9] O. Younis and S. Fahmy, HEED: A Hybrid, Energy-efficient, Distributed Clustering approach for Ad-hoc Sensor Networks, *IEEE Transactions on Mobile Computing*, Vol. 3, pp. 366–379, 2004.

[10] O. Younis and S. Fahmy, An Experimental Study of Routing and Data Aggregation in Sensor Networks, *LOCAN 2005*, 2005.