

A Sampling Data Stream Algorithm For Wireless Sensor Networks

André L.L. de Aquino¹, Carlos M.S. Figueiredo^{1,2}, Eduardo F. Nakamura^{1,2}

Luciana S. Buriol³, Antonio A.F. Loureiro¹, Antônio Otávio Fernandes¹, Claudionor J.N. Coelho Jr.¹

¹ Department of Computer Science
Federal University of Minas Gerais
Belo Horizonte, MG, Brazil

Email: {alla, mauricio, nakamura, loureiro, otavio, coelho}@dcc.ufmg.br

² FUCAPI – Research and Technological Innovation Center
Manaus, AM, Brazil

³ Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, RS, Brazil
Email: buriol@inf.ufrgs.br

Abstract—This work presents a sampling data stream algorithm for wireless sensor networks (WSNs). The proposed algorithm is based on sampling techniques applied to data histograms created from original data streams acquired by sensor nodes. As a result, the algorithm provides a sample of only $\log n$ items to represent the original data of n elements. We show that by using our algorithm, we can save energy and reduce delay in WSN applications in different scenarios while keeping a good data quality.

I. INTRODUCTION

A wireless sensor network (WSN) is a special kind of ad-hoc network that has the capacity of sensing and processing, with can be applied to many fields, such as environmental, industrial, military, and agriculture [1]. Basically, these networks are comprised of compact and autonomous devices, called sensor nodes. These sensor nodes are composed of four basic components: sensing unit, processing unit, transducer, and energy source. There are two main types of applications for sensor networks: monitoring and actuating applications. In monitoring applications, the nodes only processes the data. In actuating applications, the nodes can interfere in the monitored environment [2], [3].

For WSN applications, we must consider the characteristics of sensor networks that are related to their components, topology, and how data is collected. In this context, we can distinguish some networks. A flat network has no clustering, a homogeneous network has all nodes with the same hardware components, and programmed networks send data following a schedule previously established by the network manager [4]. Despite its potential applicability, a sensor network has several resource restrictions, such as low computational power,

reduced bandwidth, and limited energy source. Therefore, algorithms for WSNs need to be carefully designed.

In some applications for sensor networks, the data usually arrives in an online fashion, is unlimited and there is no order in the arrival of data to be processed. Data with this characteristic are called *data streams*. Due to the constraints listed above, such as limited bandwidth and radio range, sending a large amount of data can take a lot of time when sensor nodes compete to access the wireless medium in a multihop data communication towards the sink. In this sense, this work applies strategies over generated data streams to reduce network traffic since it affects the data quality.

This work proposes a data stream algorithm for WSNs that uses a sampling technique on a histogram of data to reduce data traffic and, consequently, the delay and energy consumption. This work presents a way to deal with energy and time constraints at the application level, as a complementary view of solutions that treat this problem in the lower network levels. The algorithm aims to choose the ideal sample size for processing data streams.

This work is organized as follows. In Section II, we present a few studies about data stream in WSNs applications. In Section III, we introduce the data stream problem. Next, in Section IV, we present the data stream algorithm for summarizing the sensor network data. Experimental results are given in Section V, and Section VI concludes this study and presents the future work.

II. RELATED WORK

The most common data stream applications for sensor networks consider the network as a distributed database. In this

case, the network abstraction is based on a Data Stream Management System (DSMS). These applications are concerned with how queries can be answered [5]–[8]. Some proposals use the amount of resources available at a DSMS and apply it to extract management information from the sensor network, such as energy and node location [9], [10]. However, current DSMS's are not suitable for sensor networks, since nodes have too few resources.

Applications in data stream algorithms try to establish their lower bounds. The main metrics analyzed are time and communication complexities [11]–[13]. There are proposals that present specific data stream applications that are modeled using data stream algorithms. For example, finding the rarity and similarity in a data stream or counting the triangulation in a Web graph [14]–[17]. Indyk [18] proposes a data stream algorithm (implemented by Zhao [19]) that uses a family of hash functions called *min-wise* [20] to compute properties in data streams. This algorithm uses $O(\log n)$ bits to represent a hash index. The Indyk's algorithm computes a δ - *error* and an ϵ - *approximation* for the index found.

III. PROBLEM DEFINITION

The problem addressed in this work can be stated as follows:

Problem Statement: Given a set of data provided by source nodes, we want to meet WSN requirements by reducing the data traffic by using data stream techniques and assuring a minimum data quality in such a way that energy consumption and delay are reduced.

This problem can be further assessed by answering the following questions:

- **Data quality:** How can we evaluate the quality of the processed data? In some applications, the main goal of a WSN is to deliver sensed data to an observer. Due to the network limitations and the data characteristics only samples of the data stream are sent. Thus, we must evaluate if these samples are representative. To perform this evaluation we use statistics tests to know whether the original stream data and the sampled one are equivalent, and also compare the distance between the average of their data values.
- **Data reduction:** How much data can be reduced without compromising the application objectives? This question is associated with the data quality. To answer it, we need to identify the minimum data sample that can be used. In this sense, we use a sample of $\log n$ elements to represent a population of n elements while maintaining the data quality. Other sample sizes can be used according to the application requirements.
- **Losses vs. benefits:** What is the relation among the data-quality loss and the benefits for attending network requirements? By reducing the stream size using sampling there is an impact on the data quality, which is an important aspect for the applications. However, the higher the data is reduced the more benefits are achieved for network aspects such as delay and energy. The decision about which aspect is more important depends on the

application requirements, and so the evaluation of this relation is important. In this work, we show the behavior of our algorithm comparing these aspects.

All these questions must be answered to conceive any solution to data stream problems in WSNs. To address these answers, the scope of this work consider the following assumptions:

- **Sensor network topology:** We consider a flat network composed of homogeneous sensor nodes with a single sink to receive and process data from source nodes. We use a common tree-based routing solution to evaluate the network behavior. The data evaluation is computed when data arrive in the sink.
- **Data stream processing:** The streams are processed only by the source nodes, i.e., each source processes its own data stream and sends the results towards the sink node.
- **Data stream generation:** The streams are generated continuously at regular intervals (periods) of time and follow a normal distribution to represent their values.

IV. SAMPLING BASED ALGORITHM

To address the problem stated above, we need to design an algorithm that reduces the stream size. This reduction must keep the similarity compared to the original data, and also attend the network requirements. The proposal algorithm provides a solution to allow the balance between best data quality and network requirements. This algorithm tries to use the best knowledge about the generated data by getting a proportional sample to the data distribution. The sample size can vary, but it must be representative to attend the data similarity requirement. According to network requirements, we can set the sample size between $\log n$ and n . Thus, it can attend the quality requirements in relation to network requirements.

The proposed algorithm is based on sampling techniques using histograms, and it can be divided into the following steps:

Step 1: Build a histogram of the original data.

Step 2: Create a sample based on the histogram obtained in Step 1. To create such a sample, we randomly choose the elements of each histogram class, respecting the sample size and the class frequencies of the histogram. Thus, the resulting sample will be represented by the same histogram.

Step 3: Sort the data sample according to its order in the original data.

The pseudo-code of the algorithm is given in Fig. 1. We also consider n as the number of elements in the original data stream, and m as the adopted sample size.

Analyzing the algorithm in Fig. 1 we have:

- Line 1 executes in $O(n \log n)$.
- Lines 8–13 define the inner loop that determines the number of elements at each histogram class of the resulting sample, which takes $O(m)$ steps.
- Lines 5–18 define the outer loop in which the input data is read and the sample elements are chosen. Because the inner loop is executed only when condition in line

Require:

Vector $_dataIn$;	{original data stream}
m ;	{sample size}
$_numHist$;	{number of histograms}

Ensure: $_dataOut$; {sample stream}

- 1: Sort $_dataIn$;
- 2: $histScale \leftarrow$ “Class width”;
- 3: $first \leftarrow _dataIn[0]$;
- 4: $count \leftarrow 0, j \leftarrow 0$;
- 5: **for** $i \leftarrow 0$ **to** n **do**
- 6: **if** $(_dataIn[i] > first + histScale)$ **or** $(i = n - 1)$ **then**
- 7: $colFreq \leftarrow m \times count / _dataInSize$;
- 8: **while** $colFreq > 0$ **do**
- 9: $index \leftarrow$ “random element in the histogram class”;
- 10: $_dataOut[j] \leftarrow _dataIn[index]$;
- 11: $j \leftarrow j + 1$;
- 12: $colFreq \leftarrow colFreq - 1$;
- 13: **end while**
- 14: $count \leftarrow 0$;
- 15: $first \leftarrow _dataIn[i]$;
- 16: **end if**
- 17: $count \leftarrow count + 1$;
- 18: **end for**
- 19: Re-sort $_dataOut$; {according to the original order}

Fig. 1. Pseudo-code of the sampling algorithm.

6 is satisfied, the overall complexity of the outer loop is $O(n) + O(m) = O(n+m)$. We have an interleaved execution. Consider $numClass$ the number of histogram classes, $colOrig_i$ and $colSample_i$, respectively, the columns in original and sampled histograms, where $0 < i \leq numClass$. Basically, before entering in condition of line 6, $colOrig_i$ is counted and $n/numClass$ interactions are executed. Satisfying this condition $colSample_i$ is built and $m/numClass$ interactions are executed (loop 8–13). In order to build the complete histogram, we must cover all classes ($numClass$), then we have $numClass \left(\frac{n+m}{numClass} \right) = n + m$.

- Line 19 re-sorts the sample in $O(m \log m)$.

Thus, the overall complexity is $O(n \log n) + O(n + m) + O(m \log m) = O(n \log n)$, since $m \leq n$. The space complexity is $O(n + m) = O(n)$ because we store the original data stream and the resulting sample. Since every source node sends its sample stream towards the sink, the communication complexity is $O(mD)$, where D is the largest route in the network.

V. EVALUATION

When we apply sample algorithms over data streams, data quality decreases. Thus, to meet the application requirements we must analyze this impact, especially if shorter samples are used. The error between the original stream and the sampled one can be related to the distribution approximation and the

discrepancy of their values.

A. Methodology

In order to evaluate the distribution approximation between the original and sampled streams, we use the Kolmogorov-Smirnov test (K-S test) [21]. This test evaluates if two samples have similar distributions, and it is not restricted to samples following a normal distribution. Moreover, as the K-S test only identifies if the sample distributions are similar, it is also important to evaluate the discrepancy of the values in the sampled streams, i.e., if they still represent the original stream. To quantify this discrepancy (*Data Error*) we compute the absolute value of the largest distance between the average of the original data and the lower or higher confidence interval values (95%) of the sampled data average, $Data\ Error = \text{Max}\{|lower_{value} - Generate_{avg}|, |higher_{value} - Generate_{avg}|\}$, where the pair $(lower_{value}; higher_{value})$ is the confidence interval of data sample and $Generate_{avg}$ is the average of original data.

The evaluation of the algorithm is based on the following assumptions:

- **Simulation:** we perform our evaluation through simulations and use the NS-2 (Network Simulator 2) version 2.29. Each simulated scenario was executed with 33 random scenes. At the end, for each scenario we plot the average value with 95% of confidence interval.
- **Network topology:** we used a tree-based routing algorithm called EF-Tree [22], [23], the density is controlled and all nodes have the same hardware configuration. To analyze only the application, the tree is built just once before the traffic starts.
- **Stream generation:** the streams used by the nodes are always the same, following a normal distribution, where the values are between $[0.0; 1.0]$, and the periodicity of generation is 60s. The size of the data packet is 20 bytes. For larger samples, these packets are fragmented by the sources and re-assembled at the reception.
- **Evaluated parameters and stream size:** we varied the number of nodes, stream size, and number of nodes generating data. For each evaluated parameter we analyzed the application and network behavior by using sample size of n , $n/2$ and $\log n$. All parameters used in the simulations are presented in Table I.

We evaluated the algorithm by considering two parts: evaluation of data quality and evaluation of network behavior.

B. Results

Initially, we present the impact of our solution by evaluating data quality through the K-S test and the average error. In this evaluation, we use different sample sizes ($\log n$, $n/2$, and n) in different network scenarios. We vary the network size, the amount of data generated at the source, and the number of sources.

Figs. 2, 3, and 4 show the similarity between the original and sampled stream distributions. The difference between them we call *ks-diff*. The results show that when the sample size is

TABLE I
SIMULATION PARAMETERS.

Parameter	Values
Network size	Varied with density
Queue size	Varied with stream size
Sink location	0, 0
Source location	Random
Number of nodes	128, 256, 512, 1024
Radio range (m)	50
Bandwidth (kbps)	250
Simulation time (s)	5000
Traffic start (s)	1000
Traffic end (s)	4000
Stream periodicity (s)	60
Number of sources	1, 5, 10, 20
Stream size (n)	128, 256, 512, 1024
Sample size	$\log n$, $n/2$, n

diminished the ks -diff increases. Because the data streams are generated between $[0.0; 1.0]$, ks -diff = 20% for $\log n$ sample size, and ks -diff = 10% to $n/2$ sample size.

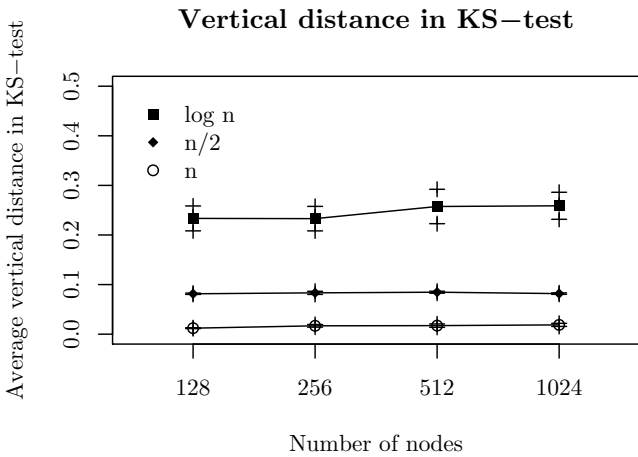


Fig. 2. K-S distance in different network sizes.

We also evaluate the data quality through the discrepancy between the original and sampled stream average values (Figs. 5, 6, and 7). This error we call $data$ -error. Like ks -diff, when the sample size is diminished the $data$ -error increases. However, $data$ -error = 10% for $sample = \log n$, and $data$ -error is almost zero for $sample = n/2$.

Next, we show the impact of our solution by evaluating the network behavior. This evaluation considers the total consumed energy of the network and the average delay to delivery a data packet to the sink. Another analyzed metric, not shown here, was the packet delivery ratio, and in all cases it was around 100% of delivered data. Again, in this evaluation, we use different sample sizes ($\log n$, $n/2$, and n) with different

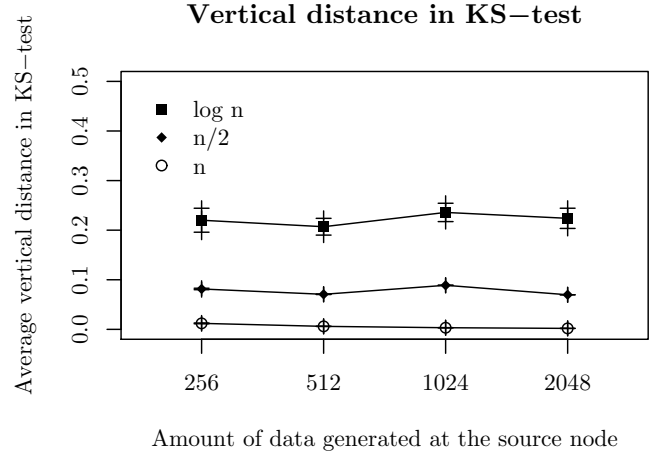


Fig. 3. K-S distance with different stream sizes.

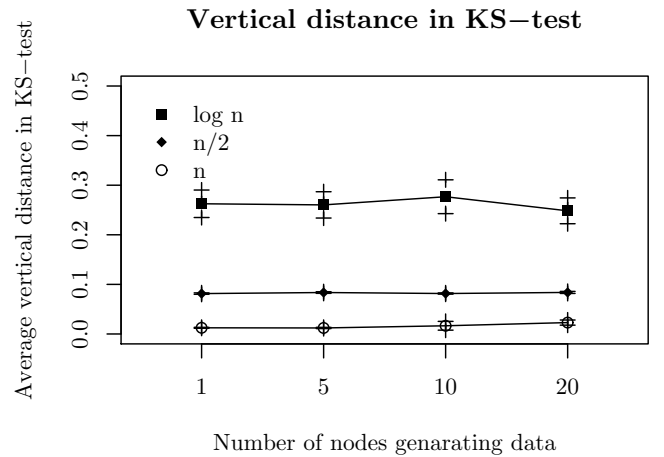


Fig. 4. K-S distance with different number of sources.

network scenarios by varying the network size, the amount of generated data at the source, and the number of sources.

Figs. 8, 9, and 10 show the energy consumption performance. The results show that when sample size is diminished the consumed energy is diminished too. When the number of nodes is varied (Fig. 8) the consumed energy does not vary. This occurs because only one source is used and the stream size is the same. When the sample size and the number of nodes generating data are varied, we can observe the impact of our solution in the energy consumption. Again, $sample = \log n$ has the best performance in all cases.

Figs. 11, 12, and 13 show the delay performance. Like the energy results, we can see that when sample size is diminished, the delay is diminished too. Again, the same effect of the number of nodes variation is observed (Fig. 11). When the sample size and number of nodes generating data are varied we can observe the delay impact by using our solution. In the

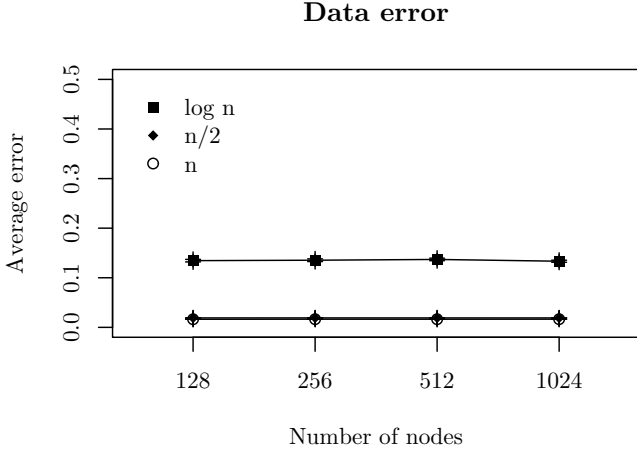


Fig. 5. Average error with different network sizes.

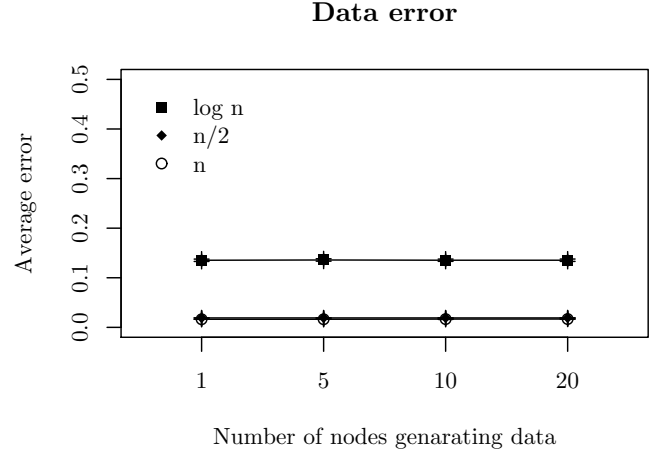


Fig. 7. Average error with different number of sources.

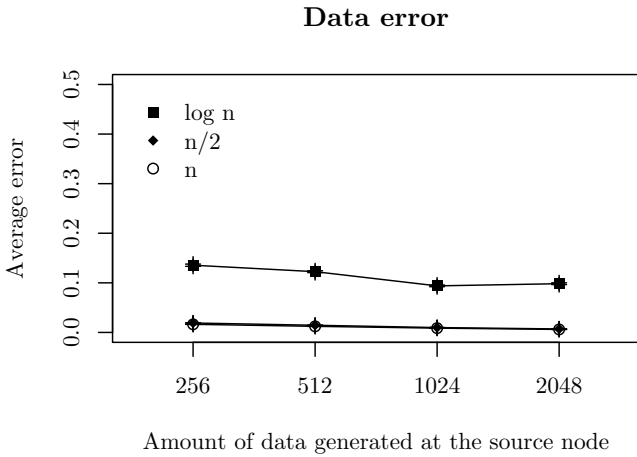


Fig. 6. Average error with different stream sizes.

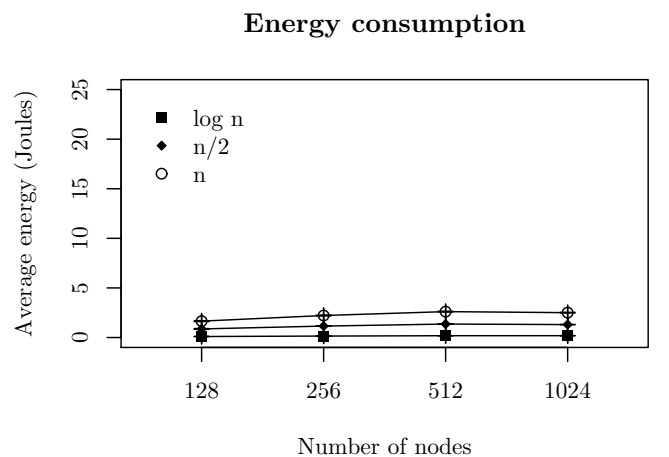


Fig. 8. Total consumed energy with different network sizes.

all cases, the $sample = \log n$ has the best performance.

In summary, when we analyze the data quality against the network behavior, we have the following conclusions:

- Our algorithm with $sample = \log n$ reduces the consumed energy and delay by reducing transmitted data. However, the data quality is affected in the distribution similarity (20%) and average discrepancy (10%). But this quality may be acceptable by a large majority of applications when the network restrictions are strong.
- Our algorithm with $sample = n/2$ is interesting either when the application priority is the average discrepancy (near zero), or we have the scenario presented in Fig. 8, in which the stream size and number of nodes generating data do not vary.
- Not using our algorithm, i.e. results with $sample = n$, is interesting when we have to keep the same data quality similarity and we do not have to worry about the WSN

restrictions.

Finally, our solution can be applied in the problem addressed in Section III, and the results answer the questions *Data quality*, *Data reduction*, and *Losses vs. benefits* presented in Section III.

VI. CONCLUSION AND FUTURE WORK

Wireless sensor networks are energy constrained, and the extension of their lifetime is one of the most important issues in the design of such networks. Usually, these networks collect a large amount of data from the environment. In contrast to the conventional remote sensing — based on satellites that collect large images, sound files, or specific scientific data — sensor networks tend to generate a large amount of sequential small and tuple-oriented data from several nodes, which constitutes data streams.

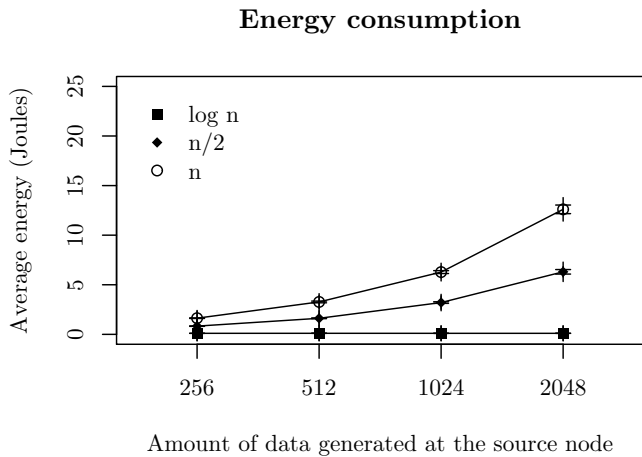


Fig. 9. Total consumed energy with different stream sizes.

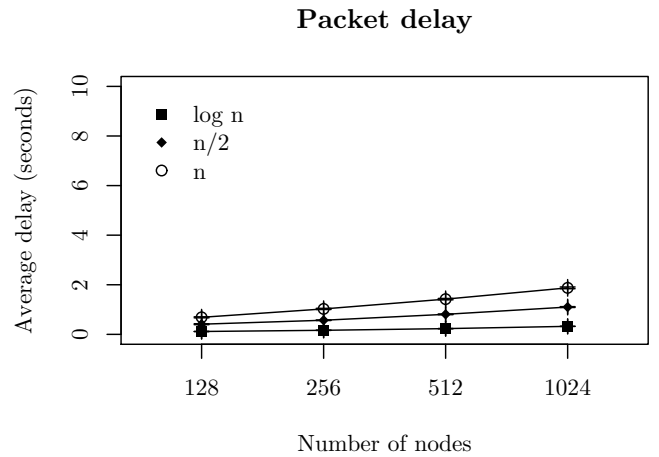


Fig. 11. Average delay with different network sizes.

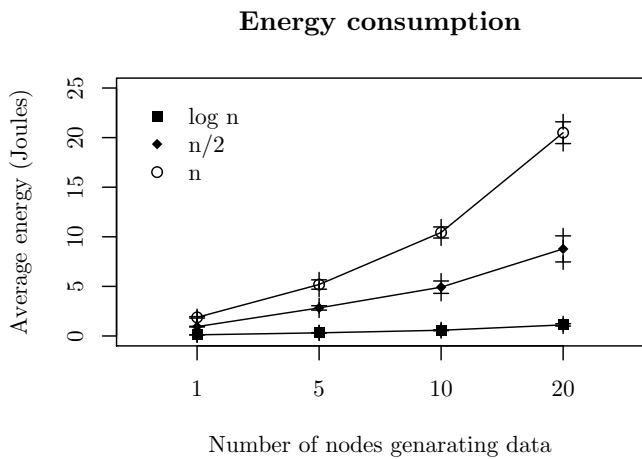


Fig. 10. Total consumed energy with different number of sources.

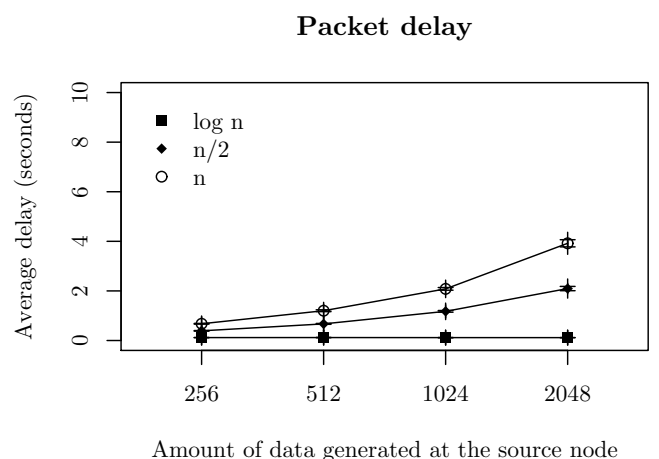


Fig. 12. Average delay with different stream sizes.

In this work, we proposed and evaluated the data stream algorithm that uses a sampling technique over a histogram to reduce data traffic, and consequently the delay and energy consumption. This work represents a way of dealing with energy and time constraints at the application level, as a complementary view of solutions that deals with this problem in the lower network levels.

The results show the efficiency of the proposed method by extending the network lifetime — since data transmission demands lots of energy — and by reducing the delay without losing data representativeness. Such a technique can be very useful to achieve energy-efficient and time-constrained sensor networks if the application is not so dependent on the data precision or the network operates in an exception situation (e.g., few resources remaining or urgent situation detection).

As future work, we intend to apply the proposed method to process data streams along the routing task. Thus, not only the

data from a source is reduced, but similar data from different sources is also reduced, resulting in more energy efficiency. In addition, we plan to use other distributions to analyze the behavior of our algorithm.

REFERENCES

- [1] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Mediterranean Control Conference (Med05)*, 2005.
- [2] A. Lins, E. F. Nakamura, A. A. Loureiro, and C. J. Coelho Jr., "Beanwatcher: A tool to generate multimedia monitoring applications for wireless sensor networks," in *Management of Multimedia Networks and Services*, ser. Lecture Notes in Computer Science, A. Marshall and N. Agoulmine, Eds., vol. 2839. Belfast, Northern Ireland: Springer-Verlag Heidelberg, September 2003, pp. 128–141.
- [3] —, "Generating monitoring applications for wireless networks," in *Proceedings of the 9th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2003)*, Lisbon, Portugal, September 2003.

Packet delay

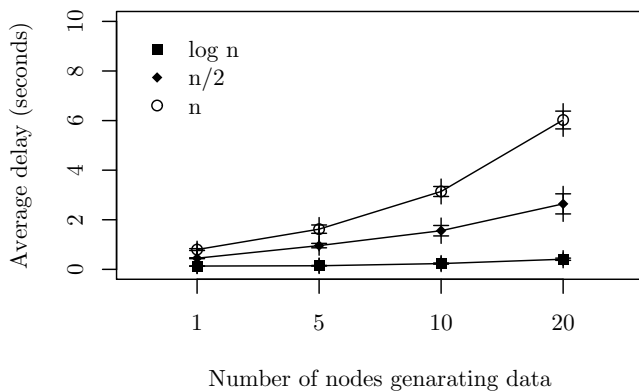


Fig. 13. Average delay with different number of sources.

- [4] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, April 2002.
- [5] D. J. Abadi, W. Lindner, S. Madden, and J. Schuler, "An integration framework for sensor networks and data stream management systems," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases. VLDB 2004*, September 2004, pp. 1361–1364.
- [6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the twenty-first ACM SIGMOD–SIGACT–SIGART symposium on Principles of database systems*, June 2002, pp. 1–16.
- [7] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: An acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 122–173, March 2005.
- [8] Y. Yao and J. Gehrke, "Query processing for sensor networks," in *First Conf. on Innovative Data Systems Research (CIDR)*, January 2003.
- [9] S. Babu, L. Subramanian, and J. Widom, "A data stream management system for network traffic management," in *Proceedings of Workshop on Network-Related Data Management (NRDM'01)*. Santa Barbara, California, USA: ACM SIGMOD, May 25 2001, p. n. 2.
- [10] J. Ledlie, C. Ng, D. A. Holland, K.-K. Muniswamy-Reddy, U. Braun, and M. Seltzer, "Provenance-aware sensor data storage," in *1st IEEE International Workshop on Networking Meets Databases (NetDB)*, April 2005.
- [11] M. Datar, A. Gionis, P. Indyk, and R. Motwani, "Maintaining stream statistics over sliding windows," *SIAM Journal on Computing*, vol. 31, no. 6, pp. 1794–1813, 2002.
- [12] M. R. Henzinger, P. Raghavan, and S. Rajagopalan, "Computing on data stream," Digital Systems Research Center. Tech. Rep., May 1998.
- [13] S. Muthukrishnan, "Data streams: Algorithms and applications," in *4th ACM-SIAM Symposium on Discrete algorithms*, Baltimore, Maryland, 2003.
- [14] Z. Bar-Yosseff, R. Kumar, and D. Sivakumar, "Reductions in streaming algorithms, with an application to counting triangles in graphs," in *13th Annual ACM-SIAM Symposium on Discrete algorithms (SODA'02)*. San Francisco, California, USA: ACM SIAM, January 6–8 2002, pp. 623–632.
- [15] L. S. Buriol, D. Donato, S. Leonardi, and T. Matzner, "Using data stream algorithms for computing properties of large graphs," in *Workshop on Massive Geometric Data Sets (MASSIVE'05)*, Pisa, Italy, June 9 2005.
- [16] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Lecture Notes In Computer Science. Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, vol. 2380. Springer-Verlag, July 2002, pp. 693–703.
- [17] M. Datar and S. Muthukrishnan, "Estimating rarity and similarity

over data stream windows," in *Lecture Notes In Computer Science. Proceedings of the 10th Annual European Symposium on Algorithms*, September 2002, pp. 323–334.

- [18] P. Indyk, "A small approximately min-wise independent family of hash functions," in *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*. Baltimore, Maryland, United States: ACM-SIAM, January 17–19 1999, pp. 454–456.
- [19] J. Zhao, "An implementation of min-wise independent permutation family," Available: <http://www.icsi.berkeley.edu/~zhao/minwise/>, May 2006.
- [20] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Computer and System Sciences*, vol. 60, pp. 630–659, October 2000.
- [21] S. Siegel and J. N. John Castellan, *Nonparametric Statistics for the Behavioral Sciences*, 2nd ed. McGraw-Hill College, January 1988.
- [22] J. Heidemann, F. Silva, and D. Estrin, "Matching data dissemination algorithms to application requirements," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys'03)*. Los Angeles, CA, USA: ACM Press, November 2003, pp. 218–229.
- [23] E. F. Nakamura, F. G. Nakamura, C. M. Figueredo, and A. A. Loureiro, "Using information fusion to assist data dissemination in wireless sensor networks," *Telecommunication Systems*, vol. 30, no. 1-3, pp. 237–254, November 2005.