An Efficient Data Transport Protocol for Event-Driven Field-Estimation on Sensor Networks

Daniel de O. Cunha[1,2], Otto Carlos M. B. Duarte[1], and Guy Pujolle[2]

[1]Grupo de Teleinformática e Automação (GTA)
Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ, Brazil

[2]Laboratoire d'Informatique de Paris 6 (LIP6)
Université Pierre et Marie Curie - Paris VI
Paris, France

***Abstract*— This paper introduces and analyzes an Event-Driven Field-Estimation (EDFE) protocol for wireless sensor networks. In our field estimation scheme the sensors derive their own notion of expected reading. Readings differing from the expected are considered events of interest and trigger a data transmission to the sink. The proposed protocol allows sensor nodes to only transmit the events of interest, suppressing the transmission of less important data and, as a consequence, saving energy. The EDFE protocol allows the sink to synchronize the received data and correctly reconstruct the sampled process. The proposed scheme is evaluated in a realistic scenario using real-site collected data and representative network conditions. It is shown that the EDFE protocol reduces up to 50% the number of packets sent to the sink while keeping low the average reconstruction error. Moreover, it is demonstrated how the parameters can be tuned to increase the robustness of the protocol to network losses.**

I. Introduction

Field estimation is an important application of wireless sensor networks because sensors are very close to the sensed process and are able to acquire detailed data in a non-invasive way [1]. This type of application deploys sensor nodes in a specific region to remotely sense space-temporally variable processes, such as temperature or UV exposure. The accuracy of the estimation depends on the spatial and temporal frequencies of sampling. These frequencies must be high enough to avoid the aliasing problem [2]. On the other hand, high frequencies generate a larger amount of data to transmit. As the data transmission is a consuming task for a sensor [3], it is interesting to trade communication for local processing.

The spatial sampling frequency is related to the number of nodes and their position. A higher spatial sampling frequency results in a larger number of nodes transmitting data to the sink. Some works aim at identifying nodes with similar readings and deactivate part of these nodes [4], [5]. The idea is to avoid the transmission of redundant information. This strategy is very similar to the data-aggregation approach. However, instead of allowing all nodes transmit and perform in-network aggregation, nodes whose data would be aggregated are identified in order to shutdown some of these nodes. Nevertheless, there are areas where the spatial variation of the process is sharp and neighbor nodes present very different measurements. In these areas, called *borders*, this kind of approach is not useful since many nodes must remain actives. Thus, it is necessary to reduce the transmission time of the nodes. The typical solution is the compression/codification of the resulting temporal series. Nevertheless, conventional compression algorithms are not suitable to wireless sensor nodes for they have limited resources [6]. Chen *et al.* [7] and Lazaridis *et al.* [8] propose efficient algorithms to reduce the amount of data sent. However, they do not show how to actually transport the resulting codified data in an efficient and robust way through the error-prone wireless environment.

We proposed and analyzed an event-driven field estimation scheme for wireless sensor networks [9]. Differing from the above discussed approaches, our scheme reduces the amount of transmitted data by only sending part of the samples. The assumption behind our scheme is that although we have to sample the process with its required temporal frequency to avoid losing important data, not all the samples will bring interesting information. Hence, the proposed scheme exploits specific features of the monitored processes in order to reduce the amount of data transmitted to the sink. Each sensor node collects the samples and decides to only send the ones considered an event of interest to itself. This mimics an event-driven system over a continuous-data transmission application. The proposed scheme is distributed and does not require synchronization between sensor nodes. Moreover, the proposed scheme is complementary to those which reduce the spatial frequence and to those based on data aggregation/fusion because it is capable of reducing the energy consumption in the border regions where these techniques have small utility.

In this paper we propose and analyze an Event-Driven Field-Estimation Protocol (EDFE Protocol), which carries the selected samples and allows the sink to identify the suppressed ones to correctly reconstruct the monitored process. The proposed protocol is evaluated in realistic scenarios developed using real-site collected data to represent the sensor readings and representative network conditions. The efficacy of the proposed protocol is evaluated through the number of transmitted packets and the average reconstruction error at the sink.

The rest of the paper is structured in the following way. In Section II, we propose the Event-Driven Field-Estimation Protocol. Then, the simulations are presented in Section III, showing the efficiency of the EDFE protocol. Finally, conclusions and future research work are discussed in Section IV.

II. The Field Estimation Scheme

The proposed field estimation scheme is composed of three different parts: the Data Processing, the Learning/Configuration, and the Networking components. The Data Processing component is responsible for analyzing the sensed data and deciding which samples must be sent to the sink. This component has been previously analyzed [9], [10]. The

This work has been supported by CNPq, CAPES, FAPERJ, FINEP, RNP and FUNTTEL.

Learning/Configuration component is responsible for tuning the other components to optimize the achieved gains based on the sensed data. It may be implemented to remotely run at the sink or in the sensor nodes theirselves when the node has enough computational/ storage power. It is also possible to combine the two approaches in heterogeneous networks, with less capable nodes being configured by the sink or by more resourceful self-configuring neighbors. The Learning/Configuration will be treated in a later work. The present work proposes and analyzes the Event-Driven Field-Estimation Protocol (EDFE Protocol), which is the networking component of the scheme.

A. Data Processing Component

In practice, the sensor identifies a recurrent pattern in the process and defines an expected behavior, or the periphery of attention [9], for the next readings. Based on this expected behavior, the sensor decides whether a sample is important or not. If the sample aggregates useful information, it sends the sample to the sink. These samples sent to ensure the quality of the estimation are called *refining samples*. Otherwise, the node economizes energy not transmitting it. The sample is taken into account to calculate the expected behavior to the future and then discarded.

The first step is to determine the periodicity of the recurrent pattern. Different approachs can be used as, for instance, data auto-correlation. Nevertheless, it is a function of the Learning/Configuration component and we will not discuss it further in this paper. The physical process used in our analysis is the temperature, which clearly has a daily periodicity. Thus, the sensor nodes must identify a daily-expected behavior, updated every day. The decision about refining samples is made based on this behavior. Therefore, the sink needs to know the expected behavior of the process, which is assumed to occur when no refining samples are received. The node must periodically send an updated expected behavior to the sink. The sink then assumes the process behaves exactly like the most recent expected-behavior vector, whenever no refining samples are received. These refining samples are used along with the expected behavior as informed by the sensor. The sensor node must decide on sending or not refining samples based on the last Expected Behavior Vector (EBV) sent to the sink. This procedure maintains the consistency between the measured and the reconstructed information. Therefore, the sensor verifies whether the measured value differs above a certain threshold from the corresponding sample of the last expected-behavior vector sent. If this difference is higher than the configured threshold, the sensor sends the refining sample.

```
DB i = α D i + (1−α) DB i−1
If update time
    last_update = DB i
    Send last_update
For all k samples in D i do
    If |D i(k) − last_update(k)| > |last_update(k)| * configured_error
        Send D i(k)
```

Fig. 1. Daily procedure.

Assuming that the daily periodicity is already known, Fig. 1 shows the daily procedure, where DB_j is the vector with the expected behavior during day j, D_i the vector with the measurements of day i, $last\_update$ is the vector with the most recent expected behavior sent to the sink, and the notation $X(k)$ is used to represent the k-th element of vector X.

As we can see in Fig. 1, the algorithm has three important parameters: the α factor, the $update$ specification, and the $configured\_error$ limitation. The α factor weights the process history importance in the generation of the expected-behavior vector. The $update$ parameter specifies the interval between the transmission of expected-behavior vectors to the sink. The $configured\_error$ is used to define what is an event for the sensor node. In this context, an event is any reading differing more than the tolerated error from the expected behavior. The $configured\_error$ parameter is also responsible for limiting the reconstruction error at the sink.

It is worth noting that we have chosen to define the expected behavior through a simple weighted moving average in order to allow the implementation of the proposed field estimation scheme in less powerful sensor nodes. The proposed algorithm performs $O(1)$ computation per sample. Although it is possible to enhance this estimation in more capable sensor nodes, it is out of the scope of this paper.

B. The Event-Driven Field-Estimation Protocol

Conventionally, field estimation is done by time stamping each transmitted sample [1]. This procedure provides accurate timing information which is used during the reconstruction. Unfortunately, this approach is too costly and incurs in high overhead. Another option, when sampling occurs at a regular interval, is to timestamp the first sample of the packet, inform the sampling interval and send the other samples in sequence. We will call this approach the Optimized Diffusion. Nevertheless, this approach does not allow the suppression of samples, since the sink will have no means to know when and how many samples have been suppressed.

The objective of the EDFE Protocol is not only to efficiently transmit the expected-behavior vectors and the refining samples selected by the data processing component but also to send enough timing information to correctly reconstruct the monitored process at the sink. It is common to adopt the use of two-byte samples and four-byte timestamps, which makes the insertion of more than one timestamp in a packet highly undesirable. Thus, the EDFE protocol is designed to use only one timestamp per packet. Moreover, we enhance the efficiency of the protocol by avoiding the use of fixed control fields in all the packets.

All the control information is included in the Expected Behavior Packets (EBP), which carry the samples of the expected behavior vectors. Due to the small-sized packet used in wireless sensor networks, an Expected Behavior Vector will be split in multiples EBPs. Refining samples may be piggybacked in the last EBP of each period to increase the efficiency of the protocol. The adoption of the piggybacking is left to the sensor node and is seamless to the sink.

The Refining Packets (RP), on the other hand, carry only refining samples and Suppression Bytes, which indicates the amount of samples suppressed between two refining samples. The differentiation between the two kinds of packets is done robbing the most significant bit of the 4-byte timestamp. This

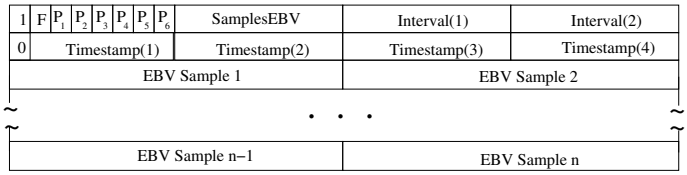


Fig. 2. Expected Behavior Packet.

bit is always *zero* for all valid timestamps in the system. If the first bit of a packet is *one*, it is a EBP and the first four bytes are control fields. On the contrary, if the first bit is *zero*, it is an RP and the first four bytes form a timestamp.

The expected behavior vectors are the most important data in our scheme. Thus, the EDFE protocol is designed to ensure the correct recognition and positioning of the expected behavior vectors during the reconstruction process. The expected behavior vector is always carried by EBPs. Its samples can not be piggybacked at the end of a RP. This is done to ensure that even when an EBP packet is lost, the sink will be able to position the expected behavior in the temporal series. There are two special fields in the EBP to allow this operation, one that informs the number of the present EBP in the period (bits P_1 to P_6 of the first byte) and another which specifies the number of samples in the expected behavior vector ($SamplesEBV$) and, as a consequence, in each period. The number of samples per expected behavior vector is used by the sink to calculate the total number of EBPs per period, since the maximum packet length is constant. Missing samples of an expected behavior are filled with the corresponding samples from the previous expected behavior vector during the reconstruction process. The EBP includes also a two-byte sampling interval ($Interval(1)$ and $Interval(2)$), which indicates the interval between consecutive samples as in the Optimized Diffusion. The next four bytes form the timestamp ($Timestamp(1)$ to $Timestamp(4)$) of the first sample in the EBP. The structure of such a packet is shown in Fig 2.

In the first byte of the EBP there is a special control bit (F), which indicates how a Suppression Byte is recognized in the following Refining Packets. This recognition is application dependent. The assumption here is that no natural process will vary between the two extremes values (maximum positive and minimum negative) possible. As a consequence, if the first two bits of what would be expected to be a refining sample are indeed an indication of the not achievable extreme, this indicates that this specific byte is a Suppression Byte. Fig. 3 shows the Suppression Byte structure.

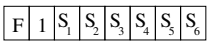


Fig. 3. Suppression Byte.

If the sensor is reading closer to the positive extreme, the flag F is set to *one*. Thus, as the second bit of the suppression byte is always one, this would represent an not-achievable extremely low reading and the Suppression Byte can be recognized. On the other hand, if the sensor is reading closer to the negative extreme, the flag is set to *zero*, forcing the recognition of the Suppression Byte through an extreme positive value. The bits S_1 to S_6 in the Suppression byte indicates the number of consecutive suppressed samples. If more than 64 samples are suppressed consecutively, the information is divided in two consecutive Suppression Bytes. This application-specific flag is set by the sensor node while sending the EBPs.

The Refining Packets start with a 4-byte timestamp of the first sample. The RP first sample is never a Suppression Byte. Since the non reception of a sample implies its replacement by the correspondent expected behavior vector sample, sending this byte would only incurs unnecessary overhead. After the first sample, the valid samples and Suppression Bytes can be distributed with no restriction, as exemplified in Fig. 4.

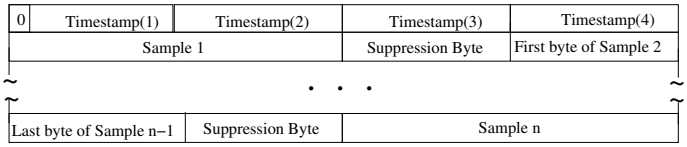


Fig. 4. Refining Packet.

At the sink, valid samples and suppressed ones are positioned for reconstruction based on the timestamp of the first valid sample and the interval between samples informed in the EBP. In order to simplify the protocol and avoid wasting energy with retransmissions, the protocol provides an unconfirmed service. Thus, except for the expected behavior vector, the protocol can not distinguish suppressed samples from samples lost due to network error. Nevertheless, the datagram nature of the sensor networks makes this limitation not important, since samples would not be resent. The reception of the EBPs acts like a heartbeat to know if the sensor is alive or not. The simulations are detailed in the next section.

III. Simulations

We analyze the proposed scheme by simulating the local processing of one sensor node, the networking component (EDFE Protocol), and the network losses. We use the tool GNU Octave [11] to perform the simulations. The EDFE protocol is evaluated considering two distinct metrics: the total number of packets sent to the sink and the average reconstruction error at the sink. We use the number of packets sent as an index to estimate the energy consumption. This preserves the generalization of the results by avoiding specific MAC-layer biases. The lower the fraction of packets sent the better. The average reconstruction error is used to evaluate how well the scheme performs the estimation and is calculated as

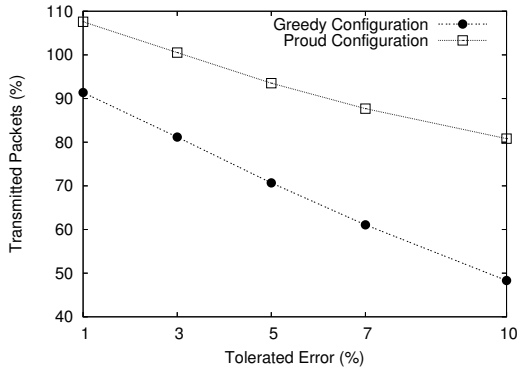
$$AE = \frac{\sum_{i=1}^{N} \frac{sample(i)' - sample(i)}{sample(i)} \cdot 100}{N}, \qquad (1)$$

where $sample(i)$ is the value sensed by the sensor for a given sample, $sample(i)'$ is the reconstructed value at the sink, and N is the total number of samples collected.

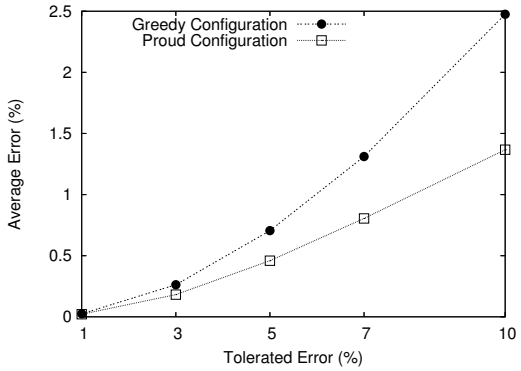
The EDFE protocol is compared to the Optimized Diffusion described in Section II-B. Whitout network losses, the Optimized diffusion has no error and is more efficient than the conventional diffusion. We analyze the performance of the proposed scheme in a temperature monitoring application based on the measurements of a Brazilian meteorological station. The results are shown with a 99% confidence interval.

A. Simulation Scenarios

Based on results in the literature which shows the significant impact of the data input in the evaluation of algorithms for sensor networks [12], we have decided to use real-site



(a) Transmitted packets.



(b) Reconstruction error without network losses.

Fig. 5. Ideal configurations.

collected data in our analysis. The field-estimation scheme is analyzed based on data available at the web page of the Department of Basic Sciences of the Universidade de São Paulo, Brazil [13]. The web-site maintains a history of meteorological data from the last eight years. The temperature measurements used in this paper presents the evolution of the temperature at an interval of 15 minutes, which results in 96 samples per day. The measurements of each day were arranged in a single vector with 96 elements. These daily-measurements vectors were concatenated to form a large vector with all the measurements available from the last eight years.

The resulting data is used to simulate the input of a node and the network is modeled through the packet error rate (PER) model. In order to develop a realistic scenario, we vary the PER in the range observed in a real implementation of an habitat monitoring application [1]. By using this range of PER we expect to take into account real network conditions and the behavior of the lower layers. We are convinced that the results are more representative than the simulation of a large scale network with data generated by simple parametric models.

B. Results

In order to better analyze the proposed protocol, Fig 5 shows the results of two ideal configurations. The first one is the Greedy configuration, which aims to minimize the number of packets sent to the sink, while the second one is the Proud configuration, which minimizes the reconstruction error. The parameters of these configurations are shown in Table I.

TABLE I
IDEAL CONFIGURATIONS.

Tolerated Error	Greedy		Proud	
	Update	α	Update	α
1%	80	1	2	1
3%	60	0.1	2	1
5%	60	0.1	2	1
7%	60	0.1	2	1
10%	60	0.1	2	1

As Fig. 5(a) shows, the Greedy configuration results in a percentage of packets sent between 16% and 31% lower than the Proud configuration. Moreover, for very strict tolerated errors (smallers than 5%) the Proud configuration results in the transmission of more than the original data. This happens because, as seen in Table I, the Proud configuration uses a small parameter $update$ and α equals to 1 to reduce the reconstruction error. Observing Fig. 5(b), it is possible to see that, indeed, the Proud configuration results in a lower reconstruction error than the Greedy configuration. Nevertheless, our goal is to reduce as much as possible the number of packets sent. The high percentage of packets sent presented by the Proud configuration is not suitable to our application. However, it might be interesting to obtain a hybrid configuration which takes the two metrics into account. It is also worth noting that the Greedy configuration presents a gain of almost 10% while keeping an average reconstruction error of less than 0.025%. The only drawback of the Greedy configuration is the relatively quick growth of the reconstruction error when the tolerated error increases. Although the increase of the reconstruction error is not desirable, the reduction in the number of packets of more than 50% is significant. Furthermore, this relative elevation of reconstruction error becomes less remarkable when we consider the networks losses. Fig. 6 shows the reconstruction error at the sink for different packet error rates when the Greedy configuration is used. Fig. 6 also shows the reconstruction error of the Optimized Diffusion, which transmits all the samples. Optimized Diffusion sends up to 100% more packets than the Greedy configuration, as seen in Fig 5(a). During reconstruction for the Optimized Diffusion, lost samples are replaced by linear interpolations of the received samples.

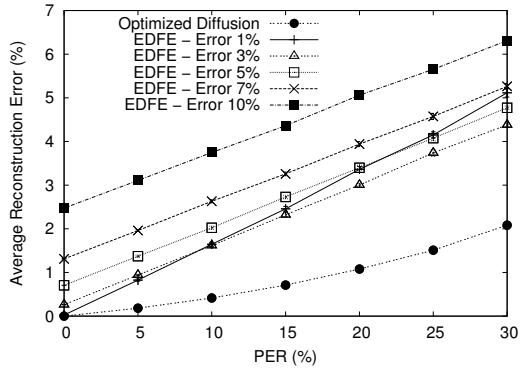


Fig. 6. Network effect over Greedy Configuration.

As it is shown in Fig. 6, the difference between the use of the EDFE protocol and the ideal case reduces as we consider realistic PER values. This occurs because the error caused by the network losses begins to dominate the final reconstruction

error. Moreover, when the proposed scheme is used, this same effect makes the difference between the results for different tolerated errors relatively smaller. This suggests that for higher PER it is more interesting to use a higher tolerated error, which results in a larger gain in the percentage of packets sent. The inclinations of the curves are robustness indicatives of each configuration. Thus, the Greedy configuration for a tolerated error equal to 1 is less robust to packet losses than the configuration for the others tolerated errors. As Table I shows, this configuration uses a very high $update$ parameter and α equal to 1 while all the other configurations use an intermediate to high $update$ parameter and a low α. In order to see how the configuration of the Data Treatment component affects the robustness of the EDFE protocol, we analyzed variations of the Greedy configuration. Table II summarizes these configurations and Fig. 7 shows the results.

TABLE II
DIFFERENT CONFIGURATIONS FOR TOLERATED ERROR = 1%.

Configuration	α	Update	Packets Sent (%)	Initial AE
Greedy	1	80	91.37	0.02484
Error Aware	1	15	93.65	0.02295
Greedy with $\alpha \neq 1$	0.1	80	91.75	0.02842
Restricted Greedy 1	0.85	20	95.43	0.03587
Restricted Greedy 2	0.1	20	96.67	0.02977

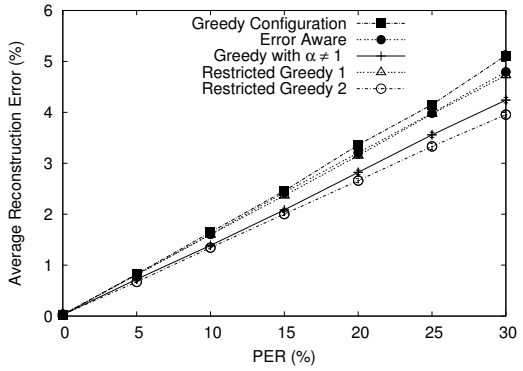


Fig. 7. Robustness of different configurations for tolerated error = 1%.

As Fig. 7 shows, the two parameters, α and $update$, impact on the robustness of the protocol. Lower α values imply more importance to the history of the process in the expected behavior calculation, showed in Fig. 1, and the results show that this increases the robustness of the protocol. Concerning the $update$ parameter, a higher value means less redundancy and, as a consequence, lower robustness. When we compare the effect of the two parameters, the variation of α has more impact on the robustness. The initial Average Error, or the error without network losses, does not have direct relation with the observed robustness. Similar analyses were performed for other tolerated errors showing similar results. These results are not shown here due to lack of space.

IV. CONCLUSION

This paper introduces and analyzes the Event-Driven Field-Estimation (EDFE) protocol. The EDFE protocol is part of a field-estimation scheme that exploits the fact that not all the collected samples result in useful information. We reduce the number of samples sent to the sink and, as a consequence, the energy consumption in the network. The EDFE protocol allows the sink to correctly reconstruct the monitored process after the suppression of the less important samples.

The EDFE Protocol is analyzed in realistic scenarios with real-site collected data and representative network conditions. The protocol is evaluated based on two metrics: the number of packets sent to the sink and the average reconstruction error. It is shown that due to the protocol design it is possible to tune the protocol to enhance the gain obtained by the Data Treatment component of the field estimation scheme. In the analyzed situations it is possible to reduce up to 50% the number of packets sent. Even with a very strict error limitation the proposed scheme results in almost 10% reduction.

The results of the robustness analyses show that with realistic packet error rates the error due to the network losses begins to dominate the reconstruction error. This implies that further gains in the percentage of packets sent may be achieved without a proportional increase in the reconstruction error. It is shown that the weight of the history of the process in the calculation of the expected behavior and the frequency with which this expected behavior is informed to the sink must be used to tune the robustness of the EDFE protocol.

In the future, we intend to develop an adaptive configuration mechanism to ensure the achievement of the best possible results, taking into account the desired gain and robustness.

REFERENCES

[1] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler, "Lessons from a sensor network expedition," in *1st European Workshop on Wireless Sensor Networks - EWSN'04*, jan 2004.

[2] A. Kumar, P. Ishwar, and K. Ramchandran, "On distributed sampling of smooth non-bandlimited fields," in *Information Processing In Sensor Networks - IPSN'04*, apr 2004, pp. 89–98.

[3] G. P. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, may 2000.

[4] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Information Processing In Sensor Networks - IPSN'04*, apr 2004, pp. 124 – 133.

[5] M. Rahimi, R. Pon, W. J. Kaiser, G. S. Sukhatme, D. Estrin, and M. Sirivastava, "Adaptive sampling for environmental robotics," in *IEEE International Conference on Robotics & Automation*, apr 2004, pp. 3537–3544.

[6] N. Kimura and S. Latifi, "A survey on data compression in wireless sensor networks," in *Internation Conference on Information Technology: Coding and Computing (ITCC'05)*, 2005.

[7] H. Chen, J. Li, and P. Mohapatra, "Race: Time series compression with rate adaptivity and error bound for sensor networks," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems - MASS 2004*, oct 2004.

[8] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *International Conference on Data Engineering (ICDE'03)*, mar 2003.

[9] D. O. Cunha, R. P. Laufer, I. M. Moraes, M. D. D. Bicudo, P. B. Velloso, and O. C. M. B. Duarte, "A bio-inspired field estimation scheme for wireless sensor networks," *Annals of Telecommunications*, vol. 60, no. 7-8, 2005.

[10] D. O. Cunha, O. C. M. B. Duarte, and G. Pujolle, "Event-driven field estimation for wireless sensor networks," in *IFIP/IEEE International Conference on Mobile and Wireless Communications Networks - MWCN 2006*, aug 2006.

[11] J. W. Eaton, *GNU Octave On-Line Manual*, 1997, http://www.gnu.org/software/octave/doc/interpreter/ - visited in Sep 2006.

[12] Y. Yu, D. Estrin, M. Rahimi, and R. Govindan, "Using more realistic data models to evaluate sensor network data processing algorithms," in *29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, 2004, pp. 569–570.

[13] Universidade de São Paulo, *Departamento de Ciências Exatas*, LCE - ESALQ - USP, 2005, http://www.lce.esalq.usp.br/indexn.html - visited in Feb. 2005.