TACO: Tunable Approximate Computation of Outliers in Wireless Sensor Networks

Nikos Giatrakos[*]
Dept. of Informatics
University of Piraeus
Piraeus,Greece
ngiatrak@unipi.gr

Yannis Kotidis[†]
Dept. of Informatics
Athens University of Economics and Business
Athens,Greece
kotidis@aueb.gr

Antonios Deligiannakis
Dept. of Electronic and Computer Engineering
Technical University of Crete
Crete,Greece
adeli@softnet.tuc.gr

Vasilis Vassalos
Dept. of Informatics
Athens University of Economics and Business
Athens,Greece
vassalos@aueb.gr

Yannis Theodoridis[*]
Dept. of Informatics
University of Piraeus
Piraeus,Greece
ytheod@unipi.gr

ABSTRACT

Wireless sensor networks are becoming increasingly popular for a variety of applications. Users are frequently faced with the surprising discovery that readings produced by the sensing elements of their motes are often contaminated with outliers. Outlier readings can severely affect applications that rely on timely and reliable sensory data in order to provide the desired functionality. As a consequence, there is a recent trend to explore how techniques that identify outlier values can be applied to sensory data cleaning. Unfortunately, most of these approaches incur an overwhelming communication overhead, which limits their practicality. In this paper we introduce an in-network outlier detection framework, based on locality sensitive hashing, extended with a novel boosting process as well as efficient load balancing and comparison pruning mechanisms. Our method trades off bandwidth for accuracy in a straightforward manner and supports many intuitive similarity metrics.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous;
H.2.8 [**Database Applications**]: Data Mining

General Terms

Algorithms, Design, Management, Measurement

Keywords

sensor networks, outliers

[*]Nikos Giatrakos and Yannis Theodoridis were partially supported by the EU FP7/ICT/FET Project MODAP.

[†]Yannis Kotidis was partially supported by the Basic Research Funding Program, Athens University of Economics and Business.

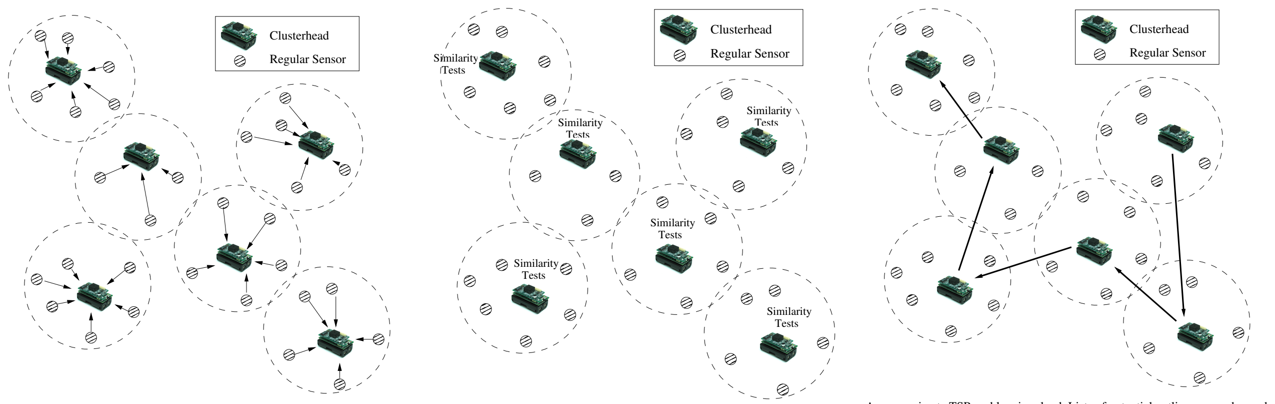
1. INTRODUCTION

Pervasive applications are increasingly supported by networked sensory devices that interact with people and themselves in order to provide the desired services and functionality. Because of the unattended nature of many applications and the inexpensive hardware used in the construction of the sensors, sensor nodes often generate imprecise individual readings due to interference or failures [14]. Sensors are also often exposed to severe conditions that adversely affect their sensing elements, thus yielding readings of low quality. For example, the humidity sensor on the popular MICA mote is very sensitive to rain drops [9].

The development of a flexible layer that will be able to detect and flag outlier readings, so that proper actions can be taken, constitutes a challenging task. Conventional outlier detection algorithms [2] are not suited for our distributed, resource-constrained environment of study. First, due to the limited memory capabilities of sensor nodes, in most sensor network applications, data is continuously collected by motes and maintained in memory for a limited amount of time. Moreover, due to the frequent change of the data distribution, results need to be generated continuously and computed based on recently collected measurements. Furthermore, a central collection of sensor data is not feasible nor desired, since it results in high energy drain, due to the large amounts of transmitted data. Hence, what is required are continuous, distributed and in-network approaches that reduce the communication cost and manage to prolong the network lifetime.

One can provide several definitions of what constitutes an outlier, depending on the application. For example in [27], an outlier is defined as an observation that is sufficiently far from most other observations in the data set. However, such a definition is inappropriate for physical measurements (like noise or temperature) whose absolute values depend on the distance of the sensor from the source of the event that triggers the measurements. Moreover, in many applications, one cannot reliably infer whether a reading should be classified as an outlier without considering the recent history of values obtained by the nodes. Thus, in our framework we propose a more general method that detects outlier readings taking into account the recent measurements of a node, as well as spatial correlations with measurements of other nodes.

Similar to recent proposals for processing declarative queries in wireless sensor networks, our techniques employ an *in-network*



Each sensor node uses LSH to encode its latest W measurements using just d bits. The node transmits the compressed d–bit representation to the clusterhead.

(a)

Each clusterhead performs similarity tests on the received meaurements. Nodes that do not gain enough support are considered as potential outliers.

(b)

An approximate TSP problem is solved. Lists of potential outliers are exchanged (along with their compressed representations and current support). The final list is transmitted to the base station.

(c)

Figure 1: Main Stages of the TACO Framework

processing paradigm that fuses individual sensor readings as they are transmitted towards a *base station*. This fusion, dramatically reduces the communication cost, often by orders of magnitude, resulting in prolonged network lifetime. While such an in-network paradigm is also used in proposed methods that address the issue of data cleaning of sensor readings by identifying and, possibly, removing outliers [6, 9, 14, 28], none of these existing techniques provides a straightforward mechanism for controlling the burden of the nodes that are assigned to the task of outlier detection.

An important observation that we make in this paper is that existing in-network processing techniques cannot reduce the volume of data transmitted in the network to a satisfactory level and lack the ability of tuning the resulting overhead according to the application needs and the accuracy levels required for outlier detection. Please note that it is desirable to reduce the amount of transmitted data in order to also significantly reduce the energy drain of sensor nodes. This occurs not only because radio operation is by far the biggest culprit in energy drain [21], but also because fewer data transmissions also result in fewer collisions and, thus, fewer re-transmissions by the sensor nodes.

In this paper we propose a novel outlier detection scheme termed TACO (TACO stands for Tunable Approximate Computation of Outliers). TACO adopts two levels of hashing mechanisms. The first is based on locality sensitive hashing (LSH) [5] which is a powerful method for dimensionality reduction [5, 12]. We first utilize LSH in order to encode the latest W measurements collected by each sensor node as a bitmap of $d \ll W$ bits. This encoding is performed locally at each node. The encoding that we utilize trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction and provides tunable accuracy guarantees based on the d parameter mentioned above. Assuming a clustered network organization [22, 31], motes communicate their bitmaps to their clusterhead, which can estimate the similarity amongst the latest values of any pair of sensors within its cluster by comparing their bitmaps, and for a variety of similarity metrics that are useful for the applications we consider. Based on the performed similarity tests, and a desired minimum support specified by the posed query, each clusterhead generates a list of *potential* outlier nodes within its cluster. At a second (inter-cluster) phase of the algorithm, this list is then communicated among the clusterheads, in order to allow potential outliers to gain support from measurements of nodes that lie within other clusters. This process is sketched in Figure 1.

The second level of hashing (omitted in Figure 1) adopted in TACO's framework comes during the intra-cluster communication phase. It is based on the hamming weight of sensor bitmaps and provides a pruning technique (regarding the number of performed bitmap comparisons) and a load balancing mechanism alleviating clusterheads from communication and processing overload. We choose to discuss this load balancing and comparison pruning mechanism separately for ease of exposition as well as to better exhibit its benefits.

The contributions of this paper can be summarized as follows:

1. We introduce TACO, an outlier detection framework which trades bandwidth for accuracy in a straightforward manner. TACO supports various popular similarity measures used in different application areas. Examples of such measures include, but are not limited to, the cosine similarity, the correlation coefficient, or the Jaccard coefficient.
2. We subsequently devise a boosting process that provably improves TACO's accuracy.
3. We devise novel load balancing and comparison pruning mechanisms, which alleviate clusterheads from excessive processing and communication load. These mechanisms result in a more uniform, intra-cluster power consumption and prolonged network unhindered operation, since the more evenly spread power consumption results in an infrequent need for network reorganization.
4. We present a detailed experimental analysis of our techniques for a variety of data sets and parameter settings. Our results demonstrate that our methods can reliably compute outliers, while at the same time significantly reducing the amount of transmitted data, with average recall and precision values exceeding 80% and often reaching 100%. It is important to emphasize that the above results often correspond to bandwidth consumptions that are lower than what is required by a simple continuous query, using a method like TAG [21]. We also demonstrate that TACO may result in prolonged network lifetime, up to a factor of 3 in our experiments. We further provide comparative results with the recently proposed technique of [9] that uses an equivalent outlier definition and supports common similarity measures. Overall, TACO appears to be more accurate up to 10% in terms of the F-Measure metric while ensuring less bandwidth consumption reaching the ratio of 1/8 in our study.

This paper proceeds as follows. In Section 2 we present related work. Section 3 presents our basic framework, while in Sections 4, 5 we analyze TACO's operation in detail. Section 6 presents our load balancing and comparison pruning mechanisms. Section 7 presents our experimental evaluation, while Section 8 includes concluding remarks.

2. RELATED WORK

The emergence of sensor networks as a viable and economically practical solution for monitoring and intelligent applications has prompted the research community to devote substantial effort to define and design the necessary primitives for data acquisition based on sensor networks [21, 30]. Different network organizations have been considered such as using hierarchical routes (i.e., the aggregation tree [8, 26, 32]), cluster formations [22, 31], or even completely ad-hoc formations [1, 17]. Our framework assumes a clustered network organization. Such networks have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime [22, 31].

Sensor networks can be rather unreliable, as the commodity hardware used in the development of the motes is prone to environmental interference and failures. The authors of [14] introduce a declarative data cleaning mechanism over data streams produced by the sensors. The work of [19] exploits localized data models that capture correlations among neighboring nodes, however the emphasis is on exploiting these models in order to reduce energy drain during query evaluation and not on outlier detection. The data cleaning technique presented in [33] makes use of a weighted moving average which takes into account both recent local samples and corresponding values by neighboring motes to estimate actual measurements. In other related work, [13] proposes a fuzzy approach to infer the correlation among readings from different sensors, assigns a confidence value to each of them, and then performs a fused weighted average scheme. A histogram-based method to detect outliers with reduced communication costs is presented in [24].

In [18], the authors discuss a framework for cleaning input data errors using integrity constraints. Of particular interest is the technique of [3] which proposes an unsupervised outlier detection technique so as to report the top-k values that exhibit the highest deviation in a network's global sample. The framework is flexible with respect to the outlier definition. However, in contrast to our framework, it provides no means of directly controlling the bandwidth consumption, thus often requiring comparable bandwidth to centralized approaches for outlier detection [3].

In [15], a probabilistic technique for cleaning RFID data streams is presented. The framework of [9] is used to identify and remove outliers during the computation of aggregate and group-by queries. Its definition of what constitutes an outlier, based on the notion of minimum support and the use of recent history, is adopted in this paper by our framework. It further demonstrates that common similarity metrics such as the correlation coefficient and the Jaccard coefficient can capture the types of dirty data encountered by sensor network applications. In [27] the authors introduce a novel definition of an outlier, as an observation that is sufficiently far from most other observations in the data set. However, in cases where the motes observe physical quantities (such as noise levels, temperature) the absolute values of the readings acquired depend, for example, on the distance of the mote from the cause of the monitored event (i.e., a passing car or a fire respectively). Thus, correlations among readings in space and time are more important than the absolute values, used in [27].

The algorithms in [9, 14] provide no easily tunable parameters in order to limit the bandwidth consumed while detecting and processing outliers. On the contrary our framework has a direct way of controlling the number of bits used for encoding the values observed by the motes. While [9] takes a best effort approach for detecting possible outliers and [14] requires transferring all data to the base station in order to accurately report them, controlling the size of the encoding allows our framework to control the accuracy of the outlier detection process.

The works in [6, 28] address the problem of identifying faulty sensors using localized voting protocols. However, localized voting schemes are prone to errors when motes that observe interesting events generating outlier readings are not in direct communication [9]. Furthermore, the framework of [28] requires a *correlation network* to be maintained, while our algorithms can be implemented on top of commonly used clustered network organizations.

The Locality Sensitive Hashing (LSH) scheme used in this work was initially introduced in the rounding scheme of [11] to provide solutions to the MAX-CUT problem. Since then, LSH has been adopted in similarity estimation [5], clustering [23], approximate nearest neighbor queries [12] or indexing techniques for set value attributes [10].

3. BASIC FRAMEWORK

3.1 Target Application

As in [9], we do not aim to compute outliers based on a mote's latest readings but, instead, take into consideration its most recent measurements. In particular let u_i denote the latest W readings obtained by node S_i. Then, given a similarity metric $sim{:}R^W \rightarrow [0, 1]$ and a similarity threshold Φ we consider the readings by motes S_i and S_j similar if

$$sim(u_i, u_j) > \Phi. \tag{1}$$

In our framework, we classify a mote as an outlier if its latest W measurements are not found to be similar with the corresponding measurements of at least *minSup* other motes in the network. The parameter *minSup*, thus, dictates the minimum support (either in the form of an absolute, uniform value or as a percentage of motes, i.e per cluster) that the readings of the mote need to obtain by other motes in the network, using Equation 1. By allowing the user/application to control the value of *minSup*, our techniques are resilient to environments where spurious readings originate from multiple nodes at the same epoch, due to a multitude of different, and hence unpredictable, reasons. Our framework can also incorporate additional *witness criteria* based on non-dynamic grouping characteristics (such as the node identifier or its location), in order to limit, for each sensor, the set of nodes that are tested for similarity with it. For example, one may not want sensor nodes located in different floors to be able to witness each other's measurements.

3.2 Supported Similarity Metrics

The definition of an outlier, as presented in Section 3.1, is quite general to accommodate a number of intuitive similarity tests between the latest W readings of a pair of sensor nodes S_i and S_j. Examples of such similarity metrics include the *cosine similarity*, the *correlation coefficient* and the *Jaccard coefficient* [5, 9]. Table 1 demonstrates the formulas for computing three of the aforementioned metrics over the two vectors u_i, u_j containing the latest W readings of sensors S_i and S_j, respectively[1].

It is important to emphasize that our framework is not limited to using just one of the metrics presented in Table 1. On the contrary, as it will be explained in Section 4.1, any similarity metric satisfying a set of common criteria may be incorporated in our framework.

[1] $E(.)$, σ and $cov(.)$ in the table stand for mean, standard deviation and covariance, respectively.

Similarity Metric	Calculation of Similarity
Cosine Similarity	$cos(\theta(u_i, u_j)) = \frac{u_i \cdot u_j}{\|\|u_i\|\| \cdot \|\|u_j\|\|}$ $\Rightarrow \theta(u_i, u_j) = arccos \frac{u_i \cdot u_j}{\|\|u_i\|\| \cdot \|\|u_j\|\|}$
Correlation Coefficient	$r_{u_i,u_j} = \frac{cov(u_i,u_j)}{\sigma_{u_i}\sigma_{u_j}} =$ $= \frac{E(u_i u_j) - E(u_i)E(u_j)}{\sqrt{E(u_i^2) - E^2(u_i)}\sqrt{E(u_j^2) - E^2(u_j)}}$
Jaccard Coefficient	$J(u_i, u_j) = \frac{\|u_i \cap u_j\|}{\|u_i \cup u_j\|}$

Table 1: Computation of some supported similarity metrics between the vectors u_i, u_j containing the latest W measurements of nodes S_i and S_j.

3.3 Network Organization

We adopt an underlying network structure where motes are organized into clusters (shown as dotted circles in Figure 1). Queries are propagated by the base station to the clusterheads, which, in turn, disseminate these queries to sensors within their cluster.

Various algorithms [22, 31] have been proposed to clarify the details of cluster formation, as well as the clusterhead election and substitution (rotation) during the lifetime of the network. All these approaches have been shown to be efficient in terms of energy dissipation, thus resulting in prolonged network lifetime. The aforementioned algorithms differ in the way clusters and corresponding clusterheads are determined, though they all share common characteristics since they primarily base their decisions on the residual energy of the sensor nodes and their communication links.

An important aspect of our framework is that the choice of the clustering algorithm is orthogonal to our approach. Thus, any of the aforementioned algorithms can be incorporated in our framework. An additional advantage of our techniques is that it requires no prior state at clusterhead nodes, thus simplifying the processes of clusterhead rotation and re-election.

3.4 Operation of the Algorithm

We now outline the various steps involved in our TACO framework. These steps are depicted in Figure 1.

Step 1: Data Encoding and Reduction. At a first step, the sensor nodes encode their latest W measurements using a bitmap of d bits. In order to understand the operation of our framework, the actual details of this encoding are not important (they are presented in Section 4). What is important is that:

- As we will demonstrate, the similarity function between the measurements of any pair of sensor nodes can be evaluated using their encoded values, rather than using their uncompressed readings.
- The used encoding trades accuracy (i.e., probability of correctly determining whether a node is an outlier or not) for bandwidth, by simply varying the desired level of dimensionality reduction (i.e., parameter d mentioned above). Larger values of d result in increased probability that similarity tests performed on the encoded representation will reach the same decision as an alternative technique that would have used the uncompressed measurements instead.

After encoding its measurements, each sensor node transmits its encoded measurements to its clusterhead.

Step 2: Outlier Detection at the Cluster Level. Each clusterhead receives the encoded measurements of the sensors within its cluster. It then performs similarity tests amongst any pair of sensor nodes that may witness each other (please note that the posed query may have imposed restrictions on this issue), in order to determine nodes that cannot reach the desired support level and are, thus, considered to be outliers at a cluster level.

Step 3: Intercluster Communication. After processing the encoded measurements within its cluster, each clusterhead has determined a set of potential outliers, along with the support that it has computed for each of them. Some of these potential outliers may be able to receive support from sensor nodes belonging to other clusters. Thus, a communication phase is initiated where the potential outliers of each clusterhead are communicated (along with their current support) to other clusterheads in which their support may increase. Please note that depending on the restrictions of the posed queries, only a subset of the clusterheads may need to be reached. The communication problem is essentially modeled as a TSP problem, where the origin is the clusterhead itself, and the destination is the base station.

The extensible definition of an outlier in our framework enables the easy application of semantic constraints on the definition of outliers. For example, we may want to specify that only movement sensors trained on the *same location* are allowed to witness each other, or similarly that only readings from vibration sensors attached to identical engines in a machine room are comparable. Such static restrictions can be easily incorporated in our framework (i.e., by having clusterheads maintain the corresponding information, such as location and type, for each sensor id) and their evaluation is orthogonal to the techniques that we present in this paper.

Symbol	Description
S_i	the $i-th$ sensor node
u_i	the value vector of node S_i
W	tumble size (length of u_i)
$\theta(u_i, u_j)$	the angle between vectors u_i, u_j
X_i	the bitmap encoding produced after applying LSH to u_i
d	bitmap length
$D_h(X_i, X_j)$	the hamming distance between bitmaps X_i, X_j
Φ,Φ_θ, Φ_{D_h}	similarity threshold used, depending on representation
minSup	the minimum support parameter

Table 2: Notation used in this paper

4. DATA ENCODING AND REDUCTION

We now present the locality sensitive hashing scheme and explain how it can be utilized by TACO. Table 2 summarizes the notation used in this section. The corresponding definitions are presented in appropriate areas of the text.

4.1 Definition and Properties of LSH

A *Locality Sensitive Hashing scheme* is defined in [5] as a distribution on a family F of hash functions that operate on a set of objects, such that for two objects u_i, u_j:

$$P_{h \epsilon F}[h(u_i) = h(u_j)] = sim(u_i, u_j)$$

where $sim(u_i, u_j) \epsilon [0, 1]$ is some similarity measure. In [5] the following necessary properties for existence of an LSH family function for given similarity measures are proved:

LEMMA 1. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance function $1 - sim(u_i, u_j)$ satisfies the triangle inequality.*

LEMMA 2. *Given an LSH function family F corresponding to a similarity function $sim(u_i, u_j)$, we can obtain an LSH function family F' that maps objects to {0, 1} and corresponds to the similarity function $\frac{1+sim(u_i,u_j)}{2}$.*

LEMMA 3. *For any similarity function $sim(u_i, u_j)$ that admits an LSH function family, the distance function $1 - sim(u_i, u_j)$ is isometrically embeddable in the hamming cube.*

4.2 TACO at the Sensor Level

In our setting, TACO applies LSH to the value vectors of physical quantities sampled by motes. It can be easily deduced that LSH schemes have the property of dimensionality reduction while preserving similarity between these vectors. Dimensionality reduction can be achieved by introducing a hash function family such that (Lemmas 2,3) for any vector $u_i \epsilon R^W$ consisting of W sampled quantities, $h(u_i) : R^W \to [0,1]^d$ with $d \ll W$.

In what follows we first describe an LSH scheme for estimating the cosine similarity between motes (please refer to Table 1 for the definition of the cosine similarity metric).

THEOREM 1 (RANDOM HYPERPLANE PROJECTION [5, 11]). *Assume we are given a collection of vectors defined on the W dimensional space. We choose a family of hash functions as follows: We produce a spherically symmetric random vector r of unit length from this W dimensional space. We define a hash function h_r as:*

$$h_r(u_i) = \begin{cases} 1 & \text{,if } r \cdot u_i \geq 0 \\ 0 & \text{,if } r \cdot u_i < 0 \end{cases}$$

For any two vectors $u_i, u_j \epsilon R^W$:

$$P = P[h_r(u_i) = h_r(u_j)] = 1 - \frac{\theta(u_i, u_j)}{\pi} \square \tag{2}$$

Equation 2 can be rewritten as:

$$\theta(u_i, u_j) = \pi \cdot (1 - P) \tag{3}$$

Note that Equation 3 expresses theta similarity as the product of the potential range of the angle between the two vectors (π), with the probability of equality in the result of the hash function application (P). Thus, after repeating a stochastic procedure using d random vectors r, the final embodiment in the hamming cube results in [29]:

$$D_h(X_i, X_j) = d \cdot (1 - P) \tag{4}$$

where $X_i, X_j \epsilon [0,1]^d$ are the bitmaps (of length d) produced and $D_h(X_i, X_j) = \sum_{\ell=1}^{d} |X_{i\ell} - X_{j\ell}|$ is their hamming distance. Hence, we finally derive:

$$\frac{\theta(u_i, u_j)}{\pi} = \frac{D_h(X_i, X_j)}{d} \tag{5}$$

This equation provides the means to compute the angle (and thus the cosine similarity) between the initial value vectors based on the hamming distance of their corresponding bitmaps. We will revisit this issue in the next section.

Let $E(u_i)$ denote the mean value of vector u_i. Simple calculations show that for $u_i^* = u_i - E(u_i)$ and $u_j^* = u_j - E(u_j)$ the equality $corr(u_i, u_j) = corr(u_i^*, u_j^*) = cos(\theta(u_i^*, u_j^*))$ holds, where *corr* denotes the correlation coefficient (see Table 1). As a result, the correlation coefficient can also be used as a similarity measure in a random hyperplain projection LSH scheme (i.e., by using the same family of hashing functions as with the cosine similarity). In [10] the authors also introduce an LSH scheme based on the Jaccard Index: $J(u_i, u_j) = \frac{|u_i \cap u_j|}{|u_i \cup u_j|}$ using minwise independent permutations and simplex codes.

We further note [5] that there exist popular similarity metrics that do not accept an LSH scheme. For instance, Lemma 1 implies

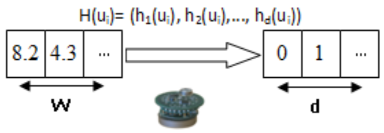


Figure 2: LSH application to mote's value vector

that there is no LSH scheme for the $Dice(u_i, u_j) = \frac{2|u_i \cap u_j|}{|u_i| + |u_j|}$ and $Overlap(u_i, u_j) = \frac{|u_i \cap u_j|}{min(|u_i|, |u_j|)}$ coefficients since they do not satisfy the triangle inequality.

In what follows, we will use as a running example the case where the cosine similarity between two vectors u_i and u_j is chosen, and where the vectors are considered similar when $sim(u_i, u_j) > \Phi \Leftrightarrow \theta(u_i, u_j) \leq \Phi_\theta$, where the Φ (or Φ_θ) threshold is defined by the query.

5. DETECTING OUTLIERS WITH TACO

5.1 Intra-Cluster Processing

As discussed in the introductory section, often outlying values cannot be reliably deduced without considering the correlation of a sensor's recent measurements with those of other sensor nodes. Hereafter, we propose a generic technique that takes into account the aforementioned parameters providing an energy efficient way to detect outlying values. To achieve that, we will take advantage of the underlying network structure and the random hyperplane projection LSH scheme.

Recalling Section 4.2, we consider a sampling procedure where motes keep W recent measurements in a tumble [4]. Sending a W-dimensional value vector as is, exacerbates the communication cost, which is an important factor that impacts the network lifetime. TACO thus applies LSH in order to reduce the amount of transmitted data. In particular, after having collected W values, each mote applies d h_r functions on it so as to derive a bitmap of length d (Figure 2), where the ratio of d to the size of the W measurements determines the achieved reduction. The derived bitmap is then transmitted to the corresponding clusterhead.

In the next phase, each clusterhead is expected to report outlying values. To do so, it would need to compare pairs of received vectors determining their similarity based on Equation 1 and the Φ_θ similarity threshold. On the contrary, the information that reaches the clusterhead is in the form of compact bitmap representations. Note that Equation 5 provides a way to express theta similarity in terms of the hamming distance and the similarity threshold $\Phi_{D_h} = d \cdot \frac{\Phi_\theta}{\pi}$. Thus, clusterheads can obtain an approximation of the initial similarity by examining the hamming distance between pairs of bitmaps. If the hamming distance between the two bitmaps is lower or equal to Φ_{D_h}, then the two initial vectors will be considered similar, and each sensor in the tested pair will be able to witness the measurements of the other sensor, thus being able to increase its support by 1. At the end of the procedure, each clusterhead determines a set of potential outliers within its cluster, and extracts a list of triplets in the form $\langle S_i, X_i, support \rangle$ containing for each outlier S_i its bitmap X_i and the current support that X_i has achieved so far.

Recall that given two vectors u_i, u_j, the probability P that the corresponding bits in their bitmap encoding are equal is given by Equation 2. Thus, the probability of satisfying the similarity test, via LSH manipulation is given by the cumulative function of a binomial distribution:

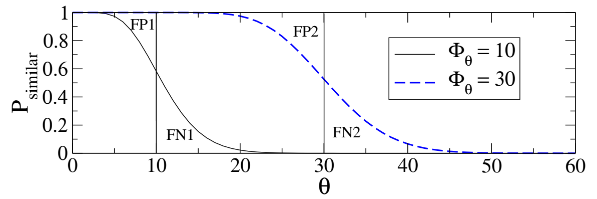


Figure 3: Probability $P_{similar}$ of judging two bitmaps as similar, depending on the angle (θ) of the initial vectors and for two different thresholds Φ_θ (W=16, reduction ratio=1/4).

$$P_{similar} = \sum_{i=0}^{\Phi_{D_h}} \binom{d}{i} P^{d-i} \cdot (1-P)^i \qquad (6)$$

Depending on whether the initial vectors are similar ($\theta(u_i, u_j) \leq \Phi_\theta$) or not, we can, therefore, estimate the expected false positive and false negative rate of the similarity test. As an example, Figure 3 plots the value of $P_{similar}$ as a function of $\theta(u_i, u_j)$ (recall that P is a function of the angle between the vectors) for two different values of Φ_θ. The area FP_1 above the line on the left denotes the probability of classifying the two vectors as dissimilar, even though their $theta$ angle is less than the threshold (false positive, when outlier detection is considered). Similarly, the area FN_1 denotes the probability of classifying the encodings as similar, when the corresponding initial vectors are not (false negative). The areas denoted as FP_2 and FN_2 correspond to the same probabilities for an increased value of Φ_θ. We can observe that the method is more accurate (i.e., leads to smaller areas for false positive and negative detections) for more strict definitions of an outlier implied by smaller Φ_θ thresholds.

In Figure 4 we depict the probability that TACO correctly identifies two similar vectors as similar, varying the length d of the bitmap. Using more LSH hashing functions (i.e choosing a higher d), increases the probability of resulting in a successful test.

5.2 Inter-Cluster Processing

Local outlier lists extracted by clusterheads take into account both the recent history of values and the neighborhood similarity (i.e., motes with similar measurements in the same cluster). However, this list of outliers is not final, as the posed query may have specified that a mote may also be witnessed by motes assigned to different clusterheads. Thus, an *inter-cluster communication* phase must take place, in which each clusterhead communicates information (i.e., its produced triplets) regarding its local, potential outlier motes that do not satisfy the *minSup* parameter. In addition, if the required support is different for each sensor (i.e., $minSup$ is expressed as a percentage of nodes in the cluster), then the $minSup$ parameter for each potential outlier also needs to be transmitted. Please note that the number of potential outliers is expected to only be a small portion of the total motes participating in a cluster.

During the inter-cluster communication phase each clusterhead transmits its potential outliers to those clusterheads where its locally determined outliers may increase their support (based on the restrictions of the query). This requires computing a circular network path by solving a TSP problem that has as origin the clusterhead, as endpoint the base station, and as intermediate nodes those sensors that may help increase the support of this clusterhead's potential outliers. The TSP can be computed either by the basestation after clusterhead election, or in an approximate way by imposing GPSR [16] to aid clusterheads make locally optimal routing de-

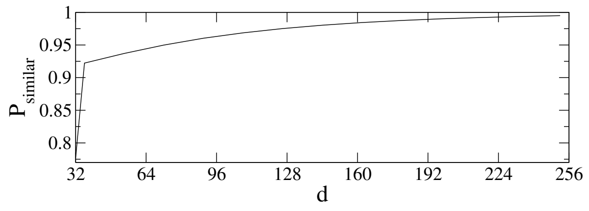


Figure 4: Probability $P_{similar}$ of judging two bitmaps (of vectors that pass the similarity test) as similar, depending on the number of bits d used in the LSH encoding (W=16, θ=5, Φ_θ=10)

cisions. However, note that such a greedy algorithm for TSP may result in the worst route for certain point - clusterhead distributions.

Any set $PotOut$ of potential outliers received by a clusterhead C is compared to local sensor bitmaps and the support parameter of nodes within $PotOut$ is increased appropriately upon a similarity occurrence. In this phase, upon a successful similarity test, we do not increase the support of motes within the current cluster (i.e., the cluster of C), since at the same time the potential outliers produced by C have been correspondingly forwarded to neighboring clusters in search of additional support. Any potential outlier that reaches the desired *minSup* support is excluded from the list of potential outliers that will be forwarded to the next clusterhead.

5.3 Boosting TACO Encodings

We note that the process described in Section 5.2 can accurately compute the support of a mote in the network (assuming a reliable communication protocol that resolves conflicts and lost messages). Thus, if the whole process was executed using the initial measurements (and not the LSH vectors) the resulting list of outliers would be exactly the same with the one that would be computed by the base station, after receiving all measurements and performing the calculations locally. The application of LSH however results in imprecision during pair-wise similarity tests. We now show how this imprecision can be bounded in a controllable manner in order to satisfy the needs of the monitoring application.

Assume that a clusterhead has received a pair of bitmaps X_i, X_j, each consisting of d bits. We split the initial bitmaps X_i, X_j in μ groups (X_{i_1}, X_{j_1}), (X_{i_2}, X_{j_2}), ..., (X_{i_μ}, X_{j_μ}), such that X_i is the concatenation of $X_{i_1}, \ldots, X_{i_\mu}$, and similarly for X_j. Each of X_{i_κ} and X_{j_κ} is a bitmap of n bits such that $d = \mu \cdot n$. For each group g_κ we obtain an estimation θ_κ of angle similarity using Equation 5 and, subsequently, an answer to the similarity test based on the pair of bitmaps in the group. We then provide as an answer to the similarity test, the answer provided by the majority of the μ similarity tests.[2]

Two questions that naturally arise are: (i) Does the aforementioned partitioning of the hash space help improve the accuracy of successful classification?; and (ii) Which value of μ should one use? Let us consider the probability of correctly classifying two similar vectors in TACO (the case of dissimilar vectors is symmetric). In our original (unpartitioned) framework, the probability of correctly determining the two vectors as similar is $P_{similar}(d)$, given by Equation 6. Thus, the failure probability of incorrectly classifying two similar vectors as dissimilar is $P_{wrong}(d) = 1 - P_{similar}(d)$.

By separating the initial bitmaps to μ groups, each containing $\frac{d}{\mu}$ bits, one can view the above classification as using μ independent Bernoulli trials, which each return 1 (similarity) with a success probability of $P_{similar}(\frac{d}{\mu})$, and 0 (dissimilarity), otherwise. Let X denote the random variable that computes the sum of these

[2]Ties are resolved by taking the median estimate of θ_ks.

μ trials. In order for our classification to incorrectly classify the two vectors as dissimilar, more than half of the μ similarity tests must fail. The average number of successes in these μ tests is $\overline{X} = \mu \times P_{similar}(\frac{d}{\mu})$. A direct application of the Chernoff bounds gives that more than half of the Bernoulli trials can fail with a probability $P_{wrong}(d,\mu)$ at most: $P_{wrong}(d,\mu) \leq e^{-2\mu(P_{similar}(\frac{d}{\mu})-\frac{1}{2})^2}$. Given that the number of bits d and, thus, the number of potential values for μ is small, we may compare $P_{wrong}(d,\mu)$ with $P_{wrong}(d)$ for a small set of μ values and determine whether it is more beneficial to use this boosting approach or not. We also need to make two important observations regarding the possible values of μ: (i) The number of possible μ values is further restricted by the fact that our above analysis holds for μ values that provide a (per group) success probability > 0.5 and (ii) Increasing the value of μ may provide worse results, as the number of used bits per group decreases. We explore this issue further in our experiments.

6. LOAD BALANCING AND COMPARISON PRUNING

6.1 Leveraging Additional Motes for Outlier Detection

In our initial framework, clusterhead nodes are required to perform data collection and reporting, as well as bitmap comparisons. As a result, clusterheads are overloaded with extra communication and processing costs, thus resulting in larger energy drain, when compared to other nodes. In order to avoid draining the energy of clusterhead nodes, the network structure will need to be frequently reorganized (by electing new clusterheads). While protocols such as HEED limit the number of messages required during the clusterhead election process, this election process still requires bandwidth. In order to limit the overhead of clusterhead nodes, we thus extend our framework, by incorporating the notion of *bucket nodes*.

Bucket nodes (or simply buckets) are motes within a cluster the presence of which aims at distributing communication and processing tasks and their associated costs. Besides selecting the clusterhead nodes, within each cluster the election process continues to elect additional B bucket nodes. This election process is easier to carry out by using the same algorithm (i.e., HEED) that we used for the clusterhead election.

After electing the bucket nodes within each cluster, our framework determines a mechanism that distributes the outlier detection duties amongst them. Our goal is to group similar bitmaps in the same bucket so that the comparisons that will take place within each bucket produce just a few local outliers. To achieve this, we introduce a second level of hashing. More precisely:

- Recall that the encoding consists of d bits. Let $W_h(X_i)$ denote the Hamming weight (that is, the number of set bits) of a bitmap X_i containing the encoded measurements of node S_i. Obviously $0 \leq W_h(X_i) \leq d$.

- Consider a partitioning of the hash key space to the elected buckets, such that each hash key is assigned to the $\lfloor \frac{W_h(X_i)}{\frac{d}{B}} \rfloor$-th bucket. Motes with similar bitmaps will have nearby Hamming weights, thus hashing to the same bucket with high probability.

- Please recall that encodings that can support each other in our framework have a Hamming distance lower or equal to Φ_{D_h}. In order to guarantee that a node's encoding can be used to witness any possible encoding within its cluster, this encoding needs to be sent to all buckets that cover the hash key range $\lfloor \frac{\max\{W_h(X_i)-\Phi_{D_h},0\}}{\frac{d}{B}} \rfloor$ to $\lfloor \frac{\min\{W_h(X_i)+\Phi_{D_h},d\}}{\frac{d}{B}} \rfloor$. Thus, the value of B determines the number of buckets to which an encoding must be sent. Larger values of B reduce the range of each bucket, but result in more encodings being transmitted to multiple buckets. In our framework, we select the value B (whenever at least B nodes exist in the cluster) by setting $\frac{d}{B} > \Phi_{D_h} \implies B < \frac{d}{\Phi_{D_h}}$. As we will shortly show, this guarantees that each encoding will need to be transmitted to at most one additional bucket, thus avoiding hashing the measurements of each node to multiple buckets.

- The transmission of an encoding to multiple bucket nodes ensures that it may be tested for similarity with any value that may potentially witness it. Therefore, the support that a node's measurements have reached is distributed in multiple buckets needing to be combined.

- Moreover, we must also make sure that the similarity test between two encodings is not performed more than once. Thus, we impose the following rules:(a) For encodings mapping to the same bucket node, the similarity test between them is performed only in that bucket node; and (b) For encodings mapping to different bucket nodes, their similarity test is performed only in the bucket node with the lowest id amongst these two. Given these two requirements, we can thus limit the number of bucket nodes to which we transmit an encoding to the range $\lfloor \frac{\max\{W_h(X_i)-\Phi_{D_h},0\}}{\frac{d}{B}} \rfloor$ to $\lfloor \frac{W_h(X_i)}{\frac{d}{B}} \rfloor$. The above range for $B < \frac{d}{\Phi_{D_h}}$ is guaranteed to contain at most 2 buckets.

Thus, each bucket reports the set of outliers that it has detected, along with their support, to the clusterhead. The clusterhead performs the following tests:

- Any encoding reported to the clusterhead by at least one, but not all bucket nodes to which it was transmitted, is guaranteed not to be an outlier, since it must have reached the required support at those bucket nodes that did not report the encoding.

- For the remaining encodings, the received support is added, and only those encodings that did not receive the required overall support are considered to be outliers.

6.2 Load Balancing Among Buckets

Despite the fact that the introduction of bucket nodes does alleviate clusterheads from comparison and message reception load, it does not guarantee by itself that the portion of load taken away from the clusterheads will be equally distributed between buckets. In particular, we expect that motes sampling ordinary values of measured attributes will produce similar bitmaps, thus directing these bitmaps to a limited subset of buckets, instead of equally utilizing the whole arrangement. In such a situation, an equi-width partitioning of the hash key space to the bucket nodes is obviously not a good strategy. On the other hand, if we wish to determine a more suitable hash key space allocation, we require information about the data distribution of the monitored attributes and, more precisely, about the distribution of the hamming weight of the bitmaps that original value vectors yield. Based on the above observations, we can devise a load balancing mechanism that can be used after the initial, equal-width partitioning in order to repartition the hash key space between bucket nodes. Our load balancing mechanism fosters simple equi-width histograms and consists of three phases:
a) *histogram calculation* per bucket, b) *histogram communication* between buckets and c) *hash key space reassignment*.

During the *histogram calculation* phase, each bucket locally constructs equi-width histograms by counting $W_h(X_i)$ frequencies belonging to bitmaps that were hashed to them. The range of histogram's domain value is restricted to the hash key space portion assigned to each bucket. Obviously, this phase takes place side by side with the normal operation of the motes. We note that this phase adds minimum computation overhead since it only involves increasing by one the corresponding histogram bucket counter for each received bitmap.

In the *histogram communication* phase, each bucket communicates to its clusterhead (a) its estimated frequency counts attached, and (b) the width parameter c that it used in its histogram calculation. From the previous partitioning of the hash key space, the clusterhead knows the hash key space of each bucket node. Thus, the transmission of the width c is enough to determine (a) the number of received bars/values, and (b) the range of each bar of the received histogram. Thus, the clusterhead can easily reconstruct the histograms that it received.

The final step involves the adjustment of the hash key space allocation that will provide the desired load balance based on the transmitted histograms. Based on the received histograms, the clusterhead determines a new space partitioning and broadcasts it to all nodes in its cluster. The aforementioned phases can be periodically (but not frequently) repeated to adjust the bounds allocated to each bucket, adapting the arrangement to changing data distributions.

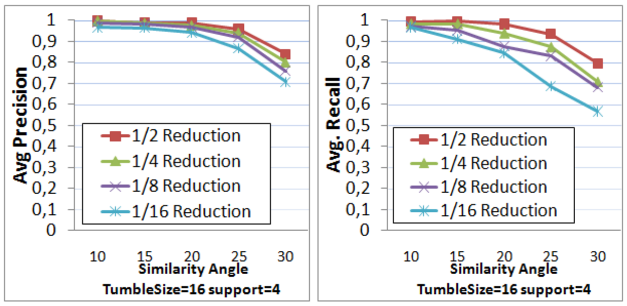
The mechanisms described in this section better balance the load among buckets and also refrain from performing unnecessary similarity checks between dissimilar pairs of bitmaps, which would otherwise have arrived at the clusterhead. This stems from the fact that hashing bitmaps based on their hamming weight ensures that dissimilar bitmaps are hashed to different buckets. We experimentally validate the ability of this second level hashing technique to prune the number of comparisons in Section 7.5.

7. EXPERIMENTS

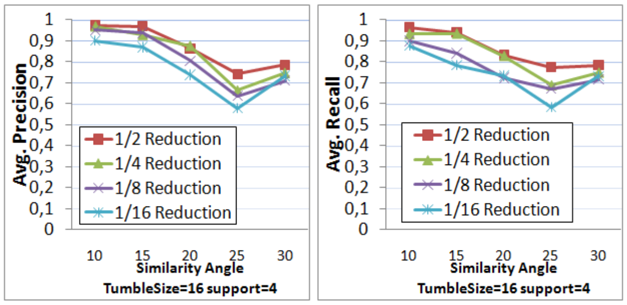
7.1 Experimental Setup

In order to evaluate the performance of our techniques we implemented our framework on top of the TOSSIM network simulator [20]. Since TOSSIM imposes restrictions on the network size and is rather slow in simulating experiments lasting for thousands of epochs, we further developed an additional lightweight simulator in Java and used it for our sensitivity analysis, where we vary the values of several parameters and assess the accuracy of our outlier detection scheme. The TOSSIM simulator was used in smaller-scale experiments, in order to evaluate the energy and bandwidth consumption of our techniques and of alternative methods for computing outliers. Through these experiments we examine the performance of all methods, while taking into account message loss and collisions, which in turn result in additional retransmissions and affect the energy consumption and the network lifetime.

In our experiments we utilized two real world data sets. The first, termed Intel Lab Data includes temperature and humidity measurements collected by 48 motes for a period of 633 and 487 epochs, respectively, in the Intel Research, Berkeley lab [9]. The second, termed Weather Data includes air temperature, relative humidity and solar irradiance measurements from the station in the University of Washington and for the year 2002 [7]. We used these measurements to generate readings for 100 motes for a period of 2000 epochs. In both data sets we increased the complexity of the temperature and humidity data by specifying for each mote a 6% probability that it will fail dirty at some point. We simulated failures using a known deficiency [9] of the MICA2 temperature sensor: each mote that fails-dirty increases its measurement (in our experiment this increase occurs at an average rate of about 1 degree per



(a) Intel.Temperature Precision, Recall vs Similarity Angle



(b) Intel.Humidity Precision,Recall vs Similarity Angle

Figure 5: Average Precision, Recall in Intel Data Set

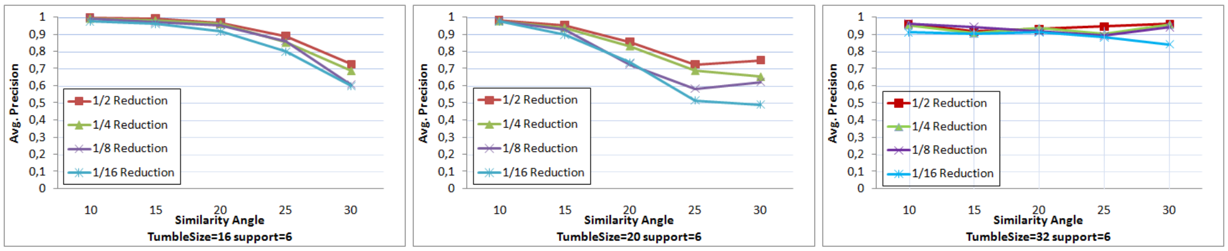
epoch), until it reaches a MAX_VAL parameter. This parameter was set to 100 degrees for the Intel lab data set and 200 degrees for the Weather data (due to the fact that the Weather data set contains higher average values). To prevent the measurements from lying on a straight line, we also impose a noise up to 15% at the values of a node that fails dirty. Additionally, each node with probability 0.4% at each epoch obtains a spurious measurement which is modeled as a random reading between 0 and MAX_VAL degrees. Finally, for solar irradiance measurements, we randomly injected values obtained at various time periods to the sequence of readings, in order to generate outliers.

We need to emphasize that increasing the complexity of the real data sets actually represents a worst-case scenario for our techniques. It is easy to understand that the amount of transmitted data during the intracluster communication phase is independent of the data sets' complexity, since it only depends on the specified parameter d that controls the dimensionality reduction. On the other hand, the amount of data exchanged during the intercluster phase of our framework depends on the number of generated outlier values. Thus, the added data set complexity only increases the transmitted data (and, thus, the energy consumption) of our framework. Despite this fact, we demonstrate that our techniques can still manage to drastically reduce the amount of transmitted data, in some cases even below what a simple aggregate query (i.e., MIN, MAX or SUM) would require under TAG [21].

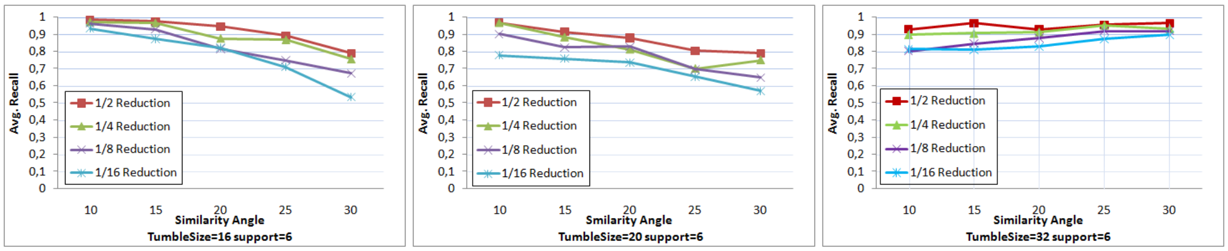
In the Intel Lab and Weather data sets we organized the sensor nodes in four and ten clusters, correspondingly. Please note that we selected a larger number of clusters for the Weather data set, due to the larger number of sensor nodes that appear in it.

7.2 Sensitivity Analysis

We first present a series of sensitivity analysis experiments using our Java simulator in order to explore a reasonably rich subset of the parameter space. To evaluate the accuracy of TACO in the available data sets we initially focus on the precision and recall metrics. In a nutshell, the precision specifies the percentage of reported outliers



(a) Weather Precision: Temperature, Humidity and Solar Irradiance vs Similarity Angle



(b) Weather Recall: Temperature, Humidity and Solar Irradiance vs Similarity Angle

Figure 6: Average Precision, Recall in Weather Data Set

that are true outliers, while the recall specifies the percentage of outliers that are reported by our framework. The set of true outliers was computed offline (i.e. assuming all data was locally available), based on the selected similarity metric and threshold, specified in each experiment. The goal of these experiments is to measure the accuracy of the TACO scheme and of the boosting process, and to assess their resilience to different compression ratios.

We used different tumble sizes ranging between 16 and 32 measurements and Φ_θ thresholds between 10 and 30 degrees. Moreover, we experimented with a reduction ratio up to 1/16 for each (W, Φ_θ) combination. In the Intel Lab data sets we found little fluctuations by changing the *minSup* parameter from 3-5 motes, so henceforth we consider a fixed *minSup*=4 (please recall that there are 48 motes in this data set). Due to a similar observation in the Weather data set, *minSup* is set to 6 motes. All the experiments were repeated 10 times. Figures 5 and 6 depict the accuracy of our methods presenting the average precision and recall for the used data sets, for different similarity angles and reduction ratios. To acquire these, we obtained precision and recall values per tumble and calculated the average precision, recall over all tumbles in the run. Finally, we proceeded by estimating averages over 10 repetitions, using a different random set of hash functions in each iteration.

As it can be easily observed, in most of the cases, motes producing outlying values can be successfully pinpointed by our framework with average precision and recall $> 80\%$, even when imposing a 1/8 or 1/16 reduction ratio, for similarity angles up to 20 degrees. The TACO scheme is much more accurate when asked to capture strict, sensitive definitions of outlying values, implied by a low Φ_θ value. This behavior is expected based on our formal analysis (see Figure 3 and Equation 2). We also note that the model may slightly swerve from its expected behavior depending on the number of near-to-threshold outliers (those falling in the areas FP, FN in Figure 3) that exist in the data set. That is, for instance, the case when switching from 25 to 30 degrees in the humidity data sets of Figures 5 and 6.

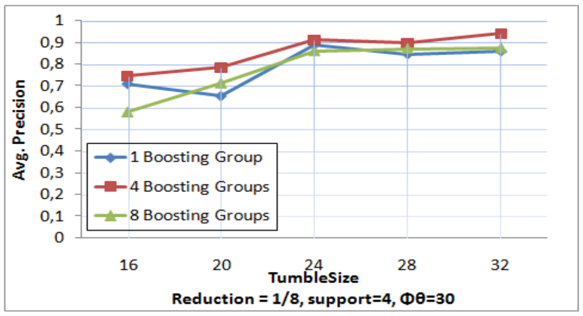
Obviously, an improvement in the final results may arise by increasing the length d of each bitmap (i.e., consider more moderate reduction ratios, Figure 4). Another way to improve performance is to utilize the boosting process discussed in Section 5.3. All previous experiments were ran using a single boosting group during the comparison procedure. Figure 7 depicts the improvement in the values of precision and recall for the Intel humidity data set as more groups are considered and for a variety of tumble sizes (the trends are similar for the other data sets, which are omitted due to space constraints, as well). Points in Figure 7 corresponding to the same tumble size W use bitmaps of the same length, so that the reduction ratio is 1/8, but differ in the number of groups utilized during the similarity estimation. It can easily be deduced that using 4 boosting groups is the optimal solution for all the cited tumble sizes, while both the 4 group and 8 group lines tend to ascend by increasing the W parameter. This comes as no surprise since the selection of higher W values results in larger (but still 1/8 reduced) bitmaps, which in turn equip the overall comparison model with more accurate submodels. Moreover, notice that using 8 groups may provide worse results (i.e., for W=16) since the number of bits per group is in that case small, thus resulting in submodels that are prone to produce low quality similarity estimations. For example using 8 groups for W=16, results in just 8 bits for each theta-estimator. Thus, for these data sets the optimal number of groups (μ) is 4.

Taking one step further we extracted 95% confidence intervals for each tumble across multiple data sets. Due to space limitations we omit the corresponding graphs, however, we note TACO exhibits little deviations (± 0.04) from its average behavior in a tumble in all of the data sets.

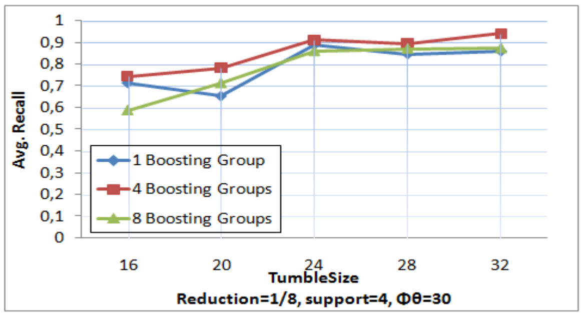
7.3 Performance Evaluation Using TOSSIM

Due to limitations in the simulation environment of TOSSIM, we restricted our experimental evaluation to the Intel Lab data set. We used the default TOSH_ DATA_ LENGTH value set to 29 bytes and applied 1/4, 1/8 and 1/16 reduction ratios to the original binary representation of tumbles containing W=16 values each.

We measured the performance of our TACO framework against two alternative approaches. The first approach, termed as *Non-TACO*, performed the whole intra and inter-cluster communication procedure using the initial value vectors of motes "as is". In the TACO and NonTACO approaches, motes producing outlying values were identified in-network, following precalculated TSP paths, and were subsequently sent to the base station by the last clusterhead in each path. In the third approach, termed as *SelectStar*, motes transmitted original value vectors to their clusterheads and,



(a) Intel.Humidity Precision vs Tumble size



(b) Intel.Humidity Recall vs Tumble size

Figure 7: Intel.Humidity Precision,Recall adjustment by boosting fixed 1/8 compressed bitmaps

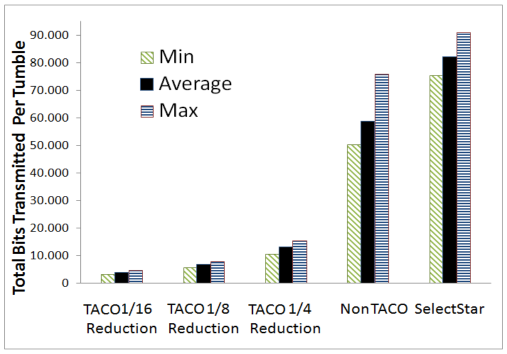


Figure 8: Total Bits Transmitted per approach

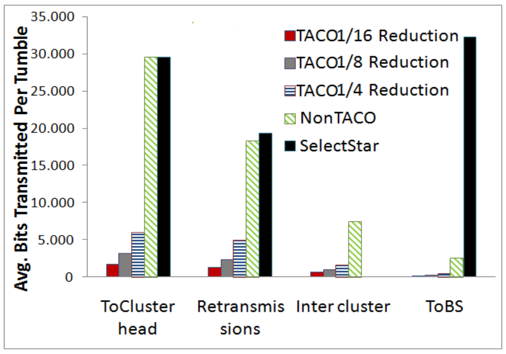


Figure 9: Transmitted bits categorization

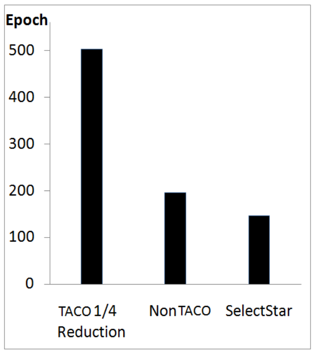


Figure 10: Average Lifetime

omitting the intercluster communication phase, clusterheads forwarded these values as well as their own vector to the base station.

Besides simply presenting results involving these three approaches (TACO, NonTACO and SelectStar), we also seek to analyze their bandwidth consumption during the different phases of our framework. This analysis yields some interesting comparisons. For example, the number of bits transmitted during the intracluster phase of NonTACO provides a *lower bound* for the bandwidth consumption that a simple continuous aggregate query (such as MAX or SUM query) requires under TAG for all epochs, as this quantity: (a) Simply corresponds to transmitting the data observations of each sensor to one-hop neighbors (i.e., the clusterheads), and (b) Does not contain bandwidth required for the transmission of data from the clusterheads to the base station. Thus, if TACO requires fewer transmitted bits than the intracluster phase of NonTACO, then it also requires less bandwidth than a continuous aggregate query.

In our setup for TACO, during the first tumble, the base station broadcasts a message encapsulating three values: (1) the tumble size parameter W; (2) the bitmap length d; and (3) a common seed, which enables the motes to produce the same $d{*}W$ matrix of uniformly distributed values composing d LSH vectors. The overhead of transmitting these values is included in the presented graphs.

Figure 8 depicts the average, maximum and minimum number of total bits transmitted in the network in a tumble for the TACO (with different reduction ratios), NonTACO and SelectStar approaches. Comparing, for instance, the performance of the middle case of 1/8 Reduction and the NonTACO executions, we observe that, in terms of total transmitted bits the reduction achieved by TACO is on the average 1/9 per tumble, thus exceeding the imposed 1/8 reduction ratio. The same observation holds for the other two reduction ratios. This comes as no surprise, since message collisions entailing retransmissions are more frequent with increased message sizes used in NonTACO, augmenting the total number of bits transmitted. Furthermore, comparing these results with the SelectStar approach exhibits the efficiency of the proposed inter-cluster communication phase for in-network outlier identification. The achieved reduction ratio of TACO 1/8 Reduction, when compared to the SelectStar approach is, on average 1/12, with a maximum value of 1/15. This validates the expected benefit derived by TACO.

Figure 9 presents a categorization of the average number of bits transmitted in a tumble. For each of the approaches, we categorize the transmitted bits as: (1) ToClusterhead: bits transmitted to clusterheads during the intra-cluster communication phase; (2) Intercluster: bits transmitted in the network during the inter-cluster communication phase (applicable only for TACO and NonTACO); (3) ToBasestation: bits transmitted from clusterheads towards the base station; (4) Retransmissions: additional bits resulting from message retransmission due to lossy communication channels or collisions. In Figure 9, please notice that the bits classified as Intercluster are always less than those in the ToClusterhead category. Moreover, the total bits of TACO (shown in Figure 8), are actually less than what NonTACO requires in its intracluster phase (Figure 9), even without including the corresponding bits involving retransmissions during this phase (73% of its total retransmission bits). Based on our earlier discussion, this implies that TACO under collisions and retransmissions is able to identify outlier readings at a fraction than what even a simple aggregate query would require.

As a final exhibition of the energy savings provided by our framework, we used PowerTOSSIM [25] to acquire power measurements yielded during simulation execution. In Figure 10 we used the previously extracted power measurements to plot the average network lifetime for motes initialized with 5000 mJ residual energy. Network lifetime is defined as the epoch on which the first mote in the network totally drains its energy. Overall, the TACO application

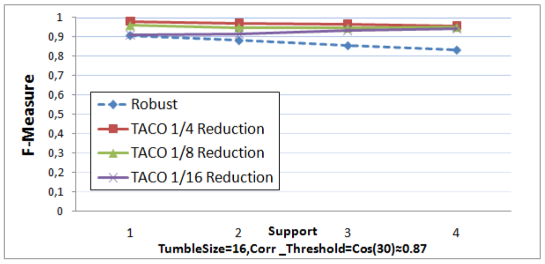


Figure 11: Intel.Temperature TACO vs Robust Accuracy varying $minSup$

reduces the power consumption up to a factor of 1/2.7 compared to the NonTACO approach. The difference between the selected reduction ratio (1/4) and the corresponding power consumption ratio (1/2.7) stems from the fact that motes need to periodically turn on/off their radio to check whether they are recipients of any transmission attempts. This fact mainly affects the TACO implementation since in the other two approaches, where more bits are delivered in the network, the amount of time that the radio remains turned on is indeed devoted to message reception. We leave the development of a more efficient transmission/reception schedule, tailored for our TACO scheme as future work.

7.4 TACO vs Hierarchical Outlier Detection Techniques

In the previous sections we experimentally validated the ability of our framework to tune the amount of transmitted data while simultaneously accurately predicting outliers. On the contrary, existing in-network outlier detection techniques, such as the algorithm of [9, 27] cannot tune the amount of transmitted information. Moreover, these algorithms lack the ability to provide guarantees since they both base their decisions on partial knowledge of recent measurements received by intermediate nodes in the hierarchy from their descendant nodes. In this subsection, we perform a comparison to the recently proposed algorithm of [9], which we will term as *Robust*. We use *Robust* as the most representative example to extract comparative results related to accuracy and bandwidth consumption since it uses an equivalent outlier definition and bases its decisions on common similarity measures. As in the previous subsection, we utilized the Intel Lab data set in our study, keeping the TACO framework configuration unchanged.

In order to achieve a fair comparison, the Robust algorithm was simulated using a tree network organization of three levels (including the base station) with a $CacheSize = 24$ measurements. Note that such a configuration is a good scenario for Robust since most of the motes that can witness each other often share common parent nodes. Thus, the loss of witnesses as data ascend the tree organization is reduced. Please refer to [9] for further details.

In the evaluation, we employed the correlation coefficient-*corr* (see Table 1) as a common similarity measure equivalent to the cosine similarity as mentioned in Section 4.2. We chose to demonstrate results regarding the temperature measurements in the data set. However, we note that the outcome was similar for the humidity data and proportional for different Φ_{corr} thresholds. Figure 11 depicts the accuracy of Robust compared to TACO with different reduction ratios varying the $minSup$ parameter. To acquire a holistic performance view of the approaches, we computed the *F-Measure* metric as *F-measure=2/(1/Precision+1/Recall)*. Notably, TACO behaves better even for the extreme case of 1/16 reduction, while Robust falls short up to 10%. To complete the picture, Fig-

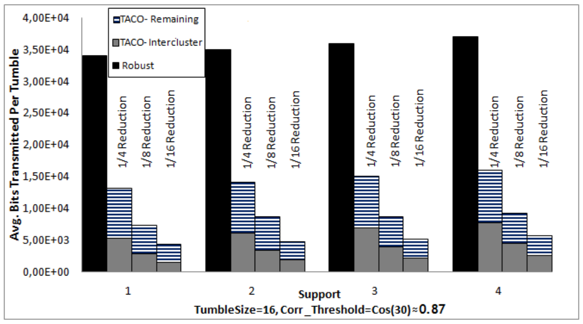


Figure 12: Intel.Temp. TACO vs Robust transmitted bits

		Φ_θ					
		10			20		
Cluster Size	Buckets	Cmps	Multihash Messages	Bitmaps Per Bucket	Cmps	Multihash Messages	Bitmaps Per Bucket
12	1	66.00	0	12	66	0	12
	2	38.08	0.90	6.45	40.92	1.36	6.68
	4	24.55	7.71	3.65	30.95	8.88	4.08
24	1	276.00	0	24	276	0	24
	2	158.06	1.62	12.81	171.80	2.76	13.38
	4	101.10	14.97	7.27	128.63	17.61	8.15
36	1	630	0	36	630	0	36
	2	363.64	2.66	19.33	394.97	4.30	20.15
	4	230.73	22.88	10.88	291.14	26.28	12.19
48	1	1128	0	48	1128	0	48
	2	640.10	3.14	25.57	710.95	5.85	26.93
	4	412.76	30.17	14.49	518.57	34.64	16.21

Table 3: The effect of bucket node introduction (W=16, d=128)

ure 12 shows the average bits transmitted by motes in the two different settings. Notice that the stacked bars in the TACO approach form the total number of transmitted bits which comprises the bits devoted to intercluster communication (*TACO-Intercluster*) and those termed as *TACO-remaining* for the remainder. The increment of the $minSup$ parameter in the graph correspondingly causes an increment in the *TACO-Intercluster* bits as more motes do not manage to find adequate support in their cluster and subsequently participate in the intercluster communication phase. TACO ensures less bandwidth consumption with a ratio varying from 1/2.6 for a reduction ratio of 1/4, and up to 1/7.8 for 1/16 reduction.

7.5 Bucket Node Exploitation

In order to better perceive the benefits derived from bucket node introduction, Table 3 summarizes the basic features ascribed to network clusters for different numbers B of bucket nodes. The table provides measurements regarding the average number of comparisons along with the average number of messages resulting from multi-hashed bitmaps. Moreover, it presents the average number of bitmaps received per bucket for different cluster sizes and Φ_θ thresholds. Focusing on the average number of comparisons per tumble (Cmps in the Table), this significantly decreases as new bucket nodes are introduced in the cluster. From this point of view, we have achieved our goal since, as mentioned in Section 6.1, not only bucket nodes do alleviate the clusterhead from comparison load, but also the hash key space distribution amongst them preserves the redundant comparisons.

Studying the number of multi-hash messages (MultihashMsgs in the Table) and the number of bitmaps received per bucket (BitmapsPerBucket) a trade-off seems to appear. The first column regards a message transmission cost mainly charged to the regular motes in a cluster, while the second involves load distribution between buckets. As new bucket nodes are adopted in the cluster, the MultihashMsgs increases with a simultaneous decrease in BitmapsPerBucket. In other words, the introduction of more bucket nodes causes a shift in the energy consumption from clusterhead and bucket nodes to regular cluster motes. Achieving appropriate balance, aids in maintaining uniform energy consumption in the whole cluster, which in turn leads to infrequent network reorganization.

8. CONCLUSIONS

In this paper we presented TACO, a framework for detecting outliers in wireless sensor networks. Our techniques exploit locality sensitive hashing as a means to compress individual sensor readings and use a novel second level hashing mechanism to achieve intra-cluster comparison pruning and load balancing. TACO is largely parameterizable, as it bases its operation on a small set of intuitive application defined parameters: (i) the length of the LSH bitmaps (d), which controls the level of desired reduction; (ii) the number of recent measurements that should be taken into account when performing the similarity test (W), which can be fine-tuned depending on the application's desire to put more or less emphasis to past values; (iii) the desired similarity threshold (Φ); and (iv) the required level of support for non-outliers. TACO is not restricted to a monolithic definition of an outlier but, instead, supports a number of intuitive similarity tests. Thus, the application can specialize and fine-tune the outlier detection process by choosing appropriate values for these parameters. We also presented novel extensions to the basic TACO scheme that boost the accuracy of computing outliers. Our framework processes outliers in-network, using a novel inter-cluster communication phase. Our experiments demonstrated that our framework can reliably identify outlier readings using a fraction of the bandwidth and energy that would otherwise be required, resulting in significantly prolonged network lifetime.

9. REFERENCES

[1] M. Bawa, H. Garcia-Molina, A. Gionis, and R. Motwani. Estimating Aggregates on a Peer-to-Peer Network. Technical report, Stanford, 2003.

[2] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD*, 2003.

[3] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-network outlier detection in wireless sensor networks. In *ICDCS*, 2006.

[4] D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring streams: a new class of data management applications. In *VLDB*, 2002.

[5] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, 2002.

[6] J. Chen, S. Kher, and A. Somani. Distributed Fault Detection of Wireless Sensor Networks. In *DIWANS*, 2006.

[7] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing Historical Information in Sensor Networks. In *ACM SIGMOD*, 2004.

[8] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Hierarchical In-Network Data Aggregation with Quality Guarantees. In *EDBT*, 2004.

[9] A. Deligiannakis, Y. Kotidis, V. Vassalos, V. Stoumpos, and A. Delis. Another Outlier Bites the Dust: Computing Meaningful Aggregates in Sensor Networks. In *ICDE*, 2009.

[10] A. Gionis, D. Gunopulos, and N. Koudas. Efficient and tunable similar set retrieval. In *SIGMOD*, 2001.

[11] M. Goemans and D. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6), 1995.

[12] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[13] Y. j. Wen, A. M. Agogino, and K.Goebel. Fuzzy Validation and Fusion for Wireless Sensor Networks. In *ASME*, 2004.

[14] S. Jeffery, G. Alonso, M. J. Franklin, W. Hong, and J. Widom. Declarative Support for Sensor Data Cleaning. In *Pervasive*, 2006.

[15] S. Jeffery, M. Garofalakis, and M. Franklin. Adaptive Cleaning for RFID Data Streams. In *VLDB*, 2006.

[16] B. Karp and H. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *MOBICOM*, 2000.

[17] D. Kempe, A. Dobra, and J. Gehrke. Gossip-Based Computation of Aggregate Information. In *FOCS*, 2003.

[18] N. Khoussainova, M. Balazinska, and D. Suciu. Towards Correcting Input Data Errors Probabilistically using Integrity Constraints. In *MobiDE*, 2006.

[19] Y. Kotidis. Snapshot Queries: Towards Data-Centric Sensor Networks. In *ICDE*, 2005.

[20] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *SenSys*, 2004.

[21] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: A Tiny Aggregation Service for ad hoc Sensor Networks. In *OSDI Conf.*, 2002.

[22] M. Qin and R. Zimmermann. VCA: An Energy-Efficient Voting-Based Clustering Algorithm for Sensor Networks. *J.UCS*, 13(1), 2007.

[23] D. Ravichandran, P. Pantel, and E. Hovy. Randomized algorithms and NLP: using locality sensitive hash function for high speed noun clustering. In *ACL*, 2005.

[24] B. Sheng, Q. Li, W. Mao, and W. Jin. Outlier detection in sensor networks. In *MobiHoc*, 2007.

[25] V. Shnayder, M. Hempstead, B. Chen, G. W. Allen, and M. Welsh. Simulating the Power Consumption of Large-Scale Sensor Network Applications. In *Sensys*, 2004.

[26] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware Routing in Mobile Ad Hoc Networks. In *MobiCom*, 1998.

[27] S. Subramaniam, T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Online Outlier Detection in Sensor Data Using Non-Parametric Models. In *VLDB*, 2006.

[28] X. Xiao, W. Peng, C. Hung, and W. Lee. Using SensorRanks for In-Network Detection of Faulty Readings in Wireless Sensor Networks. In *MobiDE*, 2007.

[29] G. Xue, Y. Jiang, Y. You, and M. Li. A topology-aware hierarchical structured overlay network based on locality sensitive hashing scheme. In *UPGRADE*, 2007.

[30] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3), 2002.

[31] O. Younis and S. Fahmy. Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach. In *INFOCOM*, 2004.

[32] D. Zeinalipour, P. Andreou, P. Chrysanthis, G. Samaras, and A. Pitsillides. The Micropulse Framework for Adaptive Waking Windows in Sensor Networks. In *MDM*, 2007.

[33] Y. Zhuang, L. Chen, S. Wang, and J. Lian. A Weighted Moving Average-based Approach for Cleaning Sensor Data. In *ICDCS*, 2007.