A Framework for Sensor Stream Reduction in Wireless Sensor Networks

Andre L. L. Aquino
Computer Institute – Federal University of Alagoas
Campus A. C. Simoes, Av. Lourival Melo Mota, s/n, Tabuleiro do Martins
Maceio-AL, Brazil, zipcode: 57072-970
Email: alla@ic.ufal.br

***Abstract*—This work presents a general methodology to perform sensor stream reduction in wireless sensor networks. This methodology considers the application requirements, the reduction design, and the data reduce validation. Specifically, the reduction design, we present a architecture that can be applied to reduce the data when it is sensed or routed through to sink. The objective of this work is to show step-by-step how we can realize reduction applications in wireless sensor networks by using our methodology. The study cases show the usefulness of our methodology applied on a general sensing and a real time scenarios.**

Keywords-Sensor stream; Reduction; Wireless sensor networks.

I. Introduction

A wireless sensor network (WSNs) [1] is a special type of network represented by a set of sensor nodes, where each node works by detecting events, performing quick local data processing, and transmitting data through a ad-hoc wireless communication. Sensor nodes work in a cooperative way, and all their measurements are sent to a special node called sink. WSNs are commonly used in applications such as environmental, habitat, or industrial monitoring [2]. The phenomena monitored usually require measurements of physical variables, such as temperature, pressure, and humidity, and a single node can monitor one or more of these variables.

The data generated by WSNs have particular characteristics, as they arrive at the sink node in an online fashion, are unlimited, and there is no control in the messages order of arrival. This type of data is nowadays referred as *data stream* [3]. Besides the usually characteristics of any data stream, a *sensor* data stream has other peculiar features, since it represents only a sample of the entire population, is usually imprecise and noisy, and of moderate size.

There would be no problem in collecting, processing and transmitting a data stream if a WSN was not limited by a series of imposed restrictions. WSNs have a limited energy source (being a microelectronic device, a sensor node can be only equipped with a limited power source), low computational power, and reduced bandwidth, plus the weakness of a wireless medium communication. From these restrictions, the energy is the most critical, as the sensor lifetime strongly depends on the battery lifetime, and in some applications renewal of power resources might not be possible.

In these conditions, dealing with all the data stream becomes an unfeasible task. If the sensor node transmits all its measurements, it spends a lot of energy, and there is no guarantee that the data will not be delayed or lost. In order to respect the WSN restrictions, many strategies for data processing were proposed, including data aggregation, data fusion and data reduction.

The simplest strategy of the aforementioned is data aggregation [4]. Data aggregation reduces the data considering some application requirement, for example the data location. The main objective of this approach is to reduce the network data traffic by reducing the number of packets by aggregating them, regardless of data semantics. Data fusion [5], in contrast, is a more sophisticated strategy that focuses on processing data gathered by sensor nodes by benefiting from their processing capability. By exploiting the synergy among the available data, fusion techniques can reduce the amount of data traffic, filter noisy measurements, and make predictions and inferences about the monitored entity. Finally, the sensor stream reduction [6] strategies take advantage of the data stream algorithms characteristics to allow an *online* reduction of the data sensing based on application requirements, and are simple to implement.

From the three strategies, considering the application restrictions and the advantages of online data processing, we focus on sensor data stream reduction. Regarding the WSN application restriction, its use is motivated by three main factors. First, data transmission requires more energy than data measurement. Hence, reducing the data transmitted reduces the energy spent. Second, in order to reduce the data, the sensor node needs to perform constant and quick large local data processing, which requires simple and smart reduction strategies. Reduction strategies based on data streams are suitable here since they process the data locally and independently of previous data, avoiding the complete data storage and/or preprocessing. At last, as we have reduced bandwidth, sending large amounts of data can be problematic, causing excessive delay in response time and invalidating the data.

Although there are many sensor stream reduction strategies available in the literature [3], they usually consider ap-

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-144-1

plication specific conditions in their implementation, turning their portability to different scenarios a difficult task. Hence, every time a new application comes at hand, a lot of work is involved in adapting these technics to the new scenarios. In this direction, this paper proposes a general framework for *sensor data stream reduction*. The framework takes into account the reduction methodology design, which shows how to reduce data as it is sensed or routed through the sink, and the reduced data validation process, which assesses if the reduced data is as representative as the original data.

The remainder of this article is organized as follows. We first describe the reduction design focusing in the architecture reduction. We show some data reduce validation that can applied. We present some specifics study case showing the efficiency of our methodology afterward. Finally, the open issues and conclusions are discussed.

II. Reduction Architecture Design

This section presents a general architecture for sensor stream reduction that can be easily applied to any WSN scenario aiming to reduce its energy consumption and data delay. The way the streams are reduced depends on the moment the reduction is going to be performed, i.e., during sensing or routing streams, and the type of data stream we are dealing with, i.e., the number of phenomena monitored by the stream generated by the sensor node.

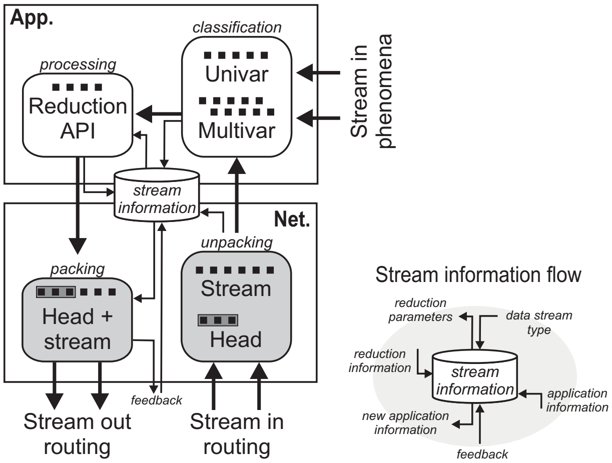


Figure 1. General architecture

The proposed methodology is illustrated in Fig. 1. As observed, the input streams can have been originated by the phenomena (sensing stream) or sent to the sensor node by another node (routing stream). The sensing reduction is recommended when the sensor device gets an excessive number of samples, and cannot be dynamically calibrated to deal with more data than it currently deals with. The routing reduction, in contrast, is performed when the network does not support the amount of data being transmitted. For instance, if the application has some requirements regarding the amount of data supported by each sensor node, data stream reduction can avoid uncontrolled data loss while guaranteeing the application requirements. Note that the sensing stream arrives in the application layer, while the routing stream arrives in the network layer.

When data arrives in the network layer, the network packets first need to be unpacked, separating the data stream from the header (that may contain some specific application information/restriction). Once it is unpacked, it is sent to the application layer, to be processed in the same way that sensing streams. At the same time, stream information is given to a cross-layer (labeled in Fig. 1 as “stream information”), responsible for making the interface between the application and network layers. The “stream information”, highlighted in Fig. 1, is responsible for choosing which reduction algorithm should be executed and its parameters. The information stored in this cross-layer includes:

- **Feedback**: Data received from other sensor nodes in order to perform the reduction calibration in an online fashion. For example, if the data reduction is dynamic, other nodes can inform the current node if more or less data can be propagated.
- **Application information**: Data received from the network layer when the stream is unpacked. Examples of application information are the deadline to stream delivery or global energy constraints.
- **Data stream type**: Data received from the application layer after the data stream is classified, and can be univariate or multivariate.
- **Reduction parameters**: Data given to the application layer in to guide it to perform the more appropriated reduction, considering the “application information” and the “data stream type”. Examples of reduction parameter is *Use a sampling algorithm with 50% reduction.*
- **Reduction information**: Data received from the application layer after the reduction. Examples of reduction information are the reduction level achieved or the reduced data size.
- **New application information**: Data given to the network layer for packing the reduced stream out. Examples of application information are the updated deadline for stream delivery, the new global energy constrains, or the new data stream size.

When data streams arrive to the application layer, they first have to classified according to the number of variables they monitor. In this context, data streams can be univariate or multivariate. Univariate streams are represented by a set of values read by a unique type of sensor, e.g., a sensor node that monitors only environmental temperature. On the other hand, multivariate streams are represented by a set of values coming from different sensors of the same sensor node, e.g., a node that monitors temperature, pressure and humidity simultaneously, or by a set of measurements coming from the same sensor type located in different sensor nodes, e.g., a node that processes data from different nodes monitoring only temperature. This classification is important because the data reduction process itself depends directly on the stream type.

After the stream type is known, we have to choose an appropriated stream reduction algorithm to effectively perform the reduction. There are various types of data stream reduction methods, such as online samples, histograms built, and sketches [3]. The reduction algorithms available in our API are depicted in Fig. 2, and explained below:

- **Random sampling**: This algorithm initially builds an histogram from the original stream. Then, for each histogram class, random elements are chosen to compose the reduced stream. The objective of this algorithm is to reduce the data keeping the class frequencies of the original histogram unchanged. It reduces the network energy consumption and delay by reducing the transmitted data while keeping its representativeness [6]. In Fig. 2(a), we show a "stream in" of 100 elements, from each 50% of its elements are randomly chosen to compose the "stream out".
- **Central sampling**: This algorithm is a variation of the random sampling. The main difference is that, instead of performing a random element choice, the central elements of the histogram classes are chosen to compose the reduced stream. In Fig. 2(b), we have a "stream in" of 100 elements, 50 of them are chosen considering the central histogram classes elements, generating the "stream out".
- **Sketch**: A data stream based algorithm that sketches data, reduce it through a data sketch, e.g., the minimum, maximum and average of a data or the data frequency. In our case, the sketch algorithm builds a histogram from the original stream, and uses the histogram class frequencies as the reduced data sketch. The sketch reduces the energy consumption and delay by keeping a constant transmission data rate, since the sketch size is fixed. After the histogram sketch arrives to the sink node, the data represented by the sketch can be artificially generated without loosing data quality. The only trouble in this strategy, when compared with the sampling, is that the sketch looses the sequence of data, despite of the good approximation when the original data is regenerated artificially [6]. In Fig. 2(c), we have a "stream in" with 100 elements, the histogram is built, and then a "stream out" is generated with the histogram class frequencies.
- **Multivariate sampling**: This algorithm uses principal component analyzes to help multivariate sampling. The principal components transformation is one of the most powerful tools for multivariate data treatment [7]. It is a transformation between γ-dimensional spaces, derived from the covariance matrix of the input data (in our case multivariate sensor data), generating a set of new data, where each resultant value is a linear combination of the original values. The number of principal components is equal to the number of dimensions of the original data and these principal components can be sorted according their variance. Thus, the first and the last principal component have the biggest and the smallest variance, respectively. In our algorithm the principal components of the original stream are computed. Then, the first component is sorted and used to rank the original stream. Based on this ranking and the data reduction size, the most correlated data are sampled [8]. In Fig. 2(d), we have a "stream in" with 100×5 elements (where each element represent data coming from different sensors in the same node or different sensor nodes), 50% of "stream in" is sampled considering the raking of first component analyzed, and then a "stream out" is generated with 50×5 elements.

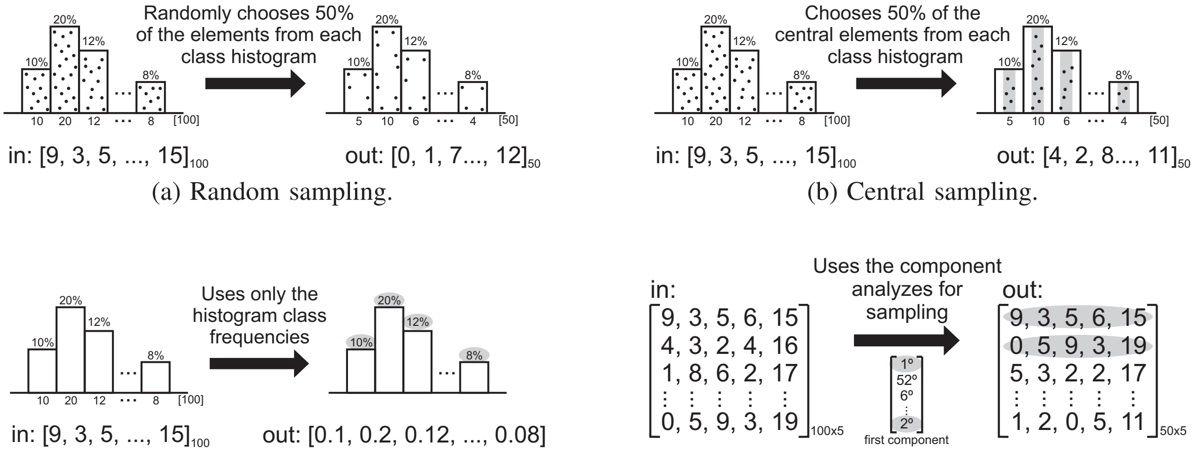
After the reduced data stream is obtained, if the stream was being routed, it is passed back to the network layer, which packs the stream and any information gathered from the cross-layer, and sends it to the sink.

III. Data reduced validation

After data stream reduction is performed, it is important to verify if the data quality of the original stream was preserved. If a sensor stream item with 100 elements is reduced for 50 elements, are these 50 elements representative?. Most of the time, data reduction validation is application specific. However, there are some simple tests that can be performed to validate the reduced data independent from the target application, considering the distribution of univariate streams, the variance analysis of multivariate streams, and the absolute relative error for both types of streams.

The distribution approximation between the original and reduced item streams can be done by the Kolmogorov-Smirnov test (KS test) [9]. This test evaluates if two univariate samples have similar distributions, and is not restricted to samples following a normal distribution. For example, if the original univariate stream item follows a Poisson distribution, this test verifies if the reduced item stream keeps the distribution characteristics. The analysis of variance (*ANOVA*) [10], used to validate multivariate data reduction, can be used to indicate if there is significant difference between the variances of the original and reduced multivariate streams.

It is also important to evaluate the discrepancy of the values in the reduced streams, i.e., if they still represent the original stream. Considering univariate data, this discrepancy can be quantified using the absolute value of the largest distance between the average of the original data and the lower or higher confidence interval values of the reduced


Randomly chooses 50% of the elements from each class histogram
20%
12%
10%
8%
10 20 12 ••• 8 [100]
5 10 6 ••• 4 [50]
in: [9, 3, 5, ..., 15]100
out: [0, 1, 7..., 12]50
Chooses 50% of the central elements from each class histogram
in: [9, 3, 5, ..., 15]100
out: [4, 2, 8..., 11]50
Uses only the histogram class frequencies
in: [9, 3, 5, ..., 15]100
out: [0.1, 0.2, 0.12, ..., 0.08]
Uses the component analyzes for sampling
in:
9, 3, 5, 6, 15
4, 3, 2, 4, 16
1, 8, 6, 2, 17
0, 5, 9, 3, 19
100x5
1º
52º
6º
2º
first component
out:
9, 3, 5, 6, 15
0, 5, 9, 3, 19
5, 3, 2, 2, 17
1, 2, 0, 5, 11
50x5

(a) Random sampling.

(b) Central sampling.

(c) Sketch.

(d) Multivariate sampling.

Figure 2. Architecture API.

data average [6]. This same process can be applied to multivariate data, but in this case the final error used is the largest error among all relative errors for each variable, i.e., the error is calculated for each sensor (or variable) and only the highest of them will be considered [8].

IV. Methodology case study

In order to illustrate the application of the methodology proposed, this section presents two problems and the simulations performed to solve them following the steps described in the previous section. The first problem considers a general sensor stream application where a scheduled reduction has to be performed in the source node, i.e. it receives data about a phenomena for a certain time and then sends it to the sink. The second problem addresses a real time application where the reduction is performed during routing. Fig. 3 shows how the problems are addressed in the phenomenon view and the routing architectures.

In both scenarios, we consider a flat network that uses a shortest path tree based routing algorithm. The network density is kept constant (8 neighbors per node), and all nodes have the same software and hardware configuration. The phenomena monitored is always the same, and is represented by a normal distribution. The evaluation is performed through simulations using the NS-2 (Network Simulator 2) version 2.33. Other default simulation parameters, such as like radio range and bandwidth, were kept as $50\,\text{m}$ and $250\,\text{kbs}$, respectively. Each simulated scenario was executed with 33 random topologies.

In the first scenario, the network has 128 nodes, with different numbers of nodes (1, 5, 10, and 20) generating 256 items ($n = 256$) every 60s. Note that in this scenario only the application layer of our architecture is considered (phenomenon view on Fig. 3). Once the data is sensed, it is classified as univariate, and this information is sent to the "stream information" module, which sets the reduction parameters selecting the central sampling reduction algorithm (Fig. 2(b)), considering a reduction of $n/2$ and $\log n$, where n is the size of the stream sensed, i.e., n temperature or pressure samples. After the reduction step, the reduced stream is routed to sink. Observe that feedback and reduction information are not considered in this design, because the application requires only the local reduction, i.e., the reduction in sensing moment.

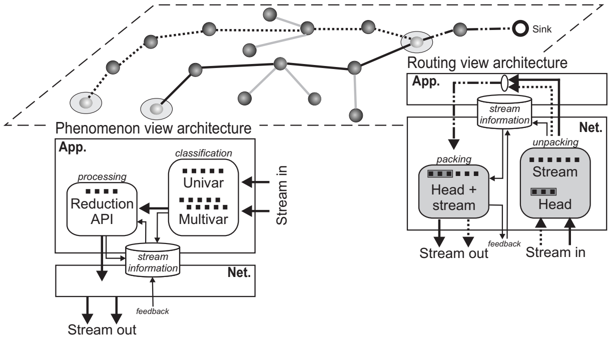
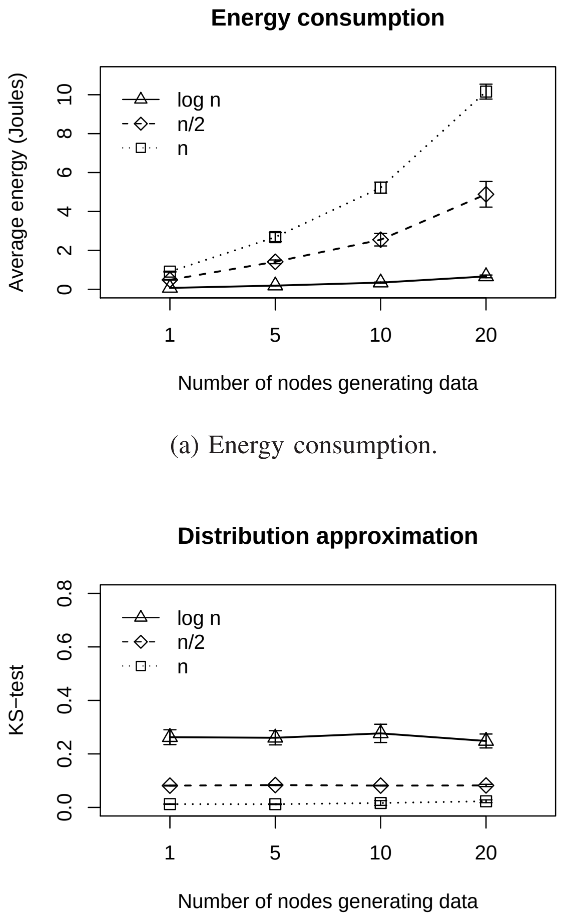


Figure 3. General architecture

Fig. 4 shows the average value of energy consumption and the difference in the original e reduced data distribution using the KS-test with a 95% confidence interval. As showed in Fig. 4, the sample $\log n$ reduces the energy consumption by reducing the transmitted data. However, the original data distribution is affected by 20%. This quality is acceptable by the large majority of applications when the network restrictions are strong. The sample of $n/2$ is interesting when the application does not have strong restrictions.

The second scenario simulated concerned a real time application, which works in both the application and network layers (phenomenon and routing view on Fig. 3). Again, only



(a) Energy consumption.

(b) Data validation.

Figure 4. Example of data reduction in a general sensor stream application.

univariate data is considered, and processed by the random and central sampling reduction algorithms, considering a reduction controlled by stream information. The stream items arrive in the network layer and is unpacked. The application information (packet head) is separated from the data application (stream) and it is passed to the stream information. However, the stream information receives the feedback of other nodes to reduce more or less data and the reduction parameters used. The packing receives from the stream information the new application information and can sent some feedback to other nodes.

The real time application operation considers: (i) again the stream item represents n samples of environment; (ii) the application has a soft deadline to deliver the stream item sensed, this deadline is packing with the stream item; (iii) the stream item is fragmented and routed through to sink; (iv) the router nodes look each packet and check if the stream item can be delivered; (v) if the deadline cannot be achieved the stream item is resembled (remember that it was fragmented) and reduced according the acceptable amount of data, e.g., the stream item with 256 elements cannot be delivered, so the router reduce it to 100 elements that can be delivered on time; and (vi) the stream reduced is repacking, now with the new application deadline, fragmented and forwarded through to sink.

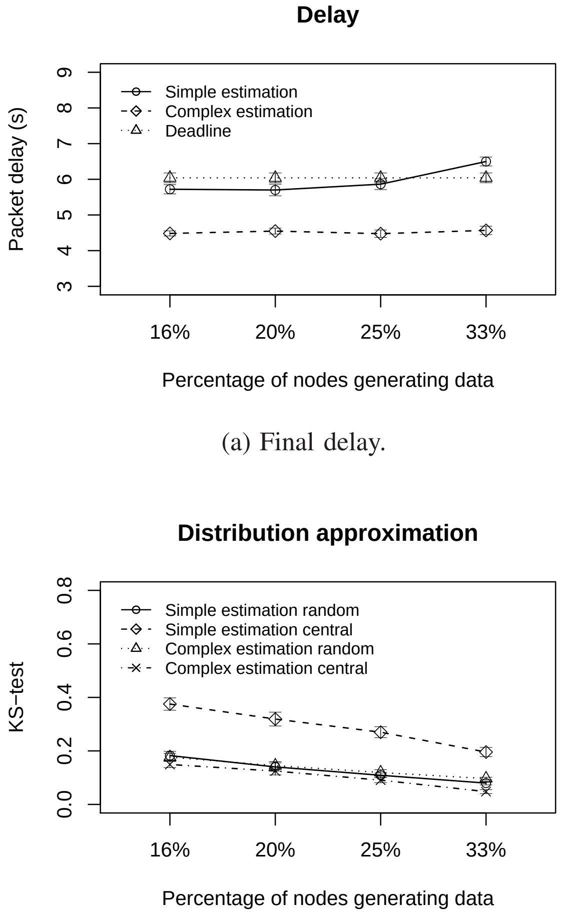
To illustrate the reduction solution in this real time applications we consider again a flat network that uses a shortest path tree based routing algorithm, the network density is kept constant (8 neighbors per node), and all nodes have the same software and hardware configuration. The phenomenon monitored is always the same, it is represented by a normal distribution. Like in general application, we perform our evaluation through simulations and use the NS-2 (Network Simulator 2) version 2.33. Each simulated scenario was executed with 33 random topologies. At the end, for each scenario we plot the average value with 95% of confidence interval. Other default simulation parameters are used, like radio range $50\,\mathrm{m}$ and bandwidth $250\,\mathrm{kbs}$.

To simulate the real time application we consider a minimum deadline (get empirical) that the “perfect network” supports. To force the in-network reduction we use concurrent traffic and all router nodes delay the packet fragments at 0.01% of the initial deadline. We set among the 128 nodes distributed in the network 16%, 20%, 25%, and 33% of this nodes generating extra traffic. However we stress the system to consider 2048 elements (n) in stream item. However, in order to highlighted the importance of the reduction controlled by stream information, we consider two estimation way: a simple that analyzes only the local node time; and a complex that tries to infer what happens during the data traffic.

In the Fig. 5(a) the application deadline is met in almost cases. The complex estimation presents a more scalable behavior considering the percentage of nodes generating data. This occurs because this estimation infers better the data traffic behavior during the routing. The Fig. 5(b) shows the simple and complex estimation using the random and central sample reduction algorithms. It is showed that in all cases we have a distribution approximation $\leq 40\%$. The central sampling algorithm with the complex estimation has a smaller error. The reason is that the complex estimation performs the maximum reduction sooner (the central algorithm is executed once or twice). This result shows that because fewer successive reductions are performed, more representativeness is kept in the reduced data, i. e., data degradation is mitigated.

V. Open issues and conclusions

This work presented a general methodology to perform sensor stream based reduction in WSNs. This methodology considered the application requirements, the reduction design, and the data reduce validation. Specifically, to reduction design was presented a architecture that can be applied



(a) Final delay.

(b) Data validating.

Figure 5. Reduction in real time stream applications.

to reduce the data when it is sensed or routed through to sink. The study cases showed the usefulness of our methodology applied on a general sensing and a real time applications. Furthermore, the methodology proposed is general enough to be applied to design reduction scenarios in which we have some application requirements.

Among some open issues, we can consider a better evaluation of the proposed methodology by considering other network scenarios, and matching the proposed application level solution with lower level ones. However, consider the architecture, not only the data from a source is reduced, but similar data from different sources can be also reduced, resulting in a more efficient reduction solution.

ACKNOWLEDGES

This work is partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under the grant number 477946/2010-0 and the Research Foundation of the State of Minas Gerais (FAPEMIG) under the grant number CEX-APQ-00577-09.

REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.

[2] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *13th IEEE Mediterranean Conference on Control and Automation (MED'05)*. Hawaii, USA: IEEE Computer Society, June 2005, pp. 719–724.

[3] S. Muthukrishnan, *Data Streams: Algorithms and Applications*. Hanover, MA, USA: Now Publishers Inc, January 2005.

[4] S. Santini and K. Romer, "An adaptive strategy for quality-based data reduction in wireless sensor networks," in *3rd International Conference on Networked Sensing Systems (INSS'06)*. Chicago, IL, USA: Transducer Research Foundation, 31 May – 2 June 2006, pp. 29–36.

[5] E. F. Nakamura, A. A. F. Loureiro, and A. C. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," *ACM Computing Surveys*, vol. 39, no. 3, pp. 9/1 – 9/55, April 2007.

[6] A. L. L. Aquino, C. M. S. Figueiredo, E. F. Nakamura, L. S. Buriol, A. A. F. Loureiro, A. O. Fernandes, and C. N. C. Junior, "Data stream based algorithms for wireless sensor network applications," in *21st IEEE International Conference on Advanced Information Networking and Applications (AINA'07)*. Niagara Falls, Canada: IEEE Computer Society, May 2007, pp. 869–876.

[7] J. E. Jackson, *A User's Guide to Principal Components*, 1st ed. Wiley-Interscience, 2003.

[8] O. S. Junior, A. L. L. Aquino, R. A. F. Mini, and C. M. S. Figueiredo, "Multivariate reduction in wireless sensors networks," in *IEEE Symposium On Computers and Communications (ISCC'09)*. Sousse, Tunisia: IEEE Computer Society, July 2009.

[9] S. Siegel and J. N. John Castellan, *Nonparametric Statistics for the Behavioral Sciences*, 2nd ed. Columbus, OH, USA: McGraw-Hill Humanities/Social Sciences/Languages, January 1988.

[10] N. Thomson, "Understanding ANOVA the APL way," *ACM SIGAPL – APL Quote Quad*, vol. 24, no. 1, pp. 295–303, August 1993.