DYSTO - A Dynamic Storage Model for Wireless Sensor Networks

Nuno M. F. Gonçalves, Aldri L. dos Santos, and Carmem S. Hara

Universidade Federal do Paraná, Brazil
{nunom, aldri, carmem}@inf.ufpr.br

Abstract. One of the main problems in wireless sensor networks (WSN) is the energy consumption needed for storing and querying sensed data, since sensors are devices with very limited resources and battery. One of the main factors that has an impact on energy consumption is the data repository location, that is, the location where the sensor data is stored. In this paper we propose DYSTO, an in-network and dynamic data storage, in which the location of the repository is chosen based on information collected over the network. Intuitively, DYSTO chooses to store sensed data close to where it is most frequently needed: close to the query entry point when the query rate is high, and close to data sources for coping with high sensed data production rates. Moreover, the query load is analyzed in order to keep values that are frequently queried together in the same repository. Our ultimate goal is to reduce the power consumption of the network based on an adaptive approach for repository selection, and a user-defined update strategy based on data thresholds. DYSTO has been implemented on NS2 network simulator and our experimental results show that it has less power consumption than similar approaches.

Categories and Subject Descriptors: H.2.4 [**Database Management**]: Distributed databases; H.3 [**Information Storage and Retrieval**]: Miscellaneous

Keywords: Data Caching, Data Management, Wireless Sensor Networks

1. INTRODUCTION

The recent development in micro-processors and communication technologies led to the emergence of new types of fully distributed networks consisting of tiny sensing units, called wireless sensor networks (WSNs). WSNs are ad-hoc networks that communicate via radio signals and may consist of hundreds or even thousands of sensors [Akyildiz et al. 2002]. Sensor platforms are unreliable because they have a limited lifespan due to energy constraints. They may also present communication problems because of interference in radio signals due to obstacles in the sensing environment. Given these unique characteristics of WSNs, one of the main research topics in this area is related to the optimization of available resources.

In WSNs the energy consumption resulting from transmissions represents one of the main factors for shorter longevity and consequent loss of data [Pottie and Kaiser 2000]. For large WSNs, raw data transmissions between sensors and the base station, either for storing or querying data, without any kind of treatment or aggregation to minimize the volume of traffic is therefore unfeasible. For this reason, data compression and aggregation techniques [Brayner et al. 2008] as well as caching techniques [Nascimento et al. 2010] have been proposed to reduce the data size and minimize the overall communication overhead. Another important factor in reducing the network overhead is the overall adaptability of the system. Due to the WSNs limitations, energy depletion and consequent topology changes lead to a constantly changing environment. Since the environment keeps changing

This work is partially supported by the Brazilian agency CAPES and CTIC/RNP $(CIA)^2$ project - *Construindo Cidades Inteligentes: da Instrumentação dos Ambientes ao Desenvolvimento de Aplicações.*

so should the storage model for supporting it.

There are several previous work related to data placement and aggregation for WSNs. Storage models can be divided into three major groups [Fangfang et al. 2007] [Shen et al. 2010]. The first group comprehends the *local* storage. Systems that follow this approach keep the data where they are produced. That is, sensors keep their own data. Although there exists no overhead for data updates on external devices, query processing can be extremely expensive because it requires broadcasting a query request on the entire WSN and sensors to send individual replies back to the query entry point. Thus, this storage model is better suited for scenarios with a high volume of data production and low query rates. The second group comprehends the *external* storage. In this storage model data are sent from the producer sensors to a base station, either in raw or aggregated format. This model generates a high volume of messages from sensors to the base station, but on the other hand it presents a very low query cost. As opposed to local storage, the external model has better performance in scenarios where there exists a large volume of queries and low data generation frequency. The third group comprehends the *data-centric* storage. This model distributes sensed data in the network based on a predefined set of rules or functions on their values. The storage cost is higher than for local storage but its query cost is lower; moreover, compared to the external storage, the data-centric approach has lower storage cost and higher query cost. In this article we propose DYSTO, a dynamic storage model for WSNs that takes advantage from each of these models, by integrating them into a single and adaptable model which is context-sensitive.

DYSTO collects information of the sensor network so that it can dynamically adapt to a suitable configuration that minimizes the overall storage and query costs. In this article we consider a WSN that collects a numeric information, such as temperature and humidity, and the user issues queries *based on sensed data values*. That is, given a certain range of values iv and a given time t, the user wants to know which sensors have readings within the interval iv at time t. In order to determine when a new storage configuration is needed, sensors periodically send summary messages to the base station with several local information such as the production rate. Information of all sensors on the field are gathered at the base station so that a suitable storage model for the current context is adopted. Intuitively, the base station selects a data placement for sensed values intervals. The placement can be at the producer sensor (local storage), the base station (external) or at a sensor in the WSN (data-centric). That is, the base station generates a mapping from values to repositories, based on the following heuristics: (a) as the query frequency on a given value increases, closer to the base station it should be stored; (b) as the production frequency of a given value increases at the sensor devices, closer to the producers it should be stored; and (c) if two values are often queried together, they must be stored in the same repository. These heuristics can be considered as adaptations of strategies traditionally used in databases to determine index clustering and fragmentation schemas in the context of WSNs.

DYSTO was developed in the application layer and it is a topology independent model; that is, it is not tied to a particular routing protocol or network topology. However, it relies on a transport protocol that implements horizontal routing, in which each sensor keeps routes not only for its direct neighbors, ascendants and descendants, but also paths for sensors in the same routing tree level. This is because sensors need to know all or at least part of the path for routing sensed data updates to the repositories directly. The model proposed in this article is an extension of the model proposed by SCOOP [Gil and Madden 2007], a system which considers the frequency of queries, data generation, and network conditions to establish storage configurations and update them. However, the system does not take into account the co-occurrences of data in queries and adopts a simple and costly model for gathering system monitoring information. Thus, the contributions of this work are:

—extension of the dynamic model proposed in [Gil and Madden 2007], considering the co-occurrences of data on queries;

—adoption of user defined thresholds for determining the transmission frequency of system monitoring

information;

—an experimental study, based on simulations, showing that the proposed strategies can significantly reduce the number of message transmissions in the system.

Although relying on the user to set system thresholds may add on the system complexity and impact its usability, our experimental studies show that the ability to change them have a big impact in terms of energy savings, compared to fixed values. In this article we present a study of the effects of these parameters on the system, and it is a first step towards a self-tuning adaptable system that can be adjusted to meet the application needs.

The rest of the article is organized as follows. Section 2 presents work related to the three types of data storage in sensor networks. The proposed dynamic model of storage is detailed in Section 3. The results of experimental studies are presented in Section 4 and the article concludes in Section 5, listing future work and some final considerations.

2. RELATED WORK

This section presents some related work that adopt storage models in the three categories considered in this article: external, local and data-centric. Among the works that adopt external storage we can mention the ones proposed in [Yao et al. 2006], [Szewczyk et al. 2004] and [Gupta and Dave 2008]. Although for this strategy the query cost is very small, sending all the data produced by sensors to a single point outside the WSN or to the base station can cause huge energy consumption due to unnecessary data transmission [Szewczyk et al. 2004]. Thus, external storage is only suitable for small WSNs with low frequency data generation and high query rate.

Local storage models, such as [Yoon and Shahabi 2007] and [Intanagonwiwat et al. 2000], store data at producer sensors themselves and require network flooding for processing queries based on values. The direct broadcast protocol [Intanagonwiwat et al. 2000] has been proposed for reducing the cost of processing data queries. It keeps fixed routes between sensors and it is optimized for processing follow up queries on the same data that have been previously accessed. Each sensor node only needs to store its own time-series data stream (or some important attributes) on its local-flash memory, and then users can issue queries by flooding or adopting a geographical routing protocol to forward query packets to appropriate sensor nodes. For local storage models, a heavy workload of user queries can burden the sensor devices and the sensor network as well.

There are several systems based on data-centric storage models, such as GHT [Ratnasamy et al. 2003] [Li et al. 2003] [Greenstein et al. 2003], DIMENSIONS [Ganesan et al. 2003], and the hierarchical models of [Yang et al. 2010] and [Yu et al. 2010]. The GHT [Ratnasamy et al. 2003] proposes a storage scheme that uses a hash function for mapping sensed data to a specific location. In this approach, all data containing the same name are stored in a single sensor and a geographic routing protocol is used to locate the data. [Li et al. 2003] and [Greenstein et al. 2003] extend the GHT providing distributed hierarchies of data rates. These two techniques, unlike the GHT, consider indexes on several attributes. DIMENSIONS [Ganesan et al. 2003] proposes a method for long term data storage that progressively discards old data while still preserving historical data characteristics in order to support data mining. Among hierarchical storage systems, the strategy proposed by [Yang et al. 2010] cache data, such that each level in the hierarchy stores data aggregations with different levels of precision. In [Yu et al. 2010] the data production rate assumes a decisive role in minimizing energy storage costs. However, the system is not adaptable and relies on fixed routes during specific periods of time. [Matos et al. 2010] proposes a storage method based in data prediction using a dynamic regression model.

To the best of our knowledge, the work most related to the model presented in this article is the SCOOP system [Gil and Madden 2007]. Similar to DYSTO, it is based on a model that dynamically

adapts the data placement to the conditions of the WSN. However, DYSTO extends SCOOP by considering new heuristics and user-defined parameters to further reduce the overall energy consumption of the system.

3. THE DYSTO MODEL

This section presents DYSTO, a dynamic storage model for wireless sensor networks (WSN). A WSN is composed of a set of sensors spread over a monitored area that communicate via radio. If two sensors are within the radio communication range of each other, we say that their distance is *one-hop* and that they are *neighbors*. Thus, communication between two arbitrary sensors may require a message to be transmitted along a number of intermediate devices for reaching its final destination. That is, WSNs are based on *multi-hop* communication, and rely on a *routing* protocol for determining paths that establish communication between sensors. In this article we assume sensors to be static, and thus have a fixed geographic coordinate. To simplify our discussion, we also assume that each sensor is responsible for monitoring a single measurement from the environment.

DYSTO aims at reducing sensor devices energy consumption. Given that for sensors the energy required for communication is much higher than in-place processing [Tilak et al. 2002], our goal is to reduce the overall number of transmissions (or hops) for storing and querying sensor readings. DYSTO follows a threefold approach. First, sensor readings are stored close to where they are more frequently needed: the source device for update operations, and the query entry point for read operations. Second, values that are frequently accessed together, such as in range queries, are stored in the same repository. Third, the update frequency is based on user-defined thresholds. The first strategy has been originally introduced by the SCOOP system [Gil and Madden 2007], while the other two are inspired by traditional approaches for distributed storage. Similar to SCOOP, the main objective of the DYSTO model is to generate a storage assignment (SA) which maps range of values to repositories. A repository can be the producer sensor itself (local storage), the base station (external storage), or a sensor that concentrates data from all sensors that have readings within a range (data-centric storage).

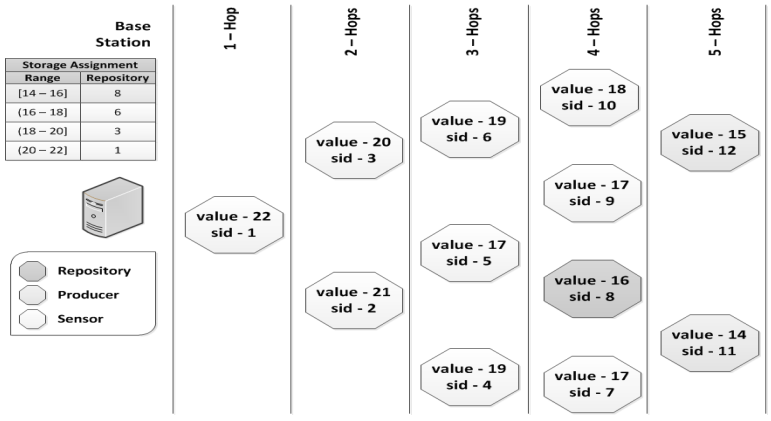


Fig. 1. Sensor field and a storage assignment (SA)

To illustrate, consider the sensor field in Figure 1. The SA maps values $[14, 16]$ to sensor 8, $(16, 18]$ to sensor 6, $(18, 20]$ to sensor 3, and $(20, 22]$ to sensor 1. Considering that queries are always issued from the base station (BS), this SA is convenient in a context in which the number of sensor readings updates is larger than the number of queries, since repositories are placed close to the devices that produce them. If the query rate for values in the range $[14, 16]$ increases then a new SA can be

generated placing the repository closer to the base station, for example, sensor 5. The goal of SA adjustments is to minimize the overall number of transmissions among sensor devices for processing queries and update repository values.

The number of entries in the SA is a system input parameter. In the example of Figure 1 it is set to 4, where the lowest sensor reading is 14 and there exist no values higher than 22. These limits are dynamically defined based on summary messages that sensor devices send to the BS for monitoring the network. Given that in our model queries are issued from the BS, it can also gather information on query rates, query probabilities and the network topology. All these information are considered for determining the repository placement in the WSN. The SA is recalculated periodically following an interval predefined by the system. Thus, an SA is always associated with a timestamp t.

A sensor device generates two types of messages: *data messages* for updating a repository with its current reading, and *summary messages* that are sent to the BS for monitoring purposes. The frequency in which sensors send these messages are defined by user-defined thresholds as follows. For data messages, a data threshold Th_d determines the minimum percentage change in its reading that requires updates in a repository. That is, if the previous reading from a sensor is v_{prev} and the current is v then the sensor should update the repository responsible for storing v if it is not in the interval $[(1-Th_d)v_{prev}, (1+Th_d)v_{prev}]$. Observe that the value of Th_d has a direct impact on the accuracy of query results. That is, high values for Th_d reduces to cost of communication but may generate query results with an error of at most Th_d percent. This is because the value stored in a repository may differ from the actual reading by this percentage.

In order to generate a summary message, each sensor computes a histogram composed of hS user-defined intervals over the last hR sensor readings $M = \{v_1, \ldots, v_{hR}\}$. That is, each sensor keeps at most hR readings following a round-robin strategy, i.e. when the last slot in M is filled, the new reading overwrites the oldest one. The minimum and maximum values in M are used to generate an equiwidth histogram with hS subintervals. The contents of a summary message consists of the histogram, along with the minimum, maximum, and summation of values in M. As an example, suppose the minimum and maximum values are 0 and 10 respectively. If $hS = 5$ the histogram entries are defined for subintervals $[0, 2], (2, 4], \ldots, (8, 10]$, each of them holding the number of values in M within its range. In order to determine when a new histogram should be sent to the BS, we compare the average value of elements in M from the previous histogram with the current one. That is, let $avgM_{prev} = \sum(M_{prev})/hR$. Then a new summary message is generated if the current average value for M has changed by at least a percentage Th_h. That is, the value is not in $[(1-Th_h)avgM_{prev}, (1+Th_h)avgM_{prev}]$.

The components of our WSN model are presented in the following definition. It considers the existence of a time sequence $T = [t_1, t_2, \ldots]$, where $t_i < t_{i+1}$.

Definition 3.1. A WSN is a 10-tuple $W = (BS, G, N, SA, f_{SA}, hR, hS, Th_d, Th_h, P)$, where:

—BS is the base station;

—$G = (S, A)$ is a graph, where S is a set of sensors $\{s_1, \ldots, s_n\}$, such that $a = (s_i, s_j) \in A$ if the sensor s_i is within the communication range of s_j. We say that s_i is one *hop* away from s_j and therefore s_i is a *neighbor* of s_j. Each sensor s collects a new data reading at time t, and this reading is given by $val(s, t)$.

—N is the number of entries in the storage assignment. That is, at any given time t let $minV$ and $maxV$ be the lowest and highest values in the set $\{val(s, t) \mid s \in S\}$. The value of N determines the partition of $[minV, maxV]$ into equally sized subintervals $I = \{iv_1 = [minV, v_1], iv_2 = (v_1, v_2], \ldots iv_N = (v_{N-1}, maxV]\}$;

—$SA[t][iv]$ is the storage assignment, which is a bidimensional matrix where each cell contains the placement at a time t for a reading within a range $iv \in I$. This location may be a sensor in S, the

base station BS or the producer sensor;

—f_{SA} is the frequency in which a new SA is computed at the BS;

—hR is the size of recent-readings buffer kept at each device;

—hS is the number of entries in the histogram, commonly called bins, computed at each device;

—Th_d is the threshold that determines when repository values are updated;

—Th_h is the threshold that determines when summary messages containing a histogram should be sent from a sensor to the BS;

—$P[s]$ is the piggybacking time each sensor s has to wait for summary messages before forwarding it to the BS. Since every sensor send summary messages to the BS, without considering any form of data aggregation, messages of this type would represent a huge weight to the system in terms of energy consumption. Thus, each sensor waits for a certain period before transmitting them. If it receives new messages during this period then they are concatenated into a single message and sent to the BS after the time expires. The function that determines $P[s]$ is given by $P[s] = maxP \times e^{-\lambda \times (dist[BS][s]-1)}$ where $maxP$ is the maximum waiting time, λ is the decay constant and $dist[BS][s]$ is the number of hops between the BS and the sensor s. Both $maxP$ and λ are parameterizable.

The system works as follows. After the calculation of an SA by the BS, a *mapping message* containing the SA is sent to all sensors in the network, along with its timestamp t. Whenever a sensor produces a value v that requires a repository update according to the threshold Th_d, it checks its last $SA[t]$ to find the repository r responsible for storing v and then sends a *data message* to r with the following information: the destination repository r, the value v, the identifier of the producer sensor s and the SA timestamp t. The timestamp t plays and important role for SA updates in the following way. Sensors on the route from the producer sensor s to repository r check whether they have received an SA with a higher timestamp t'. If this is the case, the sensor updates the message and re-routes it to the repository for v according to $SA[t']$. With this strategy, failures to deliver mapping messages with new SAs to a given sensor does not necessarily affect the correct transmission of a data message produced by this sensor to the current repository for its reading.

Given that information on sensors that generate values within each SA interval are stored in the same repository, range and exact value queries require only a single *query message* to be sent from the BS to each repository of interest according to the SA. It is worth noticing that value-based queries are usually very costly to process on non-datacentric WSN models. Other types of queries such as those involving aggregation and join can also be processed, although this is not the focus of this article. Aggregation queries can use the information in the summary messages sent by each sensor to the BS and generate results involving functions maximum, minimum, average, count and sum. This is similar to the strategy proposed by [Ammar and Nascimento 2011], which uses histogram computed at sensor devices to produce query replies directly by the BS. As stressed in [Coman and Nascimento 2007], join queries in WSNs are trickier to compute and our model will perform well only if the attribute being joined is the one being indexed through the SA. Details on the generation of an SA are presented in the next section.

3.1 Storage Assignment Computation

The generation of an SA executed at the BS is based on a number of information collected from summary messages sent by each sensor, query rates and co-occurrences among queried values. In order to promptly react to changes on the system behavior, the BS computes a new SA based on the information collected after the transmission of the last SA.

A summary message sent from each sensor s to the BS includes: a histogram of recent readings of s, the lowest and highest values, the sum of the last hR values produced by s, the timestamp t of the last SA received by s, and information on the network neighborhood topology. Recall that the SA

partitions the range of sensor readings into N subintervals $I = \{iv_1, \ldots iv_N\}$. Since all queries are disseminated in the network from the BS, the following information is also collected at the BS: the total number of queries involving each interval since the last SA computation ($totQry[iv_i]$).

In order to determine the co-occurrences of data on queries, the BS also keeps for each pair (iv_i, iv_j) of intervals in the SA, the number of queries involving values in both intervals ($corr[iv_i][iv_j]$). That is, for each query issued to the WSN, the BS determines the set of intervals $R \subseteq I$ that intersect the query result. Besides sending messages to each of the corresponding repositories for intervals in R, the BS executes the following bookkeeping actions: it increments both the $totQry[iv]$ for each $iv \in R$, and $corr[iv_i][iv_j]$ for each pair of intervals in R.

Based on the collected data, the BS can compute the following information:

—*dist[s_i][s_j]*: distance between two sensors s_i and s_j (or between BS and a sensor). This value is calculated based on the number of hops between the devices given by the topology information in summary messages;
—*prodRate[s]*: frequency in which a sensor s produces a new reading between two SA computations. This value is derived from a history of histograms sent by s;
—*probProd[s][iv]*: probability for a sensor s to produce a value in the interval iv. This is derived from the last histogram posted by s.

Figure 2 presents the SA calculation algorithm. It consists of two steps. The first determines for each sensor on the field the cost of electing it as a repository for a certain range of values. The second considers data co-occurrences on queries for electing a repository. That is, based on the costs determined in the first step and the frequency in which more than one repository need to be accessed for answering queries, a repository that minimizes the overall number of transmissions is selected for each entry in the SA.

The first step (Lines 1 to 10) is similar to the strategy proposed by SCOOP [Gil and Madden 2007]. The goal of this step is to determine the best choice of repositories for each interval in the SA without considering any query co-occurrences among intervals. This choice is based on two cost values: a) the cost for sending value updates from sensors to the repository; and b) the cost for accessing values stored in a repository for answering queries. In order to make this choice, the algorithm computes two matrices, *dataCost* and *qryCost*, relating every possible repository, which can be a sensor device or the BS, with each interval in the SA. The value of an entry *dataCost[r][iv]* contains the cost of sending update messages from every sensor s that may produce a value in the interval iv to r (Line 6). The value of an entry *qryCost[r][iv]* consists of the cost for sending a query message from the BS to r plus the cost for sending the result back to the BS (Line 7). If no query value co-occurrences is considered, the repository for each interval can be chosen as the one with the lowest value for the sum of *dataCost[r][iv]* and *qryCost[r][iv]* (Lines 8 to 10).

The second step of the algorithm takes data co-occurrences into consideration for the SA calculation. Intuitively, if two intervals in the SA are usually accessed together for answering a query, the cost of storing them in the same repository reduces the number of query messages to be transmitted to and from the BS to a repository. However it may incur in larger number of data messages for updating the repository. Thus, in order to decide whether it is worth combining repositories, the overall cost of adopting a single repository for a set of intervals has to be compared to the cost of storing them independently. This step is based on the co-occurrence value between two intervals ($corr[iv_1][iv_2]$).

The algorithm starts by considering the pairs of intervals with highest co-occurrences (Line 12), since these are the ones most likely to result in lower combination costs. For a pair of intervals iv_1, iv_2, the total number of query messages issued on both if they are allocated at the same repository is $totComb = totQry[iv_1] + totQry[iv_2] - corr[iv_1][iv_2]$, since the correlated queries do not generate separate query messages (Line 16). Thus, similar to the computation of $qryCost[r][iv]$, the cost of

Algorithm *SACalculation*
Input: S: set of sensors; t: current time; I: set of intervals in the SA; dist[][]; prodRate[]; probProd[][]; totQry[], corr[][]
Output: $SA[t]$: storage assignment for time t

```
1.  dataCost[r][iv] ← 0 for all repositories r and intervals iv;
2.  qryCost[r][iv] ← 0 for all repositories r and intervals iv;
3.  for all intervals iv in I
4.          for all candidate repositories r in S ∪ {BS}
5.                  for all sensors s in S
6.                          dataCost[r][iv]+ = probProd[s][iv] * prodRate[s] * dist[r][s];
7.                  qryCost[r][iv] = totQry[iv] * 2 * dist[BS][r];
8.  for all intervals iv ∈ I
9.          minCost[iv] ← min({dataCost[r][iv] + qryCost[r][iv] | r ∈ (S ∪ {BS})});
10.         minRep[iv] ← repository r such that (dataCost[r][iv] + qryCost[r][iv]) == minCost[iv];
11.         SA[t][iv] ← null;
12. Q = [[iv1, iv2], ...] ← list of pairs of intervals in descending order of values for corr[iv1][iv2];
13. for all [iv1, iv2] in Q
14.         if SA[t][iv1] ≠ null e SA[t][iv2] ≠ null
15.           then continue;
16.         totComb ← totQry[iv1] + totQry[iv2] − corr[iv1][iv2];
17.         if SA[t][v1] == null e SA[t][v2] == null
18.           then minCombCost ← min({(totComb * 2 * dist[BS][r]) + dataCost[r][iv1] + dataCost[r][iv2] | r ∈
                  (S ∪ {BS})});
19.                 minRepComb ← repository with minCombCost;
20.           else  if SA[t][iv1] ≠ null
21.                   then minRepComb ← SA[t][iv1];
22.                   else  minRepComb ← SA[t][iv2];
23.                 minCombCost ← (totComb * 2 * dist[BS][minRepComp]) + dataCost[minRepComb][iv1] +
                    dataCost[minRepComb][iv2];
24.         costSep ← minCost[iv1] + minCost[iv2];
25.         if minCombCost < costSep
26.           then SA[t][iv1] ← minRepComb;
27.                SA[t][iv2] ← minRepComb;
28. for all intervals iv ∈ I
29.         if SA[t][iv] == null
30.           then SA[t][iv] ← minRep[iv];
```

Fig. 2. Algorithm to calculate an SA

choosing a repository r for storing both iv_1 and iv_2 is given by $combQryCost[r] = totComb * 2 * dist[BS][r]$. For computing the total cost of choosing a repository r for both we also need to add to $combQryCost[r]$ the cost of sending value updates in both intervals to r ($dataCost[r][iv_1]$ and $dataCost[r][iv_2]$). The repository with the minimum overall cost (*minCombCost*) is chosen as the candidate (*minRepComp*) for both iv_1 and iv_2 (Lines 18 and 19). However, *minCombCost* may still be higher than storing the intervals independently (*costSep*) and thus they are compared for deciding whether the repositories for iv_1 and iv_2 should be merged (Lines 25 to 27). It is possible that when considering a pair, one of them has already been assigned to a merged repository (Lines 20 to 23). If this is the case, the algorithm checks if it is worth combining a third interval into the same repository following the same strategy.

Algorithm *SACalculation* runs in $O(|S|^2N^2log(N))$ time, where $|S|$ is the number of sensors and N is the number of intervals in an SA. The computation of *dataCost* and *qryCost* takes $|S|^2N$ time, since for every interval iv and sensor r, it computes the time for every sensor that may produce a value in iv to update its value in r. Lines 8 to 11 is $O(|S|N)$. It takes time $|S|$ to find a minimum value and this is done for each interval in the SA. Observe that the size of the matrix *corr* is $|N|^2/2$, since only half of the cells are filled. Thus, sorting this set in Line 12 takes $O(N^2log(N))$. In

Lines 13 to 27, in order to find useful co-occurrences the algorithm scans each pair in this set, and for each of them looks for the minimum cost repository among the set S. This takes $|S|(N^2/2)$ time, and the last lines (28 to 30) takes time N. Thus, the execution time of the algorithm is $O(|S|^2N + |S|N + N^2log(N) + |S|(N^2/2) + N)$, which is $O(|S|^2N^2log(N))$.

Observe that the DYSTO model is based on several tunable parameters so that it can be adjusted to distinct WSN scenarios, and thus providing a flexible and adaptable storage model. In the next section we present an experimental study that determines the impact of these user-defined parameters as well as query values co-occurrences on the number of transmissions in a WSN. Determining the impact of these parameters is an important step towards self-tuning mechanisms that would relieve the user from the task of manually setting them to fit the application needs.

4. EXPERIMENTAL STUDY

DYSTO was implemented on NS2 network simulator version 2.34[1] . The goal of the proposed model is the reduction of the overall energy consumption. Given that energy consumption is dominated by the communication overhead [Pottie and Kaiser 2000], our cost metric is the total number of transmissions sensors send collectively.

DYSTO has been inspired by the data placement strategy proposed by the SCOOP system [Gil and Madden 2007], and thus the goal of our experimental study is to compare DYSTO against SCOOP using the number of transmissions as the cost metric for three experiments. Recall that in our model there are 4 types of messages that are transmitted among network devices: *mapping messages*, containing a new SA, which are sent from the BS to all sensors on the field; *summary messages*, containing a data histogram and related information, which are sent from each sensor to the BS; *data messages*, containing sensed data, sent from a producer sensor to a repository; and *query/reply messages*, sent from the BS to repositories and back with the query reply.

In our initial simulation runnings, we have noticed that the number of mapping messages remained constant in every setting, depending only on the number of SA calculations. Observe that the frequency of SA calculations determine how quickly the system reacts to changes on the behavior of the monitored system. Given this, our experiments have been conducted for determining the impact of each of our strategies on the remaining types of messages. The first experiment determines the impact of considering data co-occurrences in queries on the number of data and query/reply messages. The second has been conducted to show how a data threshold (Th_d), which determines the frequency sensed data updates are transmitted, affects the volume of data messages compared to fixed time interval transmissions. The third experiment determines the effect of the histogram threshold (Th_h) on summary, data and query/reply messages. The fourth experiment shows the impact of piggybacking in the reduction of summary messages. The last experiment analyze the impact of the four previous strategies considered together on the overall number of transmissions of each message type.

4.1 Simulation Settings

We evaluated the performance of DYSTO using a real data set made available by the Intel Lab Data[2]. The trace was collected from 54 Mica2 motes deployed in the Intel Research Lab over a period of 35 days. The data set comprises a timestamp and information on temperature, humidity, light, and voltage. However, only temperature measures along with sensor locality information were used in our simulations. We refer to this setting as the *real scenario*.

In order to evaluate the model on a larger WSN, we have also generated a *synthetic scenario* composed of 500 sensors on a 500×500 square meter field, with similar characteristics on their sensing

[1] `http://www.isi.edu/nsnam/ns/`
[2] `http://db.csail.mit.edu/labdata/labdata.html`

data as the real trace. Based on the observation that the metrics collected in the real scenario present highly spatially correlated readings, we have generated the sensors' initial readings using a Matlab tool [Jindal and Psounis 2006], which has been especially designed to produce data with this property. The tool receives as input a correlation coefficient (h) and the size of the monitoring area (m). It generates as output a matrix D of dimension $m \times m$, used to determine sensors readings as follows: each sensor s, randomly placed at a position (x_s, y_s), gets as reading the value at $D[\lfloor x_s \rfloor], \lfloor y_s \rfloor]$. The correlation coefficient determines the level of similarity. That is, $h = 0$ generates data with no spatial similarity, and higher values of h induces higher spatial correlation. All simulations described in this section have been executed on a scenario generated with high correlation ($h = 9$). Each of the values initially calculated were then updated applying a random variation of zero to 30%, applied on the initial value, based on the sensor location in order to preserve the spatial similarity.

The simulation parameters are presented in Table I. Threshold values are not specified in the table since they vary in each experiment detailed in the following sections. All results reported consider these values as default both for SCOOP and DYSTO, unless otherwise specified, and consist of the average value collected from five executions on each simulation setting.

Table I. Simulation parameters

Parameter	real scenario	synthetic scenario
Network devices	54 sensors + 1 BS	500 sensors + 1 BS
Data source	real data	synthetic data
Sensor communication range	30 meters	
Simulation duration	40 minutes	
Sample rate	1 sensor reading every 15 seconds	
Query rate	1 query every 15 seconds	
Summary rate	1 message every 110 seconds	
SA calculation	1 calculation every 240 seconds	
Number of entries in an SA (N)	15	
Number of readings in each sensor (hR)	30	
Number of entries in the sensor histogram (hS)	10	
Maximum piggybacking waiting time ($maxP$)	15 seconds	
Piggybacking decay constant (λ)	0.7	

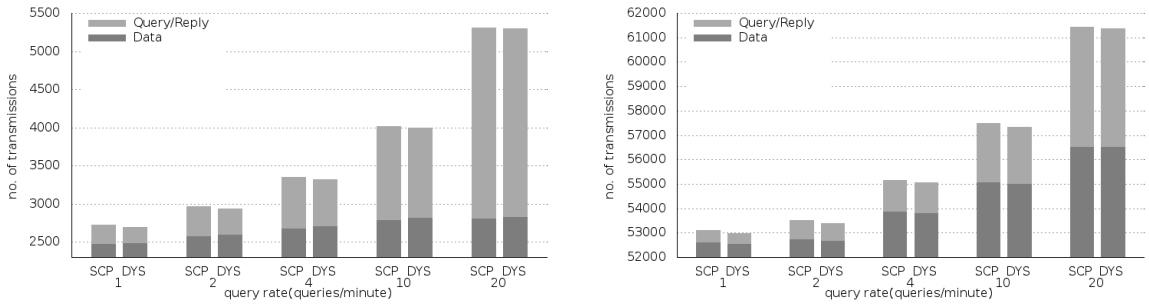
4.2 Experiment 1: Co-occurrences

The purpose of this experiment is to determine the effect of considering query data co-occurrences for determining data placement. We compare DYSTO with SCOOP, which follows the same approach as DYSTO for choosing repositories without considering this strategy. The simulations were executed with the same input parameters for both systems, as presented in Table I. We consider a query workload in which the initial value of the queried range is randomly chosen and 80% of the queries are correlated. That is, 80% of the range queries require access to more than one repository. Figure 3 shows the number of transmissions for data and query/reply messages for query rates of 1, 2, 4, 10 and 20 queries per minute. Transmissions for other types of messages are not shown because they are not affected by the query co-occurrences strategy.

The graphs show that in both SCOOP and DYSTO the number of transmissions for query/reply messages increases with higher query rates, as expected. The growth on data messages transmissions is because repositories are moved closer to the BS (and further from the producer sensors) when the query rate increases. However, this change on the repository placement has a higher impact on the number of data messages in the synthetic scenario. This is because moving a repository only 1 hop away from the producers may incur in an increase of 500 transmissions; that is, one for each sensor producer. In the real scenario, on the other hand, the impact is at most 54, which is the size of the WSN.

From the simulations on the real scenario presented in Figure 3(a), it can be observed that DYSTO's data co-occurrences strategy results in a reduction on query/reply transmissions of 18.75%, 13.28%, 8.18% 3.84% and 0,96% compared to SCOOP, for rates of 1, 2, 4, 10 and 20 queries per minute, respectively. On the other hand, it increases the number of transmissions of data messages by 0.40%, 0.77%, 1.22%, 1.18% and 0.81% respectively. For the synthetic scenario (Figure 3(b)), the number of data messages transmissions are almost identical in both systems, and for query/reply messages we can observe decreases of 15.56%, 9.32%, 6.2%, 3.6% for query rates of 1, 2, 4, and 10, respectively. With 20 queries per minute, both systems present the same behavior. The steadiness on the number of data queries in this scenario, as opposed to its increase in the real scenario, can be explained as follows. On larger WSNs there may exist several sensors with data costs close to the minimum value. Thus, taking query value co-occurrences into consideration helps the repository selection among these sensors without affecting the number of data messages. On smaller WSNs, on the other hand, such an alternative sensor may not exist.

Observe that in both scenarios, with more queries being issued to the system, the DYSTO gain on query/reply transmissions decreases. This is because when choosing a repository, both the cost of sending reading updates (*dataCost* in algorithm *SACalculation*) and the query cost (*qryCost*) are taken into consideration. As both graphs in Figure 3 show, the impact of the *dataCost* increases with the query rate. Thus, for higher query rates it gets harder to find repositories that when merged, based on query value co-occurrences, results on a reduction of the *qryCost* that surpasses the increase on the *dataCost*. Moreover, the impact of the *dataCost* is higher for larger WSNs since each sensor individually transmits a data message periodically to update its repository. This explains why for the real scenario the co-occurrence strategy presents better results than for the synthetic scenario. The query value co-occurrence strategy can be considered a refinement for the repository selection problem which provides better results than SCOOP for query rates up to 10 queries/minute, and similar results for higher query rates.



(a) Real scenario (b) Synthetic scenario

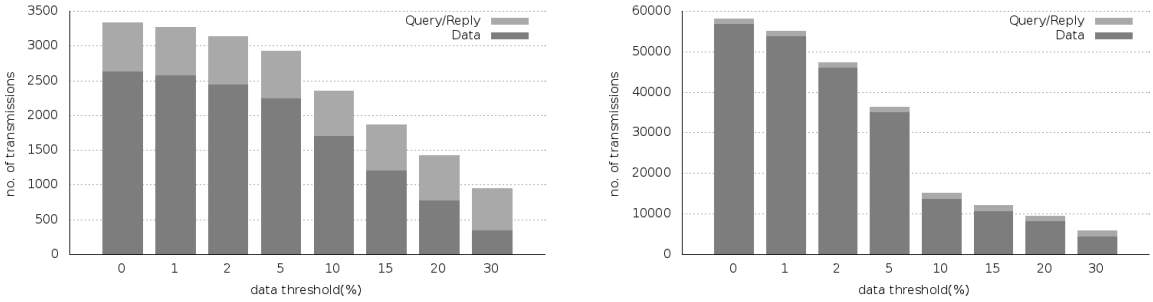
Fig. 3. Co-occurrences simulation

4.3 Experiment 2: Data Threshold Th_d

Among the 4 types of messages, data messages are the ones with higher impact on the overall energy consumption, as illustrated in Figure 7. Thus, we have conducted simulations to assess the impact of using a data threshold Th_d for determining the frequency in which a sensor updates its reading in a repository, as opposed to fixed intervals between data messages transmissions. The results are presented in Figure 4, where the values reported for a data threshold of zero corresponds to the SCOOP default setup of one message every 75 seconds.

It can be observed that compared to fixed intervals, a data threshold of 1% decreases the number of transmissions by roughly 2% for the real scenario and 5% for the synthetic one, and there is a steady decrease on these numbers with higher values for Th_d. For instance, with $Th_d = 5\%$ they decrease by

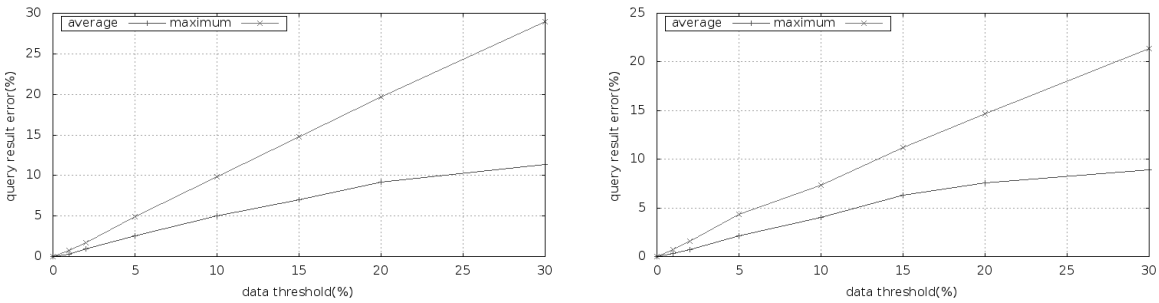
15% and 38% for the real and synthetic scenarios, respectively. Moreover, there is a small decrease on the number of query/reply transmissions. This is because the reduction on data transmissions also reduces the production rate (*prodRate*) of the sensor monitored by the BS. As a consequence, the weight of the data cost is reduced, compared to the query cost on the repository selection process. Thus, repositories are moved closer to the BS, reducing the number of transmissions for query/reply. Observe that, similar to the experiment described in Section 4.2, the number of data messages in the synthetic scenario is proportionally much larger than query/reply messages, compared to the real scenario. This is a consequence of the size of the WSN; that is, every sensor send a data message with their readings updates and thus the number of data messages increases with the size of the WSN. However, in both scenarios, we kept the same query rate of four queries/second. Thus, the ratio of query/reply messages in the total number of transmissions is much smaller in the synthetic scenario.



(a) Real scenario (b) Synthetic scenario

Fig. 4. Data threshold simulation

Adjusting the frequency of data message transmissions based on a threshold presents a positive impact by reducing the number of transmissions. However, this strategy may have an impact on the accuracy of the query results. Since in a simulation we can have the exact query result set, in Figure 5 we present both the average and maximum relative errors, by comparing the query result set, based on the values stored in a repository, with the actual readings stored at each sensor. These results show that the average error on the query results is not higher than half of Th_d for both scenarios, but it can be as high as Th_d. The relative error for the synthetic scenario is slightly smaller than for the real scenario because in the real scenario there may exist abrupt changes in the readings, while in the synthetic scenario we applied an update of at most 30% on the initial value, which is spatially correlated. Thus, in general, the difference between two consecutive readings is unlikely to differ by more than 20%. The tradeoff between the cost of data transmissions and the query result accuracy has to be considered for each application.



(a) Real scenario (b) Synthetic scenario

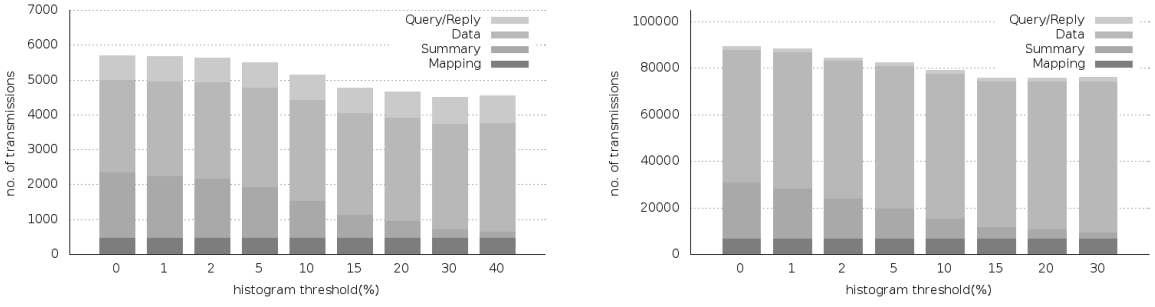
Fig. 5. Relative sensor data error due to data threshold

4.4 Experiment 3: Histogram Threshold Th_h

Similar to the previous experiment, we have also conducted simulations to compare the effect of adopting a threshold-based approach for determining the frequency for transmitting summary messages, compared to fixed intervals. It is worth noticing that the cost of summary messages, compared to data messages is lower because of our piggybacking strategy for merging them on their way to the BS.

The simulation results with varying values for the histogram threshold (Th_h) are presented in Figure 6. It compares the number of transmissions within fixed intervals of 110 seconds (as considered by SCOOP), which is reported as the bar for $Th_h = 0$, with transmissions based on increasing values of Th_h. With $Th_h = 1\%$ the number of summary transmissions reduces by 6.65% for the real scenario and 12% for the synthetic scenario, compared to the cost of fixed intervals. The reduction is higher for higher values of Th_h. For instance, with $Th_h = 20\%$, it reduces by 73.5% for the real scenario and 83% for the synthetic one. Although the reduction on the number of summary messages is considerable, the same reduction does not extend to the overall number of transmissions. This is because changes on summary messages frequency also have an impact on both data and query/reply messages. With fewer summary information, the BS calculates new SAs based on older data. Thus, updates on repository placement may differ from the one based on more recent information. In fact, higher values for Th_h increase both the number of data and query/reply transmissions, showing that repositories are placed further away from its ideal location. Indeed, as shown in Figure 6, with Th_h set to 30%, the reduction on the overall number of transmissions is 21% in the real scenario, but the reduction is only 20% for $Th_h = 40\%$. This shows that at 40%, the increase on the number of data and query/reply messages due to SAs computed with outdated information has a bigger impact on the overall cost than the reduction on the transmission of summary messages. The same phenomenon can be observed in the synthetic scenario: the overall reduction is 15% for $Th_h = 20\%$, and 14% for $Th_h = 30\%$. The maximum histogram threshold that is beneficial to the reduction of the overall number of transmissions is smaller for larger WSNs because the volume of data messages increases faster for larger WSNs.

Observe that, unlike the data threshold Th_d, the adoption of a histogram threshold for determining the frequency of summary messages has no impact on the accuracy of the query results. The strategy main impact is on postponing updates on repository placements, but repositories contain the current readings reported by the sensor devices.



(a) Real scenario (b) Synthetic scenario

Fig. 6. Histogram threshold simulation

4.5 Experiment 4: Piggybacking

This experiment is intended to study the effect of several configurations on piggybacking parameters and their impact on the overall number of summary transmissions. Since every sensor send summary messages to the BS, without considering any form of data aggregation, messages of this type represent a huge weight on the system in terms of energy consumption. Thus, we adopt a piggybacking approach

Table II. The effect of piggybacking

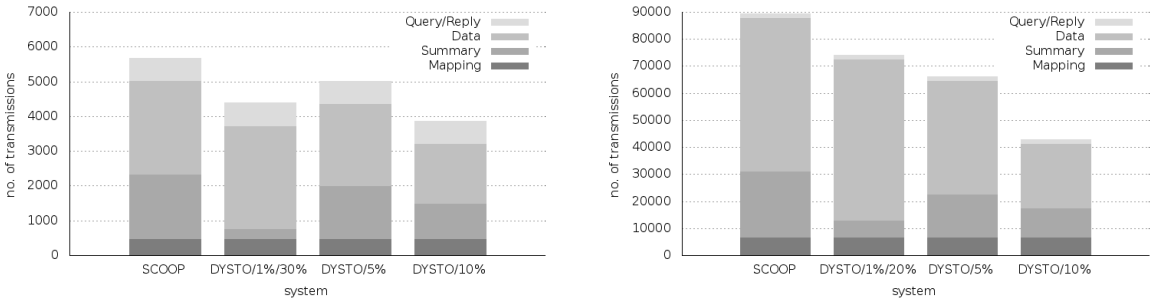
λ	# Summary transmissions	Avg. transmission time(s)
No Piggybacking	2666	15.20
0.1	2225	26.85
0.2	2194	28.81
0.3	2110	31.05
0.4	2009	34.09
0.5	1992	34.88
0.6	1860	40.34
0.7	1816	42.78
0.8	1792	60.23
0.9	1849	188.89

that works as follows. After computing a new summary, each sensor waits for a certain period of time before transmitting it. If during this time it receives summary messages to be routed to its ascendant on a path to the BS, they are concatenated in a single message, which is sent to the BS after the waiting time expires. As presented earlier, the function that determines this time ($P[s]$) is given by $P[s] = maxP \times e^{-\lambda \times (dist[BS][s]-1)}$ where $maxP$ is the maximum waiting time, which in this experiment is set to 15 seconds, λ is the decay constant and $dist[BS][s]$ is the number of hops between the BS and the sensor s. The simulation results on the real scenario, with varying values for λ, are presented in Table II. The first column contains the λ value, the second one presents the total number of summary message transmissions during the entire simulation, and the third, the average time a summary message transmitted from the furthest sensor from the BS (denoted as s_f) takes to reach the BS. For $\lambda = 0.1$, the number of transmissions reduces by 16.54% compared to the cost of using no piggybacking. The number of transmissions decreases gradually until $\lambda = 0.8$, which represents a reduction of 32.78%. While the number of transmissions decreases, the average time to route a message from s_f to the BS increases from 76.64% for $\lambda = 0.1$ to 1142% for $\lambda = 0.9$. In the experiments reported in the previous sections, we have chosen a λ value of 0.7, since it presents a good cost/benefit and because the extra time needed for routing summary messages does not affect the accuracy of the results, but only the time the system takes to update the repository placement in order to adjust to changes on the sensors' behavior in terms of production values and rates.

4.6 Experiment 5: Overall

Figure 7 shows a final comparison between SCOOP and DYSTO, considering the strategies of query data co-occurrences, data threshold and histogram threshold combined. SCOOP has been set with the same parameters as reported in the previous experiments, while we report results for DYSTO using the following settings: a) data threshold (Th_d) of 1% combined with a histogram threshold (Th_h) of 30% for the real scenario and 20% for the synthetic scenario; b) both thresholds set to 5%; and c) both thresholds set to 10%. The first setting has been chosen combining a value for Th_d that produces query results with maximum relative error of 1% and 0.5% in average, with a value for Th_h that produces the maximum reduction on the total number of transmissions, based on the experiments reported in Section 4.4.

Compared to SCOOP, reductions on the overall number of transmissions provided by DYSTO for the real scenario are 22.72%, 11.74% and 32.15%, and for the synthetic scenario, 17.02%, 26.01% and 52.15% for settings a, b, and c, respectively. These results show the potential benefits of our proposal. They also show that even applications that require high precision on the query results can benefit from the ability to define thresholds that are best suited for the underlying WSN. Although all the results presented in this article are based on simulations, they allow one to have a good prediction of their performance in the real world, as reported in previous works [Gil and Madden 2007].



(a) Real scenario (b) Synthetic scenario

Fig. 7. Comparison between SCOOP and DYSTO

5. CONCLUSION

In this article we proposed DYSTO, a storage model for WSNs that collects information on data production and query rates in order to dynamically change the data placement that is better suited for the current context. The data placement may adopt a local, external or data-centric model depending on the system's behavior. If the query rate increases then sensed data are stored closer to the base station; on the other hand, if the sensors' production rates increase then data are stored near the producer sensor. To achieve this adaptability, the base station periodically calculates a storage assignment that optimizes the overall communication costs. DYSTO extends SCOOP [Gil and Madden 2007], a previously proposed adaptable model, with the following strategies: analysis of data query co-occurrences for selecting repositories, and adoption of thresholds for determining the transmission rates of two types of messages originated at sensor devices: sensing data updates, that are sent to a repository, and summary reports, that are sent to the base station. We have run experiments, based on simulations, both on a real data set and on a larger WSN with synthetically generated sensed data. On the real scenario, DYSTO presented a reduction on the overall transmission costs of 12% and 32% for thresholds set to 5% and 10%, respectively, compared to SCOOP. For the synthetic scenario, it reduces the number of transmissions by 26% and 52%, with the same threshold settings. These results show the potential benefits of our proposed strategies. Although in this article we have applied these strategies to improve the storage assignment generated by SCOOP [Gil and Madden 2007], they can be generalized to be considered for other data placement strategies proposed for WSNs [Yu et al. 2010].

There are a number of extensions to DYSTO we intend to pursue in the future. First, in this article we have considered storage assignments (SAs) based on a single measurement collected by the sensor devices. A simple extension to the proposed model in order to manage multiple measurements is to generate multiple SAs, one for each measurement. This would help the processing of range queries involving any of the measurements. However, similar to multi-attribute index structures proposed for databases, we can envision a number of benefits of generating SAs that map multi-attribute values to repositories, such as the reduction on data messages. Thus, multi-attribute SAs is an interesting topic of future investigation. The experiments reported in this article show that system monitoring messages have a big impact on the overall energy consumption. A policy-based approach for dynamically adjusting these frequencies seems to be a promising approach, which would relieve the user from the responsibility of manually setting these parameters to suit the application requirements. Other future work include: define a cost measurement for dynamically adjusting the frequency of transmissions of new SAs; determine the effect of adopting equidepth histograms, as opposed to the current equiwidth approach; and conduct experiments on real WSNs.

REFERENCES

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine* 40 (8): 102 – 114, 2002.

Ammar, K. and Nascimento, M. A. Histogram and other aggregate queries in wireless sensor networks. In *Proceedings of the International Conference on Scientific and Statistical Databases Management*. Portland, OR, USA, pp. 527–536, 2011.

Brayner, A., Lopes, A., Meira, D., Vasconcelos, R., and Menezes, R. An adaptive in-network aggregation operator for query processing in wireless sensor networks. *Journal of Systems and Software* 81 (3): 328–342, 2008.

Coman, A. and Nascimento, M. A. A distributed algorithm for joins in sensor networks. In *Proceedings of the International Conference on Scientific and Statistical Databases Management*. Banff, Canada, pp. 27, 2007.

Fangfang, L., Zhibo, F., Chuanwen, L., Jia, X., Ge, Y., and Shenyang, C. A data storage method based on multilevel mapping index in wireless sensor networks. In *International Conference on Wireless Communications, Networking and Mobile Computing*. Shanghai, China, pp. 2747 –2750, 2007.

Ganesan, D., Estrin, D., and Heidemann, J. Dimensions: why do we need a new data handling architecture for sensor networks? *ACM SIGCOMM Computer Communication Review* 33 (1): 143–148, 2003.

Gil, T. M. and Madden, S. Scoop: An adaptive indexing scheme for stored data in sensor networks. In *Proceedings of the IEEE International Conference on Data Engineering*, R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis (Eds.). Istanbul, Turkey, pp. 1345–1349, 2007.

Greenstein, B., Ratnasamy, S., Shenker, S., Govindan, R., and Estrin, D. Difs: a distributed index for features in sensor networks. *Ad Hoc Networks* 1 (2-3): 333–349, 2003.

Gupta, S. and Dave, M. Real time approach for data placement in wireless sensor networks. *International Journal of Electronic Circuits System* 2 (3): 132–139, 2008.

Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking*, R. L. Pickholtz, S. K. Das, R. Cáceres, and J. J. Garcia-Luna-Aceves (Eds.). Boston, MA, pp. 56–67, 2000.

Jindal, A. and Psounis, K. Modeling spatially correlated data in sensor networks. *ACM Transactions on Sensor Networks* vol. 2, pp. 466–499, 2006.

Li, X., Kim, Y.-J., Govindan, R., and Hong, W. Multi-dimensional range queries in sensor networks. In *Proceedings of the International Conference on Embedded Networked Sensor Systems*, I. F. Akyildiz, D. Estrin, D. E. Culler, and M. B. Srivastava (Eds.). Los Angeles, CA, pp. 63–75, 2003.

Matos, T. B., Brayner, A., and Maia, J. E. B. Towards in-network data prediction in wireless sensor networks. In *Proceedings of the ACM Symposium on Applied Computing*, S. Y. Shin, S. Ossowski, M. Schumacher, M. J. Palakal, and C.-C. Hung (Eds.). Sierre, Switzerland, pp. 592–596, 2010.

Nascimento, M. A., Alencar, R. A. E., and Brayner, A. Optimizing query processing in cache-aware wireless sensor networks. In *Proceedings of the International Conference on Scientific and Statistical Databases Management*, M. Gertz and B. Ludäscher (Eds.). Lecture Notes in Computer Science, vol. 6187. Springer, Heidelberg, Germany, pp. 60–77, 2010.

Pottie, G. J. and Kaiser, W. J. Wireless integrated network sensors. *Communications of the ACM* 43 (5): 51–58, 2000.

Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., and Yu, F. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile networks and applications* 8 (4): 427–442, 2003.

Shen, H., Zhao, L., and Li, Z. A Distributed Spatial-Temporal Similarity Data Storage Scheme in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing* 10 (7): 1–6, 2010.

Szewczyk, R., Polastre, J., Mainwaring, A., and Culler, D. Lessons from a sensor network expedition. In *Wireless Sensor Networks*, H. Karl, A. Wolisz, and A. Willig (Eds.). Lecture Notes in Computer Science, vol. 2920. Springer, pp. 307–322, 2004.

Tilak, S., Abu-Ghazaleh, N. B., and Heinzelman, W. A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communication Review* 6 (2): 29–36, 2002.

Yang, K.-C., Yang, Y.-C., Lin, C.-L., and Wang, J.-S. Hierarchical data management for spatial-temporal information in WSNs. In *Proceedings of the International Conference on Sensor Technologies and Applications*. Venice, Italy, pp. 435–440, 2010.

Yao, Y., Tang, X., and Lim, E.-P. In-network processing of nearest neighbor queries for wireless sensor networks. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, M.-L. Lee, K.-L. Tan, and V. Wuwongse (Eds.). Lecture Notes in Computer Science, vol. 3882. Springer, Singapore, pp. 35–49, 2006.

Yoon, S. and Shahabi, C. The clustered aggregation (CAG) technique leveraging spatial and temporal correlations in wireless sensor networks. *ACM Transactions on Sensor Networks* 3 (1): 3, 2007.

Yu, Z., Xiao, B., and Zhou, S. Achieving optimal data storage position in wireless sensor networks. *Computer Communications* 33 (1): 92–102, 2010.