A Data Clustering Approach to Energy Conservation in Wireless Sensor Networks

Alexander Sagen [1] and **Cyril Labbé** [2] and **Mohamed Medhat Gaber** [3] and **Shonali Krishnaswamy** [4] and **Agustinus Borgy Waluyo** [5] and **Seng Loke** [6]

Abstract.
This paper presents a new cluster-based power preservation scheme. Our clustering strategy for power saving is defined based on the "similarity of data" coming out from the sensors. The proposed clustering works in conjunction with our learning algorithm to obtain an optimum sleeping time of the sensors without disrupting the monitoring activity. The algorithm helps to effectively regulate the activation or deactivation of the sensor node's radio transceiver, which in turn prolong the lifetime of the network.

We have carried out several real-world experiments concerning power utilisation of wireless sensor devices in different scenarios and found that the proposed method leads to a significant power efficiency improvement with up to four times longer battery lifetime than other cases without such scheme.

1 Introduction

The progressive advances of wireless sensor devices have made the adoption of such devices become increasingly essential and pervasive. Each sensor devices is equipped with a small processor, wireless transmitter, and battery. These sensors are capable of sampling, processing, and communicating environmental parameters including temperature, humidity and light or physiological signs such as blood pressure and movement [10]. After receiving the data from the sensor nodes, the gateway or base station processes the data for specific applications. Typical applications associated with these sensors include environmental monitoring [11], surveillance [8], location tracking [1] and activity recognition [12] to name a few.

As wireless sensor devices are reasonably small-sized and lightweight, energy efficiency is extremely critical considering that sensor nodes are powered only by short life-span batteries, which cannot be recharged during deployment [9]. Consequently, this requires power saving features in order to prolong the sensor node's battery life and ensure that sensors remain operational for the longest time possible.

In light of this issue, this paper aims to investigate power utilization of the sensor node in different conditions and apply a clustering scheme as part of our power saving feature, which is designed to regulate the activation or de-activation of the sensor node's radio transceiver. The proposed cluster-based power saving method is expected to preserve power of the sensor node without overly disrupting the monitoring activity. The clustering of the sensors is done based on the reading values coming out from the sensors. Subsequently, the algorithm determines the sleep time and the cluster boundaries derived from a pre-determined learning process, which takes into account the historical event occurring in the clusters.

By clustering data coming from sensors, we aim to cluster sensors according to "similar" behavior and their sensed data, yielding the notion of sensors being "similar" because they provide "similar" data readings. This knowledge can then be used to save energy, particularly when approximate values are required. A sensor that has just been assigned (by virtue of its history of readings) a cluster can afford to sleep for a while, waking up from time to time to see if it needs to transmit its readings to the base station. A sensor assigned to a cluster (of "similar" data) does not need to transmit its readings (and can even go into deep sleep), unless and until, on waking up, it finds that its current reading has changed enough to, effectively, put the sensor outside of its assigned cluster. This ability of the sensor to sleep (perhaps quite often due to our strategy) helps the sensor to save energy, while still providing a fairly accurate view of the world (compared to when it does not sleep).Because the approach is data-centric, exploiting only a simple characteristic of data values, it is applicable across sensor platforms, different sensor network sizes, different types of sensor data, and applications.This strategy is also more general and effective than the simple strategy of a sensor sending data only when there is change (compared to a previous value, for example), since it takes into account the *significance* of such changes, as measured relative to the other sensors in the cluster (change that is not enough to put the sensor outside of its assigned cluster is ignored).

The remainder of this paper is organized as follows. We present our proposed clustering and energy saving algorithm in Section 2. The infrastructure used to implement the proposed scheme is described in Section 3, followed by our experiments utilizing the newly implemented feature in Section 4. Some existing works related to this paper is given in section 5. Finally, Section 6 concludes the paper.

2 The Approach: clustering data for energy saving

In our method, given a set of data values coming from sensors, these values are clustered, and given this set of clusters (and associated information such as boundaries, centroids),

[1] Monash University, Australia, asagen@gmail.com
[2] University of Grenoble, France, Cyril.Labbe@imag.fr
[3] University of Portsmouth, UK, mohamed.gaber@port.ac.uk
[4] Monash University, Australia, Shonali.Krishnaswamy@infotech.monash.edu.au
[5] Monash University, Australia, awaluyo@infotech.monash.edu.au
[6] La Trobe University, Australia, S.Loke@latrobe.edu.au

Algorithm 1 Algorithm on each sensor: sensors are sleeping following the base station instruction

```
1: Relearn ← True ; TimeSlept ← 0
2: NextSleepTime ← 0 ; Radio ← Off
3: Boundaries ← [−∞, +∞]
4: loop
5:    Value ← Sense()
6:    if Radio = Off and (Value ∉ Boundaries or Relearn = True) then
7:       Radio ← On
8:    end if
9:    if Relearn = False and (Value ∉ Boundaries or TimeSlept ≥ MaxReportTime) then
10:      Send(Value) to base station.; TimeSlept ← 0;
11:      Receive(Boundaries, Relearn, NextSleepTime) from base station.
12:   end if
13:   if Relearn = True then
14:      while Relearn do
15:         Send(Value) to base station.
16:         Receive(Boundaries, Relearn, NextSleepTime) from base station.
17:         Radio ← Off; GoToSleep(NextSleepTime)
18:      end while
19:   end if
20:   Radio ← Off;
21:   TimeSlept ← TimeSlept + NextSleepTime
22:   GoToSleep(NextSleepTime)
23: end loop
```

corresponding sensors are assigned to a cluster. Such information is used to decide whether or not data from a particular sensor is needed, and to decide a sleep time for the sensor.

A strategy, calculating the most desirable sleep time, as a compromise between battery preservation and data resolution has been developed. Section 2.1 states necessary definitions and section 2.2 and 2.3 give algorithms executed on each sensor and on the base station.

2.1 Method overview

Lets assume a system composing of n sensors enumerated and denoted by $s_1, s_2, s_3, \ldots$. The stream d_i of data values from a sensor i is an ordered set $d_{i,1}, d_{i,2}, d_{i,3}, \ldots$ of data values.

Clustering of available data results in a set of k clusters denoted by $c = \{c_1, \ldots, c_k\}$ where c_j is the centroid of cluster j. Each sensor can then be assigned to a cluster according to the last data reading available for this sensor. Let $d_{i,.}$ be the last value sent by sensor $i, \forall i \in [1, n]$, then each time a new data value is processed by the clustering algorithm, sensor i is assigned to cluster j if and only if $dist(d_{i,.}, c_j) = Min_{l=1}^{k} dist(d_{i,.}, c_l)$ defining thus natural "boundaries" between clusters.

The idea is to use these boundaries to detect significant change in the system. For a given sensor, a new sensed data can either be *inside* or *outside* boundaries of the cluster to which the sensor was assigned regarding the previous sensed data. Making a sensor aware of the cluster boundaries it is in allows the sensor to know when its reading changes in a significant way according to the global view of the system.

At the beginning, a learning phase is needed to ensure that enough data values are available to allow the clustering algorithm to build relevant clusters.

2.2 On the sensor

We assume three different levels of energy consumption for a sensor: (i) a deep sleep mode where computing, sensing and radio are off, (ii) an intermediate mode where sensing and computing are on but radio is off, (iii) lastly, the most resource consuming mode where computing, sensing and radio are on.

Algorithm 2 Algorithm on base station: ....

```
1: {S[i] ∈ {Assigned, Relearn} state of sensor i.}
2: {B[i] bounds of the cluster to which i is assigned.}
3: {C[i] centroid of the cluster to which i is assigned.}
4: {R[i] number of needed data values from sensor i.}
5: for i ← 1 to n do
6:    R[i] ← InitNbData ; S[i] ← Relearn
7:    B[i] ← [−∞, +∞] ; C[i] ← −∞
8: end for
9: loop
10:   Receive(data) from sensor i
11:   OldB ← B; OldC ← C
12:   C ← UpdateClusters(d); B ← UpdateClusters(d)
13:   if S[i] = Relearn then
14:      if R[i] ≠ 0 then
15:         R[i] ← R[i] − 1
16:         NextSleepTime ← RealerningSleepTime
17:      else
18:         S[i] = Assigned;
19:         NextSleepTime ← Compute(...)
20:      end if
21:   else
22:      if OldC[i] ≠ C[i] then
23:         S[i] ← Relearn; R[i] ← InitNbData
24:         for j such as B[j] = B[i] do
25:            {for all sensor j in i's new cluster}
26:            S[j] ← Relearn; R[i] ← InitNbData
27:         end for
28:         NextSleepTime ← 0
29:      else
30:         NextSleepTime ← Compute(...)
31:      end if
32:   end if
33:   Send(i, B[i], (R[i] = Relearn), NextSleepTime)
34: end loop
```

On sensor, the first step of the process is a *learning phase* for a fixed time interval (and relearning when required) where readings from sensors are fed into the clustering algorithm.

On each sensor the algorithm 1 is used. As a result of the learning phase (lines 6,13-18), a *sleep time* for the sensor is computed by the base station (see section 2.3) and sent to the sensor, which goes into "deep-sleep" (line 22).

On waking up, the sensor node takes a reading (line 5) and then determines if its reading has changed significantly enough to put it outside its current cluster (by comparing the reading with boundary values : lines 6 and 9). If not, the sensor goes back to sleep (line 22) - this is where the energy savings happen. If the change is significant (lines 10-12), the sensor reports its reading to the base station. The base station then sends new cluster information where the sensor may or may not now be assigned into a new cluster, as well as a new sleep time (line 11). Note that the first $NextSleepTime$ received from the base station is the *first sleep time*. The sensor then goes back to sleep for the given time, and then wakes up to check if its reading is significant enough, and the process repeats itself.

To ensure that a sensor still reports its value from time to time to the base station, a threshold *max report time* is used. The *max report time* is the maximum amount of time given to a sensor between two consecutive reports of its data reading to the base station; if the sensor sent a reading at time t and max report time is r, it must send the next report by $t + r$.

2.3 On the base station

Overview On the base station algorithm 2 is used. There is an initial learning phase where the base station waits for

data readings from the sensor nodes for a fixed time interval. The base station receives these readings and feeds them into the clustering algorithm obtaining clusters of data values, and so, clusters of sensors (corresponding to the sensors the data values belong to). The base station then sends clustering information (centroid, boundaries, etc) to each sensor; the base station also computes and sends the first sleep time to each sensor. Each sensor is now aware of which cluster it belongs to and sleeps for the given time.

From the base station's perspective, a sensor can be in two different states: *Assigned* or *Relearn*. A sensor in the *Assigned* state is a sensor that has been assigned to a cluster after a relearning phase, whereas a sensor in a *Relearn* state is (i) a sensor which is to change cluster (i.e., the sensor reported to the base station that its reading now puts it outside its current cluster boundaries), or (ii) a sensor that is in the same cluster as the sensor which is to change cluster (i.e., a sensor assigned to a cluster in which a new sensor has been assigned "triggering a learning" phase).

Computing sleeping time : line 19 and 30. The sleeping time is computed using two different *activity* notions: the notion of *cluster activity* $\mathcal{A}_c$; this activity is related to the frequency of important events for this cluster. Such events are: boundaries change and the fact that any sensor (in this cluster) changes cluster. for each sensor, we also consider the *sensor activity* $\mathcal{A}_f$, which is related to the frequency of important events regarding this sensor and its cluster. Such event occurs when the sensor itself changes cluster. Let $\delta_{c,i}$ (resp. $\delta_{s,i}$) be the i^{th} inter-arrival time between event i and event $i-1$ for a given cluster (resp. sensor). At the time when event number k occurs, the activity of this cluster (resp. sensor) is defined as follows (with a total of $(k+1)$ events, starting from event 0 to event k):

$$\mathcal{A}_c = \sum_{i=1}^{k} \frac{\delta_{c,i}}{2^{(k-i)}} \qquad \mathcal{A}_s = \sum_{i=1}^{k} \frac{\delta_{s,i}}{2^{(k-i)}}$$

This can be seen as a weighted mean of inter-arrival time giving more importance to recent events against older ones. The $NextSleepTime$ for a sensor is defined as

$$Min\left(\frac{\mathcal{A}_c + \mathcal{A}_s}{2}, MaxSleepTime\right)$$

$MaxSleepTime$ is an upper bound on the sleep time. The intuition behind this formula is that both cluster events and events at the sensor are taken into account. This means that each sensor can receive a sleep time tailored to it. Also, the more active the cluster or the sensor (or both), the more events there are and the smaller the inter-arrival times, and the smaller the sleep time, which means that the sensor is more active (sleep less) when more events are happening.

3 Implementation: Software Infrastructure

The exposed method has been implemented using the following tools: SStreaMWare, a middleware aiming at managing large heterogeneous sensors farms, an open-mining tool kit, which provide several way to cluster data. We describe the overall system (3.1)and the clustering technique we used (3.2).

3.1 Architecture of our Testbed

Our system consists of four components, (*i*) Clustering, (*ii*) Controller, (*iii*) Middleware, and (*iv*) Sensor. The first three components reside in the base station, which is a PC in our case. As the name suggests, our proposed method is defined in the Controller component. The Controller is directly linked to the user and is responsible for maintaining cluster definition and user's query. To ensure a smooth data exchange, communication and translation, we deploy our service-oriented middleware called SSTreaMWare [5, 6, 7] This Middleware receives a query from the Controller and evaluates the query. It is also the first component that receives sensor data readings transmitted directly from the sensor devices to the base station over a wireless link.

The data flow of the system can be described as follows. First, the query is built from the Controller and passed on to the Middleware. Second, upon receiving the query from Controller, the Middleware will evaluate and execute the query. Third, the Middleware receives data readings from the sensor nodes based on the defined query and send the relevant data to the Controller. Fourth, the Controller, after receiving data from Middleware, feeds the data to the Clustering component. Fifth, a cluster definition is then obtained and passed this on to the sensor nodes. The sensor nodes will adjust its parameters based on the given cluster definition.

3.2 Clustering Technique

Different data stream clustering methods have been proposed in the literature [2]. Owing to the scarce resources of individual sensors, adaptation to availability of resources is crucial to the success of the proposed method that potentially can run on a cluster head in a physically clustered sensor network. We have adopted our *RA-Cluster* technique [3] that uses the *Granularity-based Approach* [4] to adapt to memory, battery and processing power variability in real-time. RA-Cluster is a sublinear data stream clustering technique that has proved to produce accurate results while adapting to resource availability by maintaining higher and upper bounds on the different adaptation parameters.

RA-Cluster processes data stream instances as they arrive. Memory adaptation is handled by changing a distance threshold that encourages or discourages the creation of new clusters. Changing the sampling rate is done in response to low availability of battery having the sensor node drains its power rapidly by continuously sensing or listening. Utilising the CPU power is done through randomisation of assignment of new instances by examining a calculated percentage of the already created clusters. Releasing outliers and inactive clusters periodically from memory are two strategies used by RA-Cluster to address the dynamic nature of the streaming data as well as efficiently using the limited available memory onboard the sensor. Memory, battery and CPU adaptations are referred as Algorithm Output Granularity (AOG), Algorithm Input Granularity (AIG) and Algorithm Processing Granularity (APG) respectively.

In our implementation of the proposed method, both AIG and APG have been disabled allowing the clustering technique to boost its accuracy. This is done because *RA-Cluster* runs at the base station with high availability of resources in terms of processing power. Additionally, the base station is not battery powered and is always connected to the power

source. Hence, both *APG* and *AIG* have been disabled. It is worth mentioning that if the proposed method is to run on a cluster head sensor node, both *AIG* and *APG* should be enabled.

4 Experimentation

Type of data	Initial life time	life time with energy saving
Luminosity	12 hours	70 hours
Temperature	12 hours	100 hours

Table 1. Improving life time with clustering

All reported results are coming from real-life experiments. They have been conducted using a set of two sunspots [7].

Sending less and listening doesn't improve energy saving. First set of strategies have been tested on the assumption that saving data emission from sensors to the base station will improve energy consumption. In these strategies sensors are still listening to the base station so that changes in the cluster can trigger as quickly as possible a relearning phase for involved sensors. Experiments show that these kind of strategies are not efficient on the tested sensor. This is mainly due to the fact that the cost of sending data is small compared to the cost of letting the radio turned on.

This shows that for this set of sensor, energy savings can only be done by turning the radio off. In that case, sensor can be put to sleep.

Impact of sleeping strategy on accuracy. The aim of this set of experiments is to analyze the difference of perception that is introduced by the fact that sensors are sleeping. A 12 hours run aims at gathering data from sensors every 2 second. The clustering algorithm is run on these data: this is the *reference*. This is what an application sees of the data without energy saving. The same set of data is replayed with energy saving algorithm in action. If the set of clusters found are not too different and none of the important events are missed, then the clustering approach for energy saving is "accurate".

Found clusters are highly dependent on clustering parameters, mainly, on the frequency of update, that is to say the number of data items that have to be waited for for an update of the cluster. When the algorithm for energy saving is running, the frequency of data sent drops and to obtain similar clusters, the the frequency of update has to be increased. The value that has to be chosen for the frequency of update depends on variability of data.

That's why two sets of experiments have been conducted: one with quite steady data (temperatures) and one with highly variable data (luminosity). Figure 2 and 1 shows the resulted clusters from the associated data.

Impact of sleeping strategy on life time Table 1 report results of two experiments aiming at testing the improvement of the live time of a systems composed of two sensors using the proposed method. This result shows that huge improvement have been optioned using this method: the life time is between 6 and 8 time longer than without energy saving. They also show that improvement depends on the variability of data. The longest lifetime is obtained with steady data (temperature) and highly variable data (Luminosity) leads to less improvement, as expected from our sleep time formula.

[7] http://www.sunspotworld.com/

5 Related Work

Several approaches have been developed that focus on reducing the amount of data communicated in-network and thus, prolong sensor network lifetime. The specific type of approach that can be applied to reduce communication is dependent upon the granularity of data required for theWSN application. For instance, a critical-sensing WSN application such as smart home health care systems [13,18] would require accurate values from sensor nodes at all times to monitor patient health conditions, whereas coarse-granularity data would suffice for certain event detection systems [16,17,22]. As a consequence, various data granularity control mechanisms have been proposed to cater to the data requirements of WSN applications, utilizing approximation, prediction and use of associations/correlations. Conversely, in applications where approximations in the collected data can be tolerated, approximations on sensory data can be performed instead to further reduce data transmissions. The application of approximation to reduce data transmissions in-network has been explored in works such as [23,21,13]. The third class of techniques that can be applied to reduce network data transmissions is prediction. Prediction techniques at the node level derive spatial and temporal relationships or probabilistic models from sensory data to estimate local data readings or readings of neighbouring nodes. When sensory data of particular sensor nodes can be predicted, these sensor nodes are then suppressed from transmitting the data to save communication costs. Similar to compression and approximation, prediction-based techniques are also required to run in a light-weight manner on sensor nodes. In the literature, prediction techniques have been proposed as algorithms for enhancing data acquisition in [19,15,20] and as generic light-weight learning techniques to reduce communication costs in transmissions. In [23-26] an algorithm for learning highly correlated rules from sensory data is presented that prunes/leverages the associations to develop "context-aware" strategies for energy-efficient control of sensors.

In essence, these approaches have explored how sensor data and knowledge obtained from this data can be used in some measure reduce data transmission and therefore prolong sensor lifetime. However, to the best of our knowledge none of these techniques have used changing cluster patterns as a control mechanism for energy efficient operations in a sensor network, which is the core contribution of this paper.

6 Conclusion and Future Work

Depending on the sensor, reducing communication does not always leads to energy saving, in that case the only way to provide energy saving is by turning the radio off. Finding a good sleeping time is then crucial and results in a compromise between energy consumption and accuracy.

As shown here, clustering data from sensors can be a valuable tool to determine the sleeping time of sensor and extends the life of systems. The first results reported here are promising as the life time has been increased by 6 and 8 times the initial life time. The proposed method is quite general and can be fitted to any type of sensor on which radio can be turned off.

Some important future works are needed such as to further investigate a better way of defining cluster boundaries and clearly quantify the difference of "accuracy" introduced by the deep sleep of sensors. Impact of the two introduced

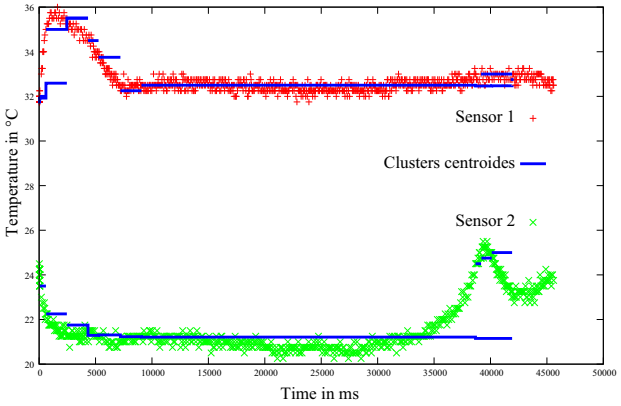

Figure 1. Temp. and clusters, sleeping for energy saving.

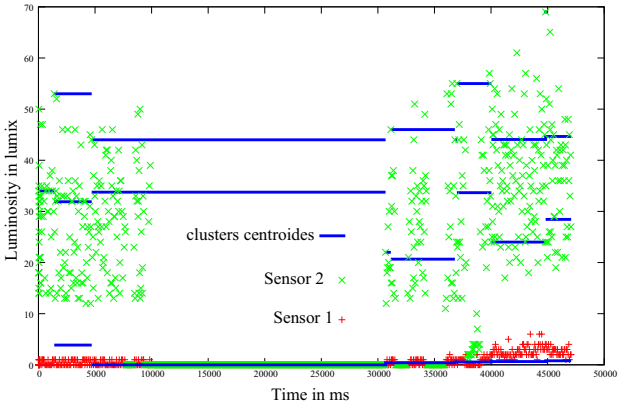

Figure 2. Lum. and clusters, sleeping for energy saving.

threshold on accuracy and energy saving have to be carefully studied and parameters of the clustering algorithm should be computed and adjusted dynamically according to characteristics of data streams. Clustering could also be done on the sensors themselves in order to achieve more efficient energy savings.

Other clustering techniques should also be investigated as well as the impact of the number of sensors involved in the systems.

REFERENCES

[1] M.X. Chen and Y.D. Wang. An efficient location tracking structure for wireless sensor networks. *Computer Communications*, 32(13-14), 2009.

[2] M.M. Gaber, A. Zaslavsky, and S. Krishnaswamy. Mining Data Streams: A Review. *ACM SIGMOD Record*, 34(1), 2005.

[3] M.M. Gaber, and P.S. Yu. A Holistic Approach for Resource-aware Adaptive Data Stream Mining. In *Journal of New Generation Computing*, 25(1):95-115, 2006.

[4] M.M. Gaber. Data Stream Mining Using Granularity-based Approach. In *Foundations of Computational Intelligence* Vol. 206/2009, Springer , Germany, 2009.

[5] L. Gurgen, C. Roncancio, C. Labbe, A. Bottaro and V. Olive. SStreaMWare: a service oriented middleware for heterogeneous sensor data management. In *Proc. of The 5th ICPS*, 2008.

[6] L. Gurgen, C. Roncancio, C. Labbe and V. Olive. Update Tolerant Execution of Continuous Queries on Sensor Data. In *Proc. of The 5th INSS*, 2008.

[7] L. Gurgen, J Nystrom-Person, A. Cherbal, C. Labbe, C. Roncancio, and S. Honiden. Plug-Manage Heterogeneous Sensing Devices. In *Demonstration in 6th Int'l Workshop DMSN*, 2009.

[8] H. Liu, P.J. Wan and N. Pissinou. Maximal lifetime scheduling in sensor surveillance networks. In *Proc. of The 24th INFOCOM*, 2005.

[9] A. Hac. *Wireless Sensor Network Designs.* John Wiley and Sons, 2003.

[10] I. F. Akyildiz, W. Su and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38:393–422, 2002.

[11] T. Wark, W. Hu, P. Corke, J. Hodge, A. Keto, B. Mackey, G. Foley, P. Sikka and M. Brnig". Springbrook: challenges in developing a long-term, rainforest wireless sensor network. In *Proc. of The 4th Int'l ISSNIP*, 2008.

[12] W. S. Yeoh, W. Jiankang, I. Pek, Y. H. Yong, X. Chen and A. B. Waluyo. Real-time tracking of flexion angle by using wearable accelerometer sensors. In *Proc. of The 5th International Workshop BSN*, 2008.

[13] C.R. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. D. Minassians, G. Dervisoglu, L. Gutnik, M. B. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa,M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E.S. Leland, T. Pering, and P.K. Wright. Wireless sensor networks for home health care. In *21st Advanced Information Networking and Applications Workshops*, Niagara Falls, Canada, 2007.

[14] J.Y. Chen, G. Pandurangan, and D. Xu. Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis. In *4th International Symposium on Information Processing in Sensor Networks*, California, USA, 2005.

[15] B. Gedik, L. Liu, and P.S. Yu. Asap: An adaptive sampling approach to data collection in sensor networks. *IEEE Transactions on Parallel and distributed systems*, 18(12), 2007.

[16] T. He, S. Krishnamurthy, J.A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu, J. Hui, and B. Krogh. Energy-efficient surveillance system using wireless sensor networks. In *2nd International Conference on Mobile Systems, Applications and Services*, pages 270–283, Boston, USA, 2004.

[17] M. Hefeeda and M. Bagheri. Wireless sensor networks for early detection of forest fires. In *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–6, Pisa, Italy, 2007.

[18] A. Keshavarz, A.M. Tabar, and H. Aghajan. Distributed vision-based reasoning for smart homecare. In *ACM SenSys Workshop on Distributed Smart Cameras*, pages 105–109, Boulder, USA, 2006.

[19] S. Madden and M.J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *18th ICDE*, San Jose, USA, 2002.

[20] M.A. Sharaf, J. Beaver, A. Labrinidis, and P.K. Chrysanthis. Tina: A scheme for temporal coherency-aware in-network aggregation. In *ACM Workshop on Data Engineering for Wireless and Mobile Access*, pages 69–76, California, USA, 2003.

[21] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: New aggregation techniques for sensor networks. In *ACM Conference on Embedded Networked Sensor Systems*, pages 239–249, Baltimore, USA, 2004.

[22] A. Tavakoli, J. Zhang, and S.H. San. Group-based event detection in undersea sensor networks. In *2nd International Workshop on Networked Sensing Systems*, San Diego, USA, 2005.

[23] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *1st IEEE International workshop on sensor network protocols and applications*, California, USA, 2003.

[24] S.K. Chong, M.M. Gaber, S. Krishnaswamy, and S.W. Loke. A rule learning approach to energy efficient clustering in wireless sensor networks. In *2nd International Conference on Sensor Technologies and Applications*, Cap Esterel, France, 2008.

[25] S.K.Chong, S.Krishnaswamy, and S.W.Loke. A context-aware approach to conserving energy in wireless sensor networks. In *3rd IEEE International Conference on Pervasive Computing and Communications Workshops*, Hawaii, USA, 2005.

[26] S.K. Chong, I. McCauley, S.W. Loke, and S. Krishnaswamy. Application of context-awareness to data muling. In *1st ACM International Workshop on Context-Awareness for Self-ManagingSystems*, , Toronto, Canada, 2007.