An approach for clustering sensor nodes by data similarity

Marcus Lemos[1,3], **Carlos Giovanni**[1], **Marcel Mei**[1], **Kenneth Sepulveda**[1], **Dara Oliveira**[1], **Abdias Viana**[1], **Ricardo Rabelo**[2], **Raimir Holanda**[3]

[1]State University of Piaui (Uespi)
Teresina, Piaui, Brazil

[2]Federal University of Piaui (UFPI)
Teresina, Piaui, Brazil

[3]University of Fortaleza (Unifor)
Fortaleza, Ceara, Brazil

{marvin,cgnc}@uespi.br, {marcel,kennethsousabr,dara}@gmail.com

abdviana@gmail.com, ricardoalr@ufpi.edu.br, raimir@unifor.br

Abstract. *In Wireless Sensor Networks, data aggregation is a crucial task since it is capable of reducing the network traffic, increasing the overall network lifetime. However, most works consider aggregation when nodes are close to each other since the common assumption is that if a set of sensor nodes are physically close to each other, they produce highly correlated data. Nevertheless, the fact that nodes are close to each other does not guarantee data correlation in order to perform aggregation. Hence, the objective of this paper is to present Akasen to clusters sensor nodes based on the data similarity. Results from experiments in IntelLab dataset show that the Akasen reduces energy consumption by* $5.5\times$*, providing a solution to be considered in wireless sensor networks scenarios.*

1. Introduction

Wireless Sensor Networks (WSN) consist of a large set of small electronic devices, called sensor nodes, capable of sensing and transmit (usually by a wireless transceiver) environmental variables, such as temperature, humidity, light and so on [Rawat et al. 2014]. Considering that sensor nodes are usually battery-powered, the energy consumption imposes severe restrictions on this kind of network. Since the transceiver requires more energy than sensing activities, some researches on the literature have directed studies to reduce the overall data traffic generated by the sensors nodes [Jiang et al. 2011].

As many applications are related to the periodic monitoring, and it is well known that, in such scenarios, redundant data may be sent by sensor nodes [Carvalho et al. 2011], most works in literature consider data aggregation to reduce the overall traffic [Kridi et al. 2016], [Gielow et al. 2015] and [Kim et al. 2014]. In these works, aggregation occurs when nodes are close to each other, exploiting the spatial correlation between them. The common assumption is that if a set of sensor nodes are physically close to each other, they produce highly correlated data [Karasabun et al. 2013]. However, the fact that nodes are close to each other does not guarantee data correlation in order to perform aggregation. To exemplify that observation, consider the temperatures taken from the Intel Lab project dataset[1]. Figure 1 compares the correlation of temperature of one sensor node

[1]http://db.csail.mit.edu/labdata/labdata.html

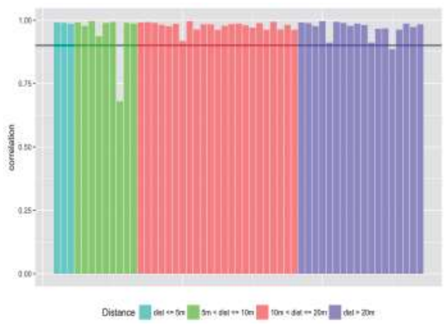

Figure 1. Correlations between node 1 and others sensor nodes, from Intel Lab dataset. The black line shows a desired threshold. Notice that even distante nodes are correlated.

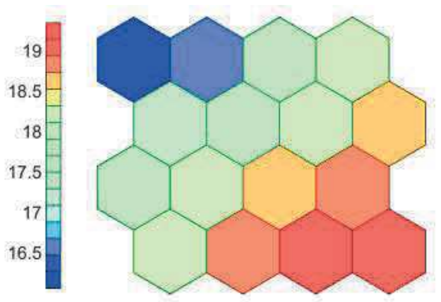

Figure 2. Heatmap showing that is possible to cluster nodes based on the similarity of their measurements

1 with all others nodes. It is clear that even distant nodes have temperature correlated with Node 1. In Figure 2, each circle of the grid means a group of sensor nodes put together based on the correlation of first 200 temperature measurements. The grid, created by SOM algorithm [Kohonen 1998] gives the idea that nodes from different geographical areas could fall in same clusters.

In fact, the authors in [Lemos et al. 2017] exploited that idea and presented ACASIM, a clustering approach where the physical sensor nodes are clustered according to their measurements similarity. And by selecting only a sub-set of sensor nodes from each cluster, ACASIM could reduce energy consumption, while attending the applications requirements. The clustering process is controlled by a error threshold, i.e., whenever the measurements from a sensor nodes does not not fit its clusters anymore, ACASIM reorganize all the clusters again. However, we argue that the reduction of energy consumption could be improved if the algorithm first tries to reconstruct only the clusters whose measurements are deviating from the pattern. Hence, we propose Akasen, *Adaptive K-Means Algorithm for Sensor Networks*, for clustering sensor nodes in a WSN. Akasen uses K-Means algorithm to group the nodes according the similarity of their measurements. The value of K is find automatically according to a $threshold$ parameter defined by the application. And differently from ACASIM, Akasen tries first to reallocate the sensor nodes whose measurements diviate from their clusters, before construct all clusters again.

The article is structured as follows: Section 2 discusses in more detail the operation of Akasen. Section 3 presents the performance evaluation of the proposed, while Section 4 discusses the results. Finally, in Section 5, the conclusions are presented along with some future research directions.

2. Adaptive K-Means Algorithm for Sensor Networks

The main goal of Akasen is to clusters sensor nodes based on the similarity of their measurements. The clustering process guarantees that the correlation between the measurements from each cluster are below an $threshold$ parameter. In that sense, it not necessary that all sensor nodes transmit their measurements anymore. As all values are correlated and the sensor nodes are homogeneous, the sink node needs to receive the measurements from one sensor node of each cluster, at least. This approach leads to a significant improvement in the energy consumption reduction. At any point in time, Akasen is in one of the following modes: configuration, training, and maintaining (Figure 4), which are described next.

2.0.1. Configuration

Initially, it is necessary the configuration of three parameters: $training\_window$, $initial\_k$, $threshold$, according Figure 4a. The $training\_window$ delimiters how many measurements the sink nodes must receive, during the training mode in order to execute K-Means algorithm, while $initial\_k$ and $threshold$ are used by the training itself to adjust the clusters formation, as explained later in this section. The values of these parameters is a decision project and are based upon the nature of the measurements. After that, training mode begins (4b).

2.0.2. Training

This step represents the main core of Akasen, since it is responsible for executing K-Means to clusters sensor nodes. First, sink node must receive measurements sent from all sensor nodes during a specific time-window, represented by $training\_window$. Whether more than one measurements from each node arrives, sink node aggregates them using a *average* operation. The measurements are stored in a matrix X_n^m, where $n > 0$ represents the values monitored by each sensor node and $m \geq 1$ corresponds to the sensors nodes in the network. In sequence, K-Means algorithm is executed. Usually, the value of K, which corresponds to the number of desired clusters, must be given *a priori*. However, is not always easy to determine the best value. A traditional strategy to select K is to execute K-Means repeated times and increment the value of K in each execution. The goal of this procedure, named *Elbow method* [Jain 2010], is to find the K value that makes the variance of the clusters falls below 0.2. The value 0.2 usually means an "elbow" in a scree plot, as pointed out by Figure 3.

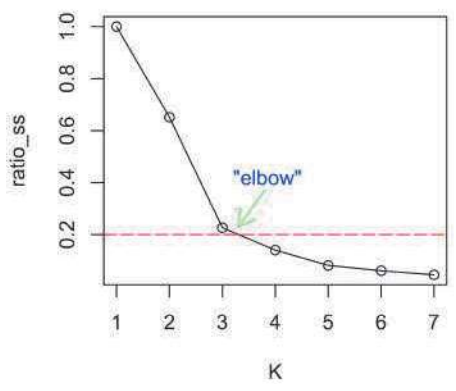

Figure 3. Scree plot. The "elbow" (pointed out by the green arrow) represents the point where adding another cluster does not produce better models.

The intuition is that below the "elbow", the variance of the generated model slowly decreases, i.e., adding another cluster does not produce better models. However, Akasen does not use variance to control the value of K. Instead, it refines the clustering models until a criterion is reached. Algorithm 1 shows how K-Means works.

```
   **Data**: $initial\_k$, $threshold$, $matrix$
   **Result**: $Clusters$
1  $K = initial\_k$;
2  **while** *True* **do**
3      $kmodel \leftarrow$ kmeans($matrix$, $K$);
4      clusters_over = exists_clusters_over_threshold($kmodel$, $threshold$);
5      **if** *clusters_over_threshold == False* **then**
6          $best\_k \leftarrow K$;
7          $best\_model \leftarrow kmodel$;
8          break;
9      **else**
10         $K \leftarrow K + 1$;
11     **end**
12 **end**
```

Algorithm 1: K-Means

Initially, the K variable is configured with the $initial_k$ value (Line 1) and, in sequence, the algorithm enters a loop (Line 2). Each loop iteration executes *k-means* with the last K value and stores the model in $kmodell$ variable (Line 3). Then Akasen executes a function, named *exists_clusters_over_threshold*, which checks if some of the clusters (in $kmodel$) contain at least one node whose euclidian distance between its measurements and the center of the clusters is over $threshold$. In the case of $True$, K is incremented, and the process is repeated. Otherwise, Akasen has just found the best K value, and the loop is broken down (after storing K and $kmodel$ values). Then, Akasen selects only one sensor node from each cluster and broadcast a special message, warning all other sensor nodes to enter a low power state, along with the coefficients of a linear model which represents the measurements of each node, according to the strategy used in [Lemos et al. 2017]. These linear models will be used during Maintaining mode. All these procedures are depicted in 4b. Then, it begins the maintaining mode.

2.0.3. Maintaining

As each cluster contains only node whose measurements are close to each other (based on $threshold$ value), it does no matter if the distance between the sensor nodes is small or not; the selected nodes can represent all other nodes from their clusters. Moreover, as only one node is selected, the overall energy consumption of the WSN is reduced. However, it can happen that the measurements from some sensors nodes start to deviate from the pattern of their clusters. In that case, Akasen should reconstruct the clusters. Therefore, in maintaining mode, each sensor node not transmitting checks the status of its measurements. Whether the absolute error between the new measurements and the measurement predicted by the linear model is bigger than $threshold$, the sensor nodes send a message to sink node, along with its new measurements. Before executing K-Means again, sink node verifies if that sensor node can be reallocated to another cluster. In the case of success, sink node generates a new linear model and send back to the sensor node. However, if the reallocation process fails, K-Means is executed again (4c).

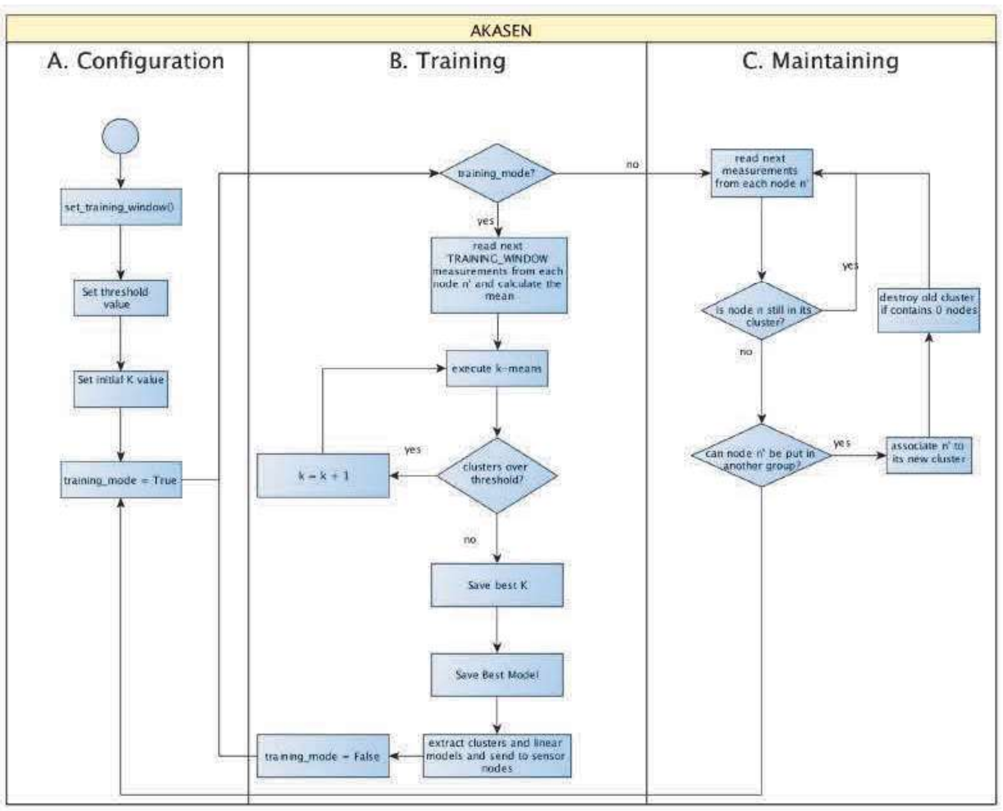
AKASEN
A. Configuration
B. Training
C. Maintaining
set_training_window()
Set threshold value
Set initial K value
training_mode = True
training_mode?
no
yes
read next TRAINING_WINDOW measurements from each node n' and calculate the mean
execute k-means
yes
k = k + 1
clusters over threshold?
no
Save best K
Save Best Model
extract clusters and linear models and send to sensor nodes
training_mode = False
read next measurements from each node n'
yes
is node n still in its cluster?
no
destroy old cluster if contains 0 nodes
can node n' be put in another group?
yes
associate n' to its new cluster

Figure 4. Akasen

3. Performance Evaluation

Akasen have been evaluated based on real data obtained from the Intel project, Berkeley Lab Data[2], as it is publicly available and used by several works on literature [Gielow et al. 2015, Carvalho et al. 2011, Jiang et al. 2011]. The dataset consists of the measurements of 54 sensor nodes measured every 31 seconds between 28 February and 4 April 2004. However, the nodes do not have the same number of measurements (possibly due to errors or failures in the sensor-reading process). Therefore, to properly evaluate the performance of the simulations, the first 5,000 measurements were taken (corresponding approximately to the measurements of the first day). The original dataset contains missing values, which were interpolated with the average of the values from the previous and subsequent measurements. The results described here have been produced by simulations performed with R[3] language. Akasen was compared with LEACH protocol [Heinzelman et al. 2000]. In the LEACH protocol, only cluster heads communicate with the sink node, and each sensor node has probability P of being elected cluster head. The cluster heads receive all measurements from nodes in their clusters and generate an aggregate value from them. In this work, the *mean* is considered as the aggregation function. The single-hop (direct) communication [Meghji 2011] was used as a benchmark case. In the single-hop communication, each node transmits its measurements directly to the sink node.

The mean squared error (MSE) and energy consumption were chosen to evaluate

[2]http://db.csail.mit.edu/labdata/labdata.html
[3]http://www.r-project.org/

Akasen properly. As the created clusters may include sensor nodes not physically close to each other, MSE helps to investigate how reliable is the measurements sent by the selected sensors, i.e., how they can be representative of multiple regions. The energy consumptions measures how well Akasen selects the sensor nodes in order to save energy. The models (from both metrics) defined in [Lemos et al. 2017] were used.

4. Results and Discussion

4.1. LEACH protocol results

In LEACH protocol, every node has a probability P of being selected as a cluster head. For this reason, it was simulated eight scenarios with different values of P. The values of P considered were 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Table 1 shows the results in the eight scenarios. Considering the table, MSE is closely correlated with the value of P, which was expected since, in the LEACH protocol, the clusters are based on the nodes' proximity (based on physical distance).

Small values of P mean that few cluster heads are selected. In this case, the MSE is higher since each cluster may contain nodes with measurements significantly different from each other. As the value of P increases, the number of cluster heads increases as well. Hence, the MSE decreases because the clusters contain more nodes close to each other (based on physical distance).

Regarding energy consumption savings (in compare with single-hop communication), as the value of P increases, more nodes are selected as cluster heads, which boosts the number of transmissions inside the network. Consequently, the energy consumption savings decrease. As the value of P approaches 1, the energy consumption of LEACH becomes more similar to the single-hop communication.

4.2. Akasen results

Akasen was simulated in eight different scenarios with the following $THRESHOLD$ values: 0.1, 0.3, 0.5, 1.0, 2.0, 3.0, 4.0, and 6.0. Table 2 shows the results in the eight scenarios simulated. It is possible to see a positive correlation between the $THRESHOLD$ and the energy consumption savings; i.e., as the $THRESHOLD$ value increases, the percentage of energy consumption savings also increase. This is because high values of that parameter generate fewer clusters, and each cluster contains a large number of nodes. As only one node responds, the energy consumption decreases. However, regarding MSE, there is an increasing tendency as the value of $THRESHOLD$ rises.

4.3. Comparing the results of LEACH and ACOSIM

The highest value of MSE in Akasen (7.68) is about $3.7\times$ lower than highest value of MSE in LEACH scenarios (28.72). Considering these same scenarios, energy savings of LEACH is slightly superior (about $1.5\times$). It is important to stress that these is just the scenarios with highest energy savings in both solutions. However, a MSE equals to 28.72 makes LEACH, with $P = 0.2$, infeasible for WSN applications. Considering Node 18, Figure 5 shows the absolute error between the transmitted values from the nodes' clusters and its real measured values during the simulations in the scenarios with $P = 0.2$. From theses results, it is possible to see that the error falls beyond $5\,^{\circ}\mathrm{C}$ in most of the time.

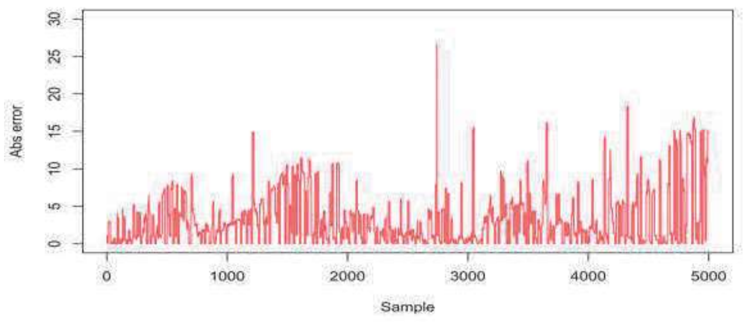

Figure 5. Absolut error in LEACH scenarios with P equals to 0.2

In Akasen, the highest value of MSE is 7.68 (when $THRESHOLD = 6$), which is about $3\times$ worse than the lowest MSE for the LEACH protocol ($MSE = 2.09$ when $P = 0.9$). However, with this same configuration, the energy consumption reduction of ACOSIM is 53.92%, which is about $5.5\times$ better than LEACH protocol with $P = 0.9$ (the scenario with the lowest MSE), where the percentage energy savings value is 9.80%.

Table 1. LEACH results

P	Energy (mJ)	Sav.(%)	MSE
0.2	26,227.10	79.91	28.72
0.3	39,334.82	69.88	22.83
0.4	52,422.89	59.86	18.06
0.5	65,567.48	49.80	13.98
0.6	78,544.14	39.86	10.39
0.7	91,599.68	29.86	7.26
0.8	104,681.71	19.85	4.50
0.9	117,804.67	9.80	2.09

Table 2. Akasen results

Thr.	Energy (mJ)	Sav.(%)	MSE
0.1	130,604.90	0.005	0.013
0.3	130,461.50	0.115	0.086
0.5	130,287.10	0.24	0.14
1.0	90,532.30	2.53	0.48
2.0	70,338.40	13.10	1.87
3.0	60,688.79	29.02	3.89
4.0	55,551.39	38.33	5.63
6.0	30,679.35	53.92	7.68

5. Conclusions and Future Works

This paper presented Akasen, an algorithm to clusters sensor nodes based on the similarity of their measurements with the purpose of reducing energy consumption. Akasen performs the clustering based on a modified version of K-Means which automatically choose the value of K considering a $threshold$ parameter. This approach leads to group together sensor nodes not close physically (based on the physical distance), as traditional works on literature do. However, since the measurements are correlated, according to an $threshold$ parameter, sink node may select only few sensor nodes from each cluster in order to transmit its measurements, while the others may change their states to a low-power mode operation. Hence, there is an improvement in the overall energy reduction of the WSN.

Simulations have shown that Akasen, compared to LEACH protocol, has better performance regarding energy consumption (about $5.5 \times \%$) and MSE ($3.7\times$), indicating the feasibility of the proposed algorithm. As future work, we intend to: (i) perform simulations in more complex environments, taking into account other variables such as the routing protocol, (ii) compare Akasen with moderns clustering algorithms, including ACASIM, and (iii) develop an algorithm to perform the optimal selection of sensor nodes that will transmit the measurements.

References

Carvalho, C., Gomes, D. G., Agoulmine, N., and de Souza, J. N. (2011). Improving prediction accuracy for wsn data reduction by applying multivariate spatio-temporal correlation. *Sensors*, 11(11):10010–10037.

Gielow, F., Jakllari, G., Nogueira, M., and Santos, A. (2015). Data similarity aware dynamic node clustering in wireless sensor networks. *Ad Hoc Networks*, 24:29–45.

Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. (2000). Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, volume 8 of *HICSS '00*, pages 8020–, Washington, DC, USA. IEEE Computer Society.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651 – 666. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

Jiang, H., Jin, S., and Wang, C. (2011). Prediction or not? an energy-efficient framework for clustering-based data collection in wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):1064–1071.

Karasabun, E., Korpeoglu, I., and Aykanat, C. (2013). Active node determination for correlated data gathering in wireless sensor networks. *Computer Networks*, 57(5):1124 – 1138.

Kim, J.-Y., Sharma, T., Kumar, B., Tomar, G. S., Berry, K., and Lee, W.-H. (2014). Intercluster ant colony optimization algorithm for wireless sensor network in dense environment. *International Journal of Distributed Sensor Networks*, 2014.

Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1):1 – 6.

Kridi, D. S., de Carvalho, C. G. N., and Gomes, D. G. (2016). Application of wireless sensor networks for beehive monitoring and in-hive thermal patterns detection. *Computers and Electronics in Agriculture*, 127:221 – 235.

Lemos, M., d. Carvalho, C., Lopes, D., Rabelo, R., and Filho, R. H. (2017). Reducing energy consumption in provisioning of virtual sensors by similarity of heterogenous sensors. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, pages 415–422.

Meghji, Mahirand Habibi, D. (2011). *Transmission Power Control in Single-Hop and Multi-hop Wireless Sensor Networks*, pages 130–143. Springer Berlin Heidelberg, Berlin, Heidelberg.

Rawat, P., Singh, K. D., Chaouchi, H., and Bonnin, J. M. (2014). Wireless sensor networks: a survey on recent developments and potential synergies. *The Journal of Supercomputing*, 68(1):1–48.