Data Stream Algorithms For Processing of Wireless Sensor Network Application Data

André L.L. de Aquino[1], Carlos M.S. Figueiredo[1,2], Eduardo F. Nakamura[1,2]

Luciana S. Buriol[3], Antonio A.F. Loureiro[1], Antônio Otávio Fernandes[1], 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 two data stream algorithms for wireless sensor networks (WSNs), based in sample and sketch technics. For each case, we show that by using our algorithms, we can save energy and reduce delay in WSN applications in different scenarios. Specifically, the sampling solution, provides a sample of only $\log n$ items to represent the original data of n elements. Despite of reduction, the sampling solution keep a good data quality.**

I. Introduction

The data that the wireless sensor networks (WSNs) [1]–[3] process usually arrives in an online fashion, is unlimited and there is no control in the arrival order of the elements to be processed. Data with this characteristic is called *data stream*. However, there is a difference between sensor stream and traditional stream. The sensor streams are only samples of the entire population, usually imprecise and noisy, and typically of moderate size. On the other hand, in traditional stream the entire population is usually available, the data is exact, error-free and huge [4].

Recently research in traditional data stream algorithms try to establish their lower bounds. The main metrics analyzed are time and communication complexities [5]–

[7]. 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 [8]–[11]. Indyk [12] proposes a data stream algorithm (implemented by Zhao [13]) that uses a family of hash functions called *min-wise* [14] to compute properties in data streams. This algorithm uses $O(\log n)$ bits to represent a hash index. The Indyk's algorithm computes a $\delta - error$ and an $\epsilon - approximation$ for the index found.

There are many techniques, in traditional streaming, that reduce the volume of data that can be applied or adapted in sensor stream. Examples of some technics are: sampling, histograms, sliding windows, sketches, wavelets, and others. Applications of each one of these techniques generates data similar to the real ones. The similarity of the generated data and the real data depends on how the technic adopted is conduced and the applications requires to be computed.

There are two main types of applications for WSNs: monitoring and actuating applications. In monitoring applications, the nodes only processes the data. In actuating applications, the nodes can interfere in the monitored environment [15], [16]. In the both cases we can to apply the data stream technics, to process the sensor stream in monitoring case or to compose stream queries in actuating case.

The most common sensor stream 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 [17]–[20]. Some proposals use the amount of resources available at a DSMS and apply it to extract management information from the WSN, such as energy and node location [21], [22]. However, current DSMS's are not suitable for WSNs, since nodes have too few resources.

If a node sends all its measurements, it will spend much energy, and part of the data probably will be delayed or lost. For avoiding that, part of the data is not processed. Data stream algorithms based on sampling process only part of the data, producing data similar to the original. The data stream algorithms that sketches data, reduce the data through a data sketch. As an example, calculating the minimum, maximum and average of a data [18] or counting the data frequency. Histogram is another technique, used to capture the distribution or the data behavior, e.g., data is analyzed and accumulated according its kind, in such way that only one data in this distribution is stored [23].

There are some solutions in WSNs that use process like data stream. In same cases the application does adaptive sampling, where the samplings is the data sensing [24]–[27]. In other cases, the solutions are based in data reduction or aggregation, normally based in correlated information about the data sensing [28]–[30].

In this sense, this work applies sensor stream technics to reduce the network traffic keeping the data quality and representativeness. We propose two data stream algorithms for WSNs that use a sampling and sketch of data. With our solutions it is possible 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. In special, the sampling algorithm aims to choose the ideal sample size for processing data streams.

This work is organized as follows. In Section II, we introduce the data stream problem. Next, in Section III, we present the data stream algorithms for sensor network data. Experimental results are given in Section IV, and Section V concludes this study and presents the future work.

II. Problem Definition

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

Problem Statement: Given a sensor stream, we want to meet WSN requirements by reducing the data traffic by using data stream techniques and assuring a minimum data quality order to reduce energy consumption and delay.

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 or sketches of the data stream are sent. In order this, we must evaluate if these data sent are representative. To perform this evaluation we can use statistics tests to know whether the original sensor stream 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? In the sampling case, we need to identify the minimum data sample that can be used in specific application. 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. When we use the sketch, it represents all data, using the fixed size. In this case we loss the data sequence.
- **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 the sketch case we loss the sequence of data however we have a good approximation of the original sensor stream where the data can be regenerated artificially in the sink.

All these questions must be answered to conceive some solution to sensor stream. 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.

III. Sensor Stream Solutions

To address the problem stated above, we need to design algorithms that reduce the traffic in the network.

This reduction must keep the data similarity, and also attend the network requirements. The solutions use the sampling and sketch technics and they are described in the follow subsections.

A. Sampling Based Algorithm

This solution is motivated by the problem address in Section II. Where the *data reduction*, can be provided by sampling of the original data. This solution tries to keep the data quality and the sequence of sensor stream. Our sampling based algorithm provides a solution to allow the balance between best data quality and network requirements. 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 sampling algorithm can be divided into the following steps:

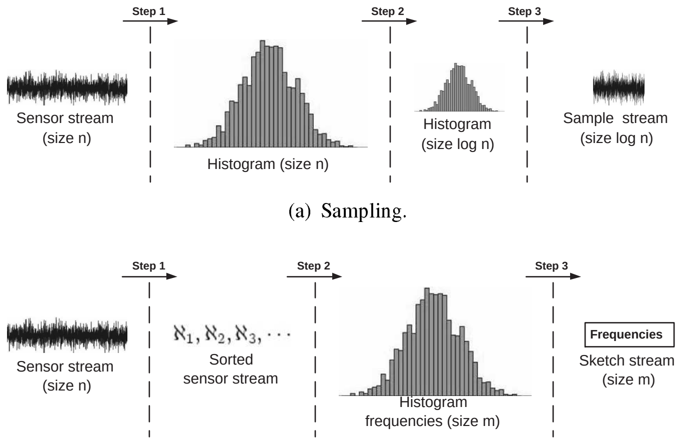
Step 1: Build a histogram of the sensor stream.

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.

These steps is showed in Fig. 1(a). The original sensor stream is composed of n elements. The histogram of the sensor streaming is built in step 1. A minor histogram is built in step 2, it has the sample size required (in the case $\log n$), and keeping the same frequencies of original histogram. Finally, the minor built histogram is reordered to keep data sequence in the step 3.

The pseudo-code of the sampling algorithm is given in Fig. 2. We also consider n as the number of elements in the original data stream, and m as the adopted sample size.



(a) Sampling.

(b) Sketch.

Fig. 1. Example of algorithms execution.

Require:
 Vector *_dataIn*; {original data stream}
 m; {sample size}
Ensure: *_dataOut*; {sample stream}
Sort *_dataIn*;
histScale $\leftarrow$ *"Class width"*;
first $\leftarrow$ *_dataIn*[0];
count $\leftarrow 0$, $j \leftarrow 0$;
for $i \leftarrow 0$ **to** n **do**
 if (*_dataIn*[i] $>$ *first* + *histScale*) **or** ($i = n - 1$) **then**
  colFreq $\leftarrow m\times$ *count/dataInSize*;
  while *colFreq* > 0 **do**
   index $\leftarrow$ *"random element in the histogram class"*;
   _dataOut[j] $\leftarrow$ *_dataIn*[*index*];
   $j \leftarrow j + 1$;
   colFreq $\leftarrow$ *colFreq* $- 1$;
  end while
  count $\leftarrow 0$;
  first $\leftarrow$ *_dataIn*[i];
 end if
 count $\leftarrow$ *count* + 1;
end for
Re-sort *_dataOut*; {according to the original order}

Fig. 2. Pseudo-code of the sampling algorithm.

Analyzing the algorithm in Fig. 2 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 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 $numClass(\frac{n+m}{numClass}) = n + m$.
- 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.

B. Sketch Based Algorithm

Like sampling, this solution is motivated by the problem address in Section II. The *data reduction*, can be provided by sketch of the original data. This solution tries to keep the frequency of the data values without losses by using a little constant packet size. With the information passed the data can be generated artificially in the sink node. However, the sketch solution losses the sequence of sensor streaming. The sketch algorithm can be divided into the following steps:

Step 1: Order the data an identify the minimum and maximum values in the sensor stream.

Step 2: Build the data out, only with the histogram frequencies.

Step 3: Mount the sketch stream, with the data out and the information about the histogram.

The execution of algorithm is showed in Fig. 1(b). The original sensor stream is composed of n elements. The sensor stream is sorted, and the sketch information is acquired in step 1. The histogram frequencies is built in step 2, where m is the number of column in histogram. The sketch stream with the frequencies and sketch information is mounted in step 3.

The pseudo-code of the algorithm is given in Fig. 3. We also consider n as the number of elements in the original data stream, and m as the histogram column number.

Analyzing the algorithm in Fig. 3 we have, line 1 executes in $O(n \log n)$. Lines 6–14 execute in $O(n)$. Thus, the overall time complexity is $O(n \log n) + O(n) = O(n \log n)$. The space complexity is $O(n+m) = O(n)$ if we store the original data stream and the resulting sketch. Since every source node sends its sketch stream towards the sink, the communication complexity is $O(mD)$, where D is the largest route in the network.

IV. Evaluation

When we apply data stream solutions in WSN we have to analyze the network and data quality behavior. That is, what the impact over the network, when we apply our data stream solutions? And, how much the application

```
Require:
   Vector _dataIn;                          {original data stream}
   m;                                       {sketch size}
Ensure: _dataOut;                           {sketch stream}
Sort _dataIn;
histScale ← "Class width";
first ← _dataIn[0];
m ← (_dataIn[n] - _dataIn[0]) / histScale;
count ← 0, j ← 0, index ← 0;
for i ← 0 to n do
   if (_dataIn[i] > first + histScale) or (i = n − 1) then
      _dataOut[index] ← count;
      index ← index + 1;
      count ← 0;
      first ← _dataIn[i];
   end if
   count ← count + 1;
end for
```

Fig. 3. Pseudo-code of the sketch algorithm.

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	$\log n$, $n/2$
Sketch size	10

data losses when we use our solutions? These questions are answered in the next subsections.

A. Methodology

The evaluation of the algorithms 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 topologies. 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 [31], [32], the density is kept constant, 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. In the sampling case, for each evaluated parameter we analyzed the application and network behavior by using sample size of $n/2$ and $\log n$. All parameters used in the simulations are presented in Table I.

We evaluated the algorithms by considering two parts: evaluation of network behavior by use the sampling and sketch solutions and evaluation of data quality only with sampling solution. In order to evaluate the distribution approximation between the original and sampled streams, we use the Kolmogorov-Smirnov test (K-S test) [33]. 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, $Data\ Error = Max\{|lower_{value} - Generate_{avg}|, |higher_{value} - Generate_{avg}|\}$, where the pair $(lower_{value}; higher_{value})$ is the confidence interval of data sample and $Generate_{avg}$ is the average of original data.

B. 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. In this evaluation, for sampling algorithm we use different sample sizes ($\log n$ and $n/2$) and the complete sensor stream (n) and for sketch algorithm we use a fixed size (10 ranges). Both cases are analyzed with different network scenarios by varying the network size, the amount of generated data at the source, and the number of sources.

Figs. 4, 5, and 6 show the energy consumption performance. We observe in all cases with the sampling solution when sample size is diminished the consumed energy is diminished too. The sketch solution follows the sample-$\log n$ result. This occurs because the packet size is constant and near of sample-$\log n$ packet size.

Analyzing separately, when the number of nodes is varied (Fig. 4) the consumed energy does not vary. This occurs because only one source is used, the sensor streaming size is the same, and the network density is kept. Even trough, in this scenario the sample-$\log n$ and the sketch solution have less impact over the consumed energy.

When the sensor streaming size is varied (Fig. 5), we can observe the impact of our solutions in the energy consumption. The sample-$\log n$ and the sketch have the best performance in all cases, and the energy consumed do not vary when sample size increase. In the sample-$\log n$ case, this occur because the packet size is increase only one element when we increase the sensor stream size (256, 512, 1024, 2048), and in the sketch case the packet size used is always constant. The others results (sample-$n/2$ and n) have worse performance because the packet size is increased proportionally when the sensor streaming size is increased.

When the number of nodes generating data are varied (Fig. 6), one more time, the sample-$\log n$ and the sketch have the best performance in all cases. This occur because, in this scenario more packets are passing through the network when we increase the number of nodes generating data. Each source using the sample-$\log n$ or sketch solution use only one packet (the packet size is not more 20B) to send its data at the sink. The others results (sample-$n/2$ and n) each source node generate more than one packet for application, this overload the network, causing more energy consumption.

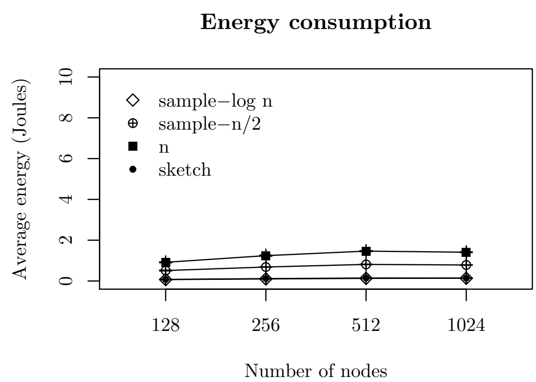


Fig. 4. Total consumed energy with different network sizes.

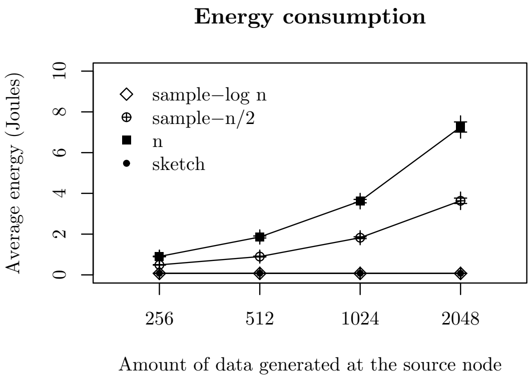


Fig. 5. Total consumed energy with different stream sizes.

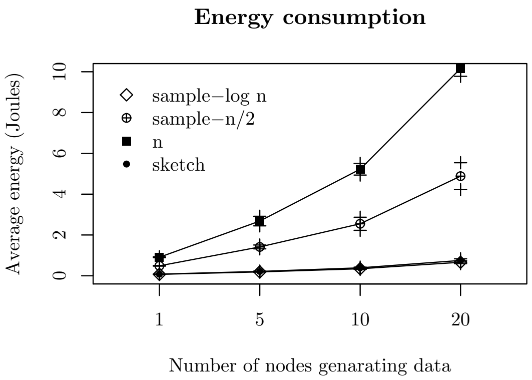


Fig. 6. Total consumed energy with different number of sources.

The delay performance is showed in Figs. 7, 8, and 9. Like the energy results, we can see that when sample size is diminished, the delay is diminished too for the same reason. Again, the same effect of the number of nodes variation is observed (Fig. 7). When the sensor stream size and number of nodes generating data are varied we can observe the delay impact by using our solution. Again, in the all cases, the sample-$\log n$ and sketch have the best performance.

C. Data Quality

Here, we present the impact of our solution by evaluating data quality. This evaluation is only for sampling

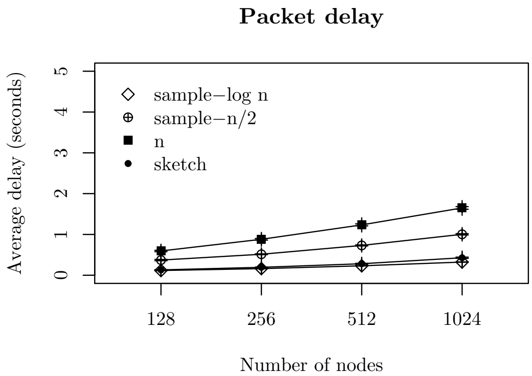


Fig. 7. Average delay with different network sizes.

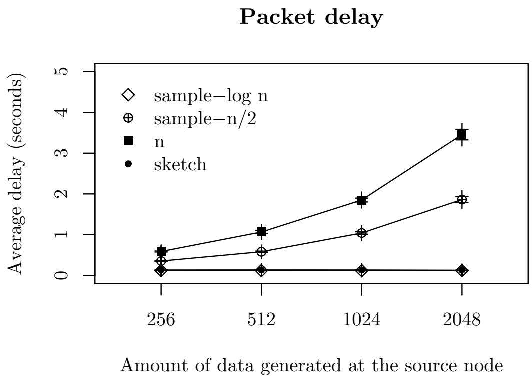


Fig. 8. Average delay with different stream sizes.

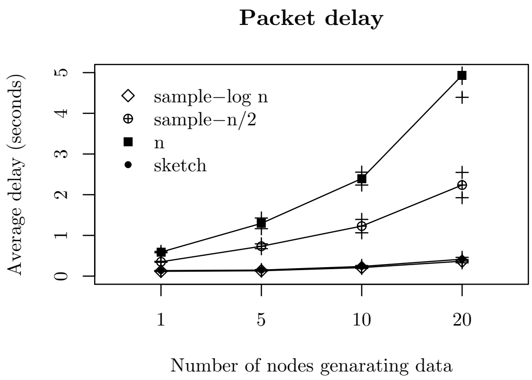


Fig. 9. Average delay with different number of sources.

solution, because this solution losses information in its process, so is important to evaluate the impact of this losses in the data quality. In the sketch case, all data can be generated artificially when arrive in the sink node, so the losses are not identified when the data tests are applied. The only impact generated by sketch solution is the lost of the data sequence which was not evaluated here.

In order to this, the impact of sampling solution is made through the K-S test and the average error. Like the network evaluation, we use different sample sizes ($\log n$ and $n/2$) and the complete sensor stream (n) in different network scenarios. We vary the network size, the amount of data generated at the source, and the number of sources.

Figs. 10, 11, and 12 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 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. In all cases, the error is constant, this occurs because the data lost in the network is very little. The greater error occur when we use a minor sample size but the data similarity is kept.

We also evaluate the data quality through the discrepancy between the original and sampled stream average values (Figs. 13, 14, and 15). 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-$n/2$. Again, in all cases the error is constant for the same reason of the *ks-diff*. However an important observation is that the *data-error* is the same for use sample-$n/2$ and n. So if we want to keep the maximum data quality, considering the *data-error* we must send only sample-

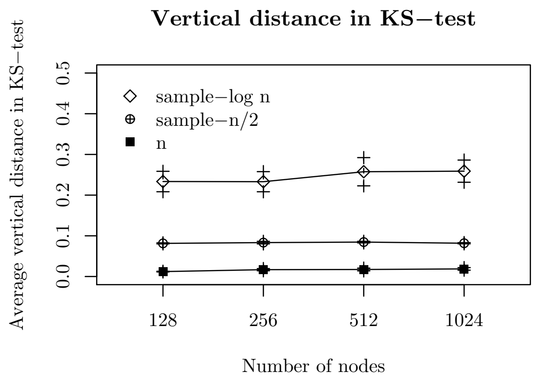


Fig. 10. K-S distance in different network sizes.

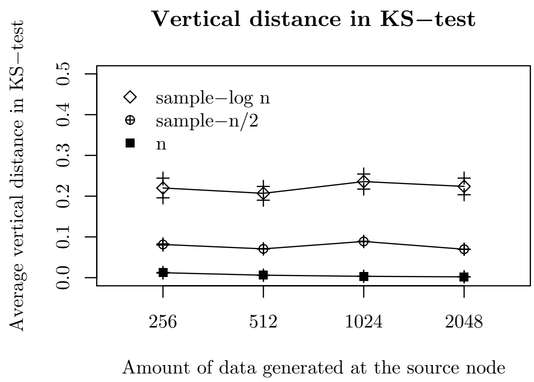


Fig. 11. K-S distance with different stream sizes.

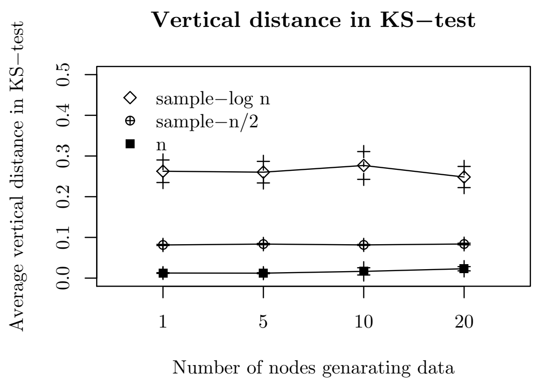


Fig. 12. K-S distance with different number of sources.

$n/2$.

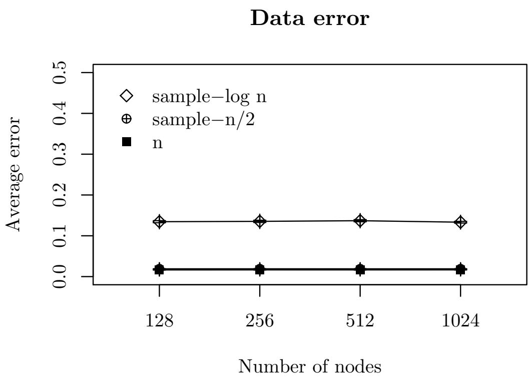


Fig. 13. Average error with different network sizes.

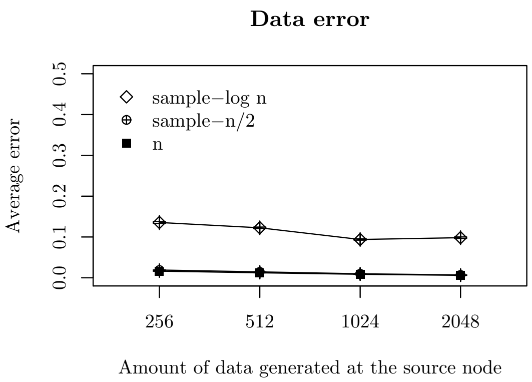


Fig. 14. Average error with different stream sizes.

D. Results Summary

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

- The sketch reduces the consumed energy and delay by keep a constant transmitted data. Once, the data can be generate artificially in the sink, the data quality is not affected in the distribution similarity and average discrepancy. The problem is the sequence

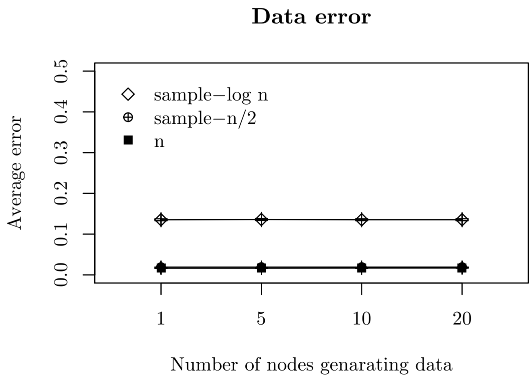


Fig. 15. Average error with different number of sources.

of data that is lost. But the sequence lost may be acceptable by a large majority of applications when the network restrictions are strong. The cases where the data sequence is important we must use the sampling solution.
- The 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.
- The sample-$n/2$ is interesting either when the application priority is the average discrepancy (near zero), or we have the scenario presented in Fig. 4, 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, when do we use sampling or sketch? In the case where the data sequence is important we can use the sampling, in this case we can always

analyze the application or network requirements to decide about the best sample size. If the sequence is not important we can use the sketch because it always has the best network performance keeping the integrity of all data. The advantages of the sketch over sampling is that the sketch solution can be modified for on-line processing of the sensor stream, without the storage of the original data.

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

V. Conclusion and Future Work

WSNs 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.

In this work, we proposed and evaluated two data stream algorithms that use sampling and sketch techniques to reduce data traffic, and consequently reduce 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 methods 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 sensor streams along the routing task and in clustered networks. Thus, not only the data from a source is reduced, but similar data from different sources can be also reduced, resulting in more energy efficiency. We also intend to evaluate other data stream solution like wavelets where specifically data characteristics can be analyzed. However, we plan to use other data distributions to analyze the behavior of our algorithms and use other scenarios when data lost can be affected in data quality.

References

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: Scalable coordination in sensor networks," in *Fifth Annual International Conference on Mobile Computing and Networks (MobiCom'99)*. Seattle, Washington, USA: ACM Press, August 1999, pp. 263–270.

[2] 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.

[3] T. Arampatzis, J. Lygeros, and S. Manesis, "A survey of applications of wireless sensors and wireless sensor networks," in *Mediterranean Control Conference (Med05)*, 2005.

[4] E. Elnahrawy, "Research directions in sensor data streams: Solutions and challenges," Rutgers University, Tech. Rep. DCIS-TR-527, May 2003.

[5] 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.

[6] M. R. Henzinger, P. Raqhavan, and S. Rajagopalan, "Computing on data stream," Digital Systems Research Center, Tech. Rep., May 1998.

[7] S. Muthukrishnan, "Data streams: Algorithms and applications," in *4th ACM-SIAM Symposium on Discrete algorithms*, Baltimore, Maryland, 2003.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] J. Zhao, "An implementation of min-wise independent permutation family," Available: http://www.icsi.berkeley.edu/ zhao/minwise/, May 2006.

[14] 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.

[15] 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.

[16] ——, "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.

[17] 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.

[18] 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.

[19] 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.

[20] Y. Yao and J. Gehrke, "Query processing for sensor networks," in *First Conf. on Innovative Data Systems Research (CIDR)*, January 2003.

[21] 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.

[22] 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.

[23] Y. E. Ioannidis and V. Poosala, "Histogram-based approximation of set-valued query answers," in *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, 1999.

[24] D. Ganesan, S. Ratnasamy, H. Wang, and D. Estrin, "Coping with irregular spatio-temporal sampling in sensor networks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. Issue 1, pp. 125–130, January 2004.

[25] R. Willett, A. Martin, and R. Nowak, "Backcasting: adaptive sampling for sensor networks," in *Proceedings of the third international symposium on Information processing in sensor networks*, ACM. Berkeley, California, USA: ACM Press. New York, NY, USA, April 2004, pp. 124–133.

[26] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, vol. 72, ACM. Toronto, Canada: ACM Press. New York, NY, USA, August 2004, pp. 10–16.

[27] A. D. Marbini and L. E. Sacks, "Adaptive sampling mechanisms in sensor networks," in *London Communications Symposium - LCS2003*, London, UK, September 2003.

[28] K. R. Silvia Santini, "An adaptive strategy for quality-based data reduction in wireless sensor networks," in *3rd International Conference on Networked Sensing Systems (INSS 2006)*, Chicago, USA, 31 May – 2 June 2006.

[29] P. von Rickenbach and R. Wattenhofer, "Gathering correlated data in sensor networks," in *Proceedings of the 2004 joint workshop on Foundations of mobile computing*. Philadelphia, PA, USA: ACM Press New York, NY, USA, October 2004, pp. 60–66.

[30] J. Zhu and S. Papavassiliou, "A resource adaptive information gathering approach in sensor networks," in *Sarnoff Symposium on Advances in Wired and Wireless Communication, 2004 IEEE*. Nassau Inn in Princeton, NJ, USA: IEEE, April 2004, pp. 115–

118.

[31] 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.

[32] 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.

[33] S. Siegel and J. N. John Castellan, *Nonparametric Statistics for the Behavioral Sciences*, 2nd ed. McGraw-Hill College, January 1988.