PAO: Power-Efficient Attribution 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

ABSTRACT

Sensor nodes constitute inexpensive, disposable devices that are often scattered in harsh environments of interest so as to collect and communicate desired measurements of monitored quantities. Due to the commodity hardware used in the construction of sensor nodes, the readings of sensors are frequently tainted with outliers. Given the presence of outliers, decision making in sensor networks becomes much harder. In this work, we introduce PAO, a framework that can reliably and efficiently detect outliers in wireless sensor networks. PAO significantly reduces the bandwidth consumption during the outlier detection procedure, while being able to operate over multiple window types. Moreover, our framework possesses the ability to operate in either an exact mode, or an approximate one that further reduces the communication cost, thus covering a wide variety of application requirements.

1. INTRODUCTION

Many monitoring applications rely on wireless sensory infrastructures in order to obtain measurements of the surrounding environment. Examples include habitat monitoring applications that collect meteorological data (like temperature, pressure, humidity etc), military surveillance applications that track movement of personnel or detect potentially hazardous chemicals, as well as vehicle tracking and traffic surveillance applications. Despite their diversity and differences, all such applications share the need to collect measurements that accurately reflect the conditions of the physical world being monitored. However, sensory infrastructures, in order to provide an economically viable solution, typically rely on inexpensive hardware used for the construction of the nodes. As a result, sensor nodes often generate imprecise individual readings due to interference or failures [10]. In several application scenarios, sensor nodes are often thrown in hazardous environments and need to operate in an unattended manner for long periods of time. Such nodes, may be exposed to severe conditions that adversely affect their sensing elements, thus yielding readings of low quality. As an example, the humidity sensors on the popular MICA mote is very sensitive to rain drops [5]. A question that naturally arises is whether (and how) one can build applications that take mission-critical decisions, given the lack of trust on the baseline measurements provided as input by the sensory infrastructure.

In order to address this question, a flurry of recent work has targeted the problem of outlier detection in sensor networks [3, 5, 12, 17, 18]. Although the detection of outliers is an old research problem, the particular challenges introduced when considering ad-hoc sensor networks render conventional outlier detection algorithms [2] unsuitable for this new setting. In particular, sensor nodes often have limited processing and storage capabilities. As a result, sensory data that is continuously collected by the nodes may be maintained in memory for only a limited amount of time. Moreover, since data is collected continuously (typically in predefined intervals, called epochs), outlier detection needs on-line mechanisms that will work in this restrictive, streaming setting. However, the main constraint in sensor network applications is often the limited energy capacity of each sensor node. In many applications, sensor nodes are powered by batteries that cannot be easily replaced if depleted, given that nodes are often thrown in remote sites. Therefore, in order to ensure the longevity of the sensor network, we need to devise techniques that can detect outliers in an energy-efficient manner, thus reducing the energy drain of the nodes. It is well understood that radio communication is by far the the biggest culprit in energy drain [13]. This means that a central collection of all sensory data (and subsequently, computation of outliers using existing centralized techniques) is not feasible 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.

In this paper we introduce PAO (PAO stands for Power-efficient Attribution of Outliers), an outlier detection framework tailored for the particular constraints faced in typical sensor networks applications. Our framework follows the in-network paradigm, meaning that computation of outliers is performed inside the network, avoiding in this way the communication of raw sensor measurements to the base station. Furthermore, PAO possesses the ability to perform over multiple types of windows of observations collected by motes. Similar to previous techniques [5, 8], our framework takes into account both temporal and spatial correlations in order to characterize the readings of a sensor node. Temporal correlations are captured by considering the latest measurements of a sensor node and by computing localized regression models out of them. These compact models are of fixed size and are used to replace the original data values, reducing in this way the size of data that is transmitted in the network. In PAO, we adopt a clustered network organization [14, 19], where nodes communicate their regression models to a clusterhead, which computes the similarity amongst the latest values of any pair of sensors within its cluster. Based on the performed similarity tests and a desired minimum support specified by the posed query, each clusterhead generates a list of *potential* outliers within its cluster. This list is then communicated, in a second (inter-cluster) communication phase of PAO, among the clus-

DMSN '10, September 13, 2010, Singapore

terheads, in order to allow potential outliers to gain support from measurements of nodes that lie in other clusters. The whole process is sketched in Figure 2.

In order to alleviate clusterheads from comparing models from all nodes within their cluster, we introduce PAO+, an extended version of the original framework where additional nodes within each cluster are utilized for that purpose. This extended scheme is made possible by introducing a simple, yet effective hashing scheme over the regression parameters computed by the sensor nodes. The benefits of this extended scheme are twofold. Not only do we spread the load into multiple nodes, but also (as will be explained) manage to avoid many comparisons between regression models of motes that are provably not similar and, thus, cannot support each other. The PAO+ scheme also offers a load balancing mechanism that periodically adopts the hashing functions to the characteristics of the collected data, resulting in more effective comparison pruning.
The contributions of this paper can be summarized as follows:

1. We introduce PAO, an in-network outlier detection framework that permits computation of outliers in a clustered sensor network. PAO is capable of performing over different window types, it takes into account both temporal and spatial correlations among the measurements of the nodes and utilizes simple regression models in order to reduce communication overhead. We provide techniques for suppressing update messages by the motes, in continuous queries, resulting in fewer transmitted bits.
2. We describe PAO+, which extends PAO with a novel load balancing and comparison pruning mechanism. The proposed extensions alleviate clusterheads from excessive processing and communication load and result in a more uniform, intra-cluster power consumption and prolonged network unhindered operation.
3. We present a detailed experimental analysis of our techniques using real data sets. Our analysis demonstrates that our techniques manage to detect outliers using only a fraction of the bandwidth that a centralized approach would require.

This paper proceeds as follows. In Section 2 we comment on related work. Section 3 presents preliminary concepts. Our PAO framework is introduced in Section 4, while its extensions are discussed in Section 5. Section 6 presents our experimental evaluation, while Section 7 includes concluding remarks.

2. RELATED WORK

Recently, substantial effort has been devoted to the development of efficient outlier detection techniques that manage to pinpoint motes producing low quality readings or observe interesting undergoing phenomena so that proper actions can be taken [21]. [6, 10] introduce data cleaning mechanisms over sensor data streams after their central collection at the query source. Nevertheless, the central collection of data is not feasible nor desired as it has a cumulative effect on the amount of communicated data, which in turn depletes the residual energy of the motes.

Localized voting protocols [3, 18] have been proposed to determine faulty motes in completely ad-hoc network formations. However, such voting schemes are prone to errors when motes generating imprecise measurements are not able to communicate with each other due to physical obstacles or other unpredictable disturbances in their surrounding [5]. A fused weighted average scheme is proposed in [9] where a fuzzy mechanism is utilized to infer the correlation among sensor readings. In other related work, [22] makes use of a weighted moving average to clean imprecise samples while a histogram-based outlier attribution method is presented in [16].

The authors of [17] foster kernel functions to estimate the data distribution of motes and subsequently detect distance-based outliers leveraging this information. The work of [5] manages to provide outlier reports on par with the execution of aggregate and group-by queries posed to an aggregation tree [1, 13, 20]. It thus excludes extraordinary measurements avoiding the distortion of the final aggregate result and simultaneously allows users to acquire important information of motes exhibiting abnormal behavior. The recent work of TACO [8], manages to efficiently determine outliers by providing a mechanism based on Locality Sensitive Hashing [7], which trades bandwidth consumption for accuracy during the outlier detection procedure in a straightforward way. However, PAO is applicable to multiple types of window queries. Message suppression schemes in sensor networks for continuous aggregate queries have been studied in [4, 15]. Our work differs in that we do not suppress raw measurements but updates to model parameters instead.

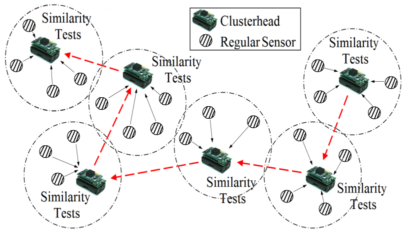


Figure 1: Main Stages of PAO. Step 1: Intra-cluster communication between regular motes and clusterheads (solid black arrows). Step 2: Similarity tests are performed by clusterheads. Step 3: An approximate TSP problem is solved, potential outliers are exchanged (dashed red arrows). The final outlier list is transmitted to the base station (not shown).

3. PRELIMINARIES

3.1 Network Model

We adopt an underlying network structure where motes are organized into clusters (shown as dotted circles in Figure 2) using any existing network clustering algorithm [14, 19]. Queries are propagated by the base station to the clusterheads, which, in turn, disseminate these queries to sensors within their cluster.

3.2 Analyzing Trends in Mote Time Series

A time series constitutes a sequence of observations $Y_{t_0}, Y_{t_1}, \ldots, Y_{t_{w-1}}$, where $t_0 < t_1 < \cdots < t_{w-1}$, regarding a studied attribute of interest Y_t, in w different time instances. In our sensor network setting, a posed outlier detection query dictates the epoch parameter e, which is the time interval between consecutive samples. As a result, after obtaining w quantities every mote S_u has formed a series $Y_t^{S_u}$ with $t = (0, e, \ldots, (w-1) \cdot e)$ as the vector of the corresponding timestamps.

Time series analysis techniques aim at capturing the implied behavioral pattern in the observed data. A fundamental component describing the existing patterns is the *trend* of the series, which is able to describe the rate of change on the values of an attribute. This in turn provides a compact picture of the presence of interesting phenomena imprinted on previously acquired samples. A simple yet popular way to depict the trend of time series data is through linear models in which:

$$\hat{Y}_t = \hat{a} \cdot t + \hat{b} \qquad (1)$$

According to Equation 1, the value of a studied attribute given the time vector is expected to be encompassed by a line with parameters $\hat{a}$ and $\hat{b}$ taking values:

$$\begin{cases} \hat{a} = \frac{12(\sum_{i=0}^{w-1} t_i \cdot Y_{t_i} - \frac{e \cdot w \cdot (w-2)}{2(w-1)} \sum_{i=0}^{w-1} Y_{t_i})}{e^2 \cdot w \cdot (w^2+2)} \\ \\ \hat{b} = \overline{Y}_t - \hat{a} \cdot \frac{e \cdot w}{2} \end{cases} \qquad (2)$$

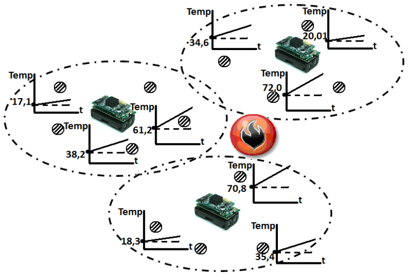


Figure 2: Trends and intercept points in mote time series depending on the spatial proximity to the source of the fire burst.

$\overline{Y}_t$ refers to the mean of Y_{t_i}s. Parameter $\hat{a}$ expresses the *slope* of the linear model, while $\hat{b}$ represents the *intercept* point between the line and Y_t axis. Hence, $\arctan(\hat{a})$ computes the actual angle $\measuredangle Y_t$ that the linear model's slope forms in respect with the time axis and $-\frac{\pi}{2} < \measuredangle Y_t < \frac{\pi}{2}$.

As an example, consider a sensor network in a forest designed to sample attributes such as temperature, humidity etc. Should a fire burst arise (Figure 2) nearby motes will collect increasing temperature values. The absolute sampled values actually depend on the radius around the source of the event that a mote is placed but its rate and, thus, the trend of the corresponding time series will be similar. In other words, upon utilizing a linear representation so as to model the trend on motes data, parameter $\hat{b}$ regards the proximity of a mote to the source of an event, contrary to $\hat{a}$ which shows the actual rate in change of values. The same observation holds for other physical attributes such as humidity (i.e., flood occurrence where motes obtain increased humidity values sensed in the air), sound vibrations, radiance measurements etc.

Nevertheless, in practice samples within a specific time window may exhibit extraordinary deviation where no clear trend seems to appear. Situations like these should be handled differently due to the lack of a certain behavioral pattern. The question is how could someone check whether such a pattern does exist. To achieve that we reside to the *correlation of determination*, which shows the amount of variance in Y_t explained by the model:

$$R^2 = \frac{\sum_{i=0}^{w-1}(\hat{Y}_{t_i} - \overline{Y}_t)^2}{\sum_{i=0}^{w-1}(Y_{t_i} - \overline{Y}_t)^2}, 0 \leq R^2 \leq 1 \qquad (3)$$

High values of R^2 validate that a trend exists and is well modeled by Equation 1. On the contrary, low values indicate the absence of this kind of motive.

3.3 Outlier Definition

Based on our previous discussion we formalize our definition of outlying values. We assume that the posed outlier detection query has specified a couple of parameters p, $\measuredangle_{thres}$, whose meaning will be introduced shortly. Given the time series $Y_t^{S_u}, Y_t^{S_v}$ of motes S_u, S_v we initially utilize Equation 3 to check whether behavioral patterns that can be described by linear models occur based on threshold p. That is, we simply check whether $R^2 \geq p$ for $Y_t^{S_u}$ and $Y_t^{S_v}$, respectively. Please note that each test can be performed independently by the corresponding mote. If this is true, then, as already mentioned, we only need to compare the trends based on the value of $\hat{a}$s and more precisely on the equivalent angles $\measuredangle Y_t^{S_u}, \measuredangle Y_t^{S_v}$.

Given a similarity threshold $\measuredangle_{thres}$ specified by the posed query we consider $Y_t^{S_u}, Y_t^{S_v}$ as similar if:

$$|\measuredangle Y_t^{S_u} - \measuredangle Y_t^{S_v}| \leq \measuredangle_{thres} \qquad (4)$$

Acquired samples that do not pass the R^2 linearity test do not exhibit any profound motif and should be compared separately. Such cases can be handled using the *cosine coefficient* so as to compute the angle similarity [8] between vectors Vec_{S_u}, Vec_{S_v}. Value vector $Vec_{S_u} \in \mathbb{R}^w$ contains the measurements of S_u during the latest w samples and the angle similarity in this case is calculated by $\measuredangle(Vec_{S_u}, Vec_{S_v}) = \arccos \frac{Vec_{S_u} \cdot Vec_{S_v}}{||Vec_{S_u}|| \cdot ||Vec_{S_v}||}$.

As in [5, 8], we require our techniques to be resilient to environments where spurious readings originate from multiple node time series, due to a multitude of different and unpredictable factors. Thus, for a mote not to be classified as an outlier it should be found similar with at least *minSup* other motes. The value of *minSup* can be expressed either as an absolute value or as a percentage of motes.

4. OUR PAO ALGORITHM

We now present our PAO algorithm in detail. We assume that an outlier attribution query of the form:

```
SELECT $c.S_u$
FROM Clusterheads $c$
WHERE $c.Support_{S_u} < minSup$
USING [
SAMPLING PARAMETERS ($Interval = e$,$Size = w$),
TESTS(Linearity $p$, Similarity $\measuredangle_{thres}$),
CHECKS ON <set of specifications on similarity tests>),
WINDOW TYPE={ Disjoint | Sliding } with $\epsilon$]
```

has been posed to the sensor network. The parameters of the query have been presented in the previous sections. Regarding the set of specifications noted in CHECKS ON line of the USING compartment, we note that it refers to motes that may find support outside their cluster based on a set of static specifications. For instance, users may allow motes within a certain radius to be able to witness each other, irrespectively of whether they have been assigned to the same cluster, as they are expected to be able to sense similar conditions (i.e, the fire burst in the example of Figure 2). The last line of the query refers to the window type (disjoint or sliding). In a nutshell, using disjoint windows the query evaluation utilizes a set of w new samples (not used in previous query evaluations - this is often referred to as a tumble), while sliding windows always utilize the latest w observations (thus at each step taking into account $w-1$ observations that were also used in the previous evaluation, but then also adding the latest observation by the mote). Parameter ϵ that accompanies the window type involves a message suppression choice provided by PAO so as to support the potential for approximate detection of outliers with further reduced communication costs, as it will be explained at the end of the current section.

PAO at Individual Motes. After it receives a corresponding query, every mote S_u in the network assembles a time series $Y_t^{S_u}$. Initially, S_u computes $\hat{a}, \hat{b}$ using Equation 2, the correlation of determination R^2 using Equation 3 and performs the linear trend existence test by checking whether $R^2 \geq p$. Recall that p expresses a tolerance on the deviation of the collected measurements. In practice, an amount of this deviation is due to systematic calibration errors of the inexpensive hardware used in the construction of sensor nodes. As a result, knowing the specifications of the available mote hardware infrastructure, users can appropriately set the desired value for p. Subsequently, depending on the result of the latter test, S_u calculates $\measuredangle Y_t^{S_u} = arccos(\hat{a})$ which is then transmitted to the clusterhead. $R^2 < p$ results in communicating the analytical form Vec_{S_u} of samples to the clusterhead.

Intra-cluster Processing. Clusterheads receive data from motes in their cluster and organize this information in a tabular format with columns S_u, $\measuredangle Y_t^{S_u}$ or Vec_{S_u} for motes that did not pass the linearity test, and $Support_{S_u}$.

Data collection is horizontally fragmented between the clusterheads and further separated in motes with captured behavioral patterns and those which do not adapt to the model. The $Support_{S_u}$ column is set to 0 at the beginning of this phase. Subsequently, each clusterhead performs similarity tests as in Equation 4 on the first category of motes, while applying $\measuredangle(Vec_{S_u}, Vec_{S_v})$-based tests for the second. Each successful test increases the support of the participating motes by 1. At the end of the procedure, each clusterhead forms a list of tuples $\langle S_u, \measuredangle Y_t^{S_u}, Support_{S_u}\rangle$ and $\langle S_u, Vec_{S_u}, Support_{S_u}\rangle$ for sensors that did not manage to obtain enough witnesses to reach *minSup*.

Inter-cluster Processing. As already noted, lists of motes with $Support_{S_u} < minSup$ are not final outliers since the query may have allowed motes in different clusters to be tested for similarity. Motes in the lists extracted by cluster C_i that are not subjected to such kind of specifications can be directly reported to the query source. Otherwise, triplets are placed in a list $PotOut_{C_i}$ of potential (i.e., not yet determined) outliers. Given the current cluster as the starting node, query-specified clusterheads as intermediate sites and the base station for the destination, the intercluster communication problem is modeled as a TSP according to which $PotOut_{C_i}$s are exchanged between clusterheads participating in the path. The TSP problem can be solved by the base station after clusterhead election. Every $S_u \in PotOut_{C_i}$ that manages to reach *minSup* is excluded from the list that will be forwarded to the next site.

Approximate Processing over Multiple Window Types. So far, the procedure presented in PAO reduces the communication burden only for disjoint time windows. In other words, motes collect w quantities, form corresponding time series, and intra- as well as inter-clustering processes are then triggered. Provided that S_u succeeds in its linearity test, only $\measuredangle Y_t^{S_u}$s (instead of the entire series) are transmitted. Subsequently, these steps are repeated every w new measurements. On sliding window queries, new results are to be provided based on the $w-1$ previous observations and the latest w-th measurement, obtained every e time units. In this case, letting motes transmit $\measuredangle Y_t^{S_u}$ does not provide any savings in communication costs as it would be sufficient to merely send the newest w-th measurement instead.

To efficiently handle sliding windows PAO fosters a message suppression strategy to maintain low communication burden, while providing approximate answers of satisfactory quality. In particular, consider a clusterhead which has received $\measuredangle Y_t^{S_u}$ from S_u and assume a parameter ϵ encapsulated in the basestation's inquiry. In the upcoming window, we allow motes to suppress their own messages when $\measuredangle Y_{t_{new}}^{S_u} \in [\measuredangle Y_{t_{previous}}^{S_u} - \epsilon, \measuredangle Y_{t_{previous}}^{S_u} + \epsilon]$, where $\measuredangle Y_{t_{previous}}^{S_u}$ refers to the last value that the mote has transmitted to its clusterhead and $\measuredangle Y_{t_{new}}^{S_u}$ refers to the latest computed (but not necessarily transmitted) $\measuredangle Y_t^{S_u}$ value.

At clusterheads, similarity tests are performed in the same way as before. Nevertheless, the suppression of messages introduces approximate characteristics to PAO. It can easily be observed that for pairs of motes which did not suppress their messages the corresponding test between them will provide exact result. We now outline the cases of accurate similarity estimation despite message suppression:

- For pairs of motes that both suppress their messages clusterheads rely on $\measuredangle Y_{t_{previous}}^{S_u}, \measuredangle Y_{t_{previous}}^{S_v}$ to obtain an answer regarding their similarity. Without loss of generality, we assume that $\measuredangle Y_{t_{previous}}^{S_u} < \measuredangle Y_{t_{previous}}^{S_v}$. When $\measuredangle Y_{t_{previous}}^{S_u} + \epsilon + \measuredangle_{thres} < \measuredangle Y_{t_{previous}}^{S_v} - \epsilon$ the test is always accurate.

- Provided that S_v does not suppress its message while S_u does, clusterheads take into account $\measuredangle Y_{t_{previous}}^{S_u}, \measuredangle Y_{t_{new}}^{S_v}$. Assuming $\measuredangle Y_{t_{previous}}^{S_u} < \measuredangle Y_{t_{new}}^{S_v}$ (the other case is symmetric), a correct answer is ensured when $\measuredangle Y_{t_{previous}}^{S_u} + \epsilon + \measuredangle_{thres} < \measuredangle Y_{t_{new}}^{S_v}$.

Otherwise, the result of the test might be either faulty or correct, depending on the actual changes in $\measuredangle Y_{t_{new}}^{S_u}, \measuredangle Y_{t_{new}}^{S_v}$. Obviously, setting $\epsilon = 0$ is equivalent to requiring exact results. Moreover, notice that the above strategy manages to save communication costs irrespectively of the window type. Eventually, we note that ϵ can be dynamically adjusted by motes. Due to space limitations, we omit the corresponding discussion, but we refer interested readers to [4] for further details.

5. FROM PAO TO PAO+

During the intra- and inter-cluster communication phases of our PAO algorithm clusterheads are assigned the load of angle/vector reception as well as the processing burden of similarity test determination. This means that they consume extra power resources during these procedures compared to regular nodes in the clusters. Remaining energy is a primary criterion in any clustering protocol for a mote to be maintained as clusterhead. Thus, the network will need to frequently pause its operation and be led to a reorganization process so as to elect new clusterheads (which also yields an amount of communication cost for sensors). Bearing these, in PAO+ we introduce a hashing mechanism that spreads the intra-cluster communication and comparison load. Moreover, recall our similarity test $|\measuredangle Y_t^{S_u} - \measuredangle Y_t^{S_v}| \leq \measuredangle_{thres}$. A different reading of the inequality says that sensors with angle differences above $\measuredangle_{thres}$ should not be compared for similarity since we know in hand that the test cannot be successful. Nonetheless, having arrived at a clusterhead comparisons will inevitably be performed even for motes with high angle differences. PAO+ hashing mechanism also manages to prune unnecessary comparisons of motes exhibiting highly dissimilar behavioral patterns.

Load Distribution and Comparison Pruning in PAO+. Apart from electing the clusterhead we choose additional B nodes in the formed cluster as the hashing *Buckets*. To define the hashing procedure we need to clarify: i) the hash key, ii) the hash key space, iii) the hash function application. We proceed by presenting the aforementioned parameters. The hash key is set to be the angle $\measuredangle Y_t^{S_u}$ of mote S_u which means that the hash key space is determined by the fact that $-\frac{\pi}{2} < \measuredangle Y_t < \frac{\pi}{2}$. The previous range is equally distributed between the available buckets such that a bucket B_i holds a range between $[-\frac{\pi}{2}+i\cdot\frac{\pi}{B}, -\frac{\pi}{2}+(i+1)\frac{\pi}{B})$. Next, the hash function that is applied by motes in order to decide the receiver bucket is: $H(\measuredangle Y_t^{S_u}) = \lfloor \frac{\measuredangle Y_t^{S_u}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor = B_i$. Thus, in the intracluster processing, instead of letting all regular nodes transmit their data towards the clusterhead we impose that they should be sent to the $\lfloor \frac{\measuredangle Y_t^{S_u}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor$-th bucket.

Obviously, the process groups motes with similar trends in buckets while highly dissimilar motes hash in distant buckets in terms of the hash key space assignment. However, at the edges of the buckets similarity may still exist. More precisely, to guarantee that a node can be witnessed by any similar within the cluster, $\measuredangle Y_t^{S_u}$ needs to be sent to the bucket nodes that cover the range $\lfloor \frac{\max\{\measuredangle Y_t^{S_u} - \measuredangle_{thres}, 0\}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor$ to $\lfloor \frac{\min\{\measuredangle Y_t^{S_u} - \measuredangle_{thres}, \pi\}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor$. Utilizing more buckets reduces the range of each, but results in more $\measuredangle Y_t^{S_u}$s being transmitted to multiple buckets. PAO+ selects the value B (whenever at least B nodes exist in the cluster) by setting $B < \frac{\pi}{\measuredangle_{thres}}$ which limits the number of bucket nodes to which motes transmit their data to the range $\lfloor \frac{\max\{\measuredangle Y_t^{S_u} - \measuredangle_{thres}, 0\}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor$ to $\lfloor \frac{\measuredangle Y_t^{S_u}}{\frac{\pi}{B}} + \lceil\frac{B}{2}\rceil\rfloor$. The latter range is guaranteed to contain at most 2 buckets.

In order to make sure that the similarity test is not performed more than once we impose the following rules: (a) For $\measuredangle Y_t^{S_u}$s mapping to the same bucket, the similarity test between them is performed only in that bucket node; and (b) For $\measuredangle Y_t^{S_u}$s mapping to different bucket nodes, their similarity test is performed only in the bucket node with the lowest B_i.

Each bucket reports the set of outliers that it detected, along with their support, to the clusterhead. Any mote reported by at least one, but not all buckets to which it was transmitted, is guaranteed not to be an outlier, as it has reached $minSup$ at some bucket. Even when a mote is reported by all the buckets it was hashed, the support that its measurements have gained is distributed between buckets needing to be summed up. Hence, the received support is added, and only those $\measuredangle Y_t^{S_u}$s that did not receive the required overall support, from the buckets they were hashed to, are considered outliers.

We note that the whole process does not change the organization of the data as presented in Section 4, as it simply introduces an extra fragmentation step on the data. Eventually, sensor nodes that do not manage to pass the linearity test are assigned to a separate bucket to be compared with each other, omitting the hashing mechanism.

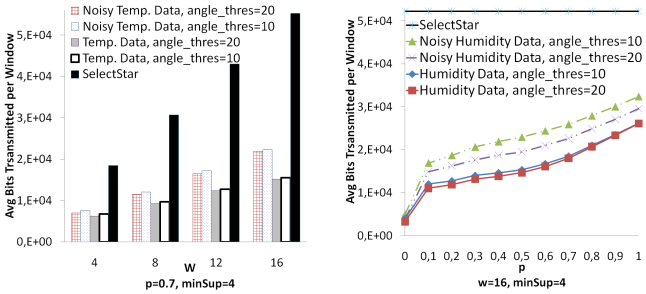
Load Balance in PAO+. The hashing mechanism we discussed distributes the processing and communication load during the intracluster phase of our algorithm. However, it does not guarantee that load apportionment will be equal between buckets. A naive way to confront occasions of high unbalanced load is to let buckets locally redetermine the hash key space for themselves and simply route any hashed information outside the new range to other left/right neighboring buckets. However, in this case our primary concern regarding bandwidth preservation is violated which at last deteriorates the power consumption for those buckets.

To balance the load amongst buckets and simultaneously achieve an efficient way to do so, PAO+ takes into consideration the monitored trends' distribution. More precisely, PAO+'s load balancing mechanism acts after the initial hash key space assignment and involves the construction of simple equi-width histograms. As buckets receive data from other motes in the cluster they maintain frequency counts of $\measuredangle Y_t^{S_u}$s. Subsequently, each bucket communicates to its clusterhead the estimated frequency counts along with the width parameter used in their construction. Every clusterhead is aware of the current hash key space assignment since it took part in the previous partitioning and can easily reconstruct the histograms.

Finally, based on these compact representation of the monitored patterns distribution, a new key space allocation is determined and broadcasted to all nodes in the cluster. The whole procedure can be periodically repeated i.e every a number of $w \cdot e$ time intervals to allow adaptations to changing data distributions.

6. EVALUATION RESULTS

Experimental Setup. In order to perform a comprehensive study of our algorithms varying different parameters, we developed a custom simulator. We randomly located sensors in a rectangular area and set the packet size to 16 bytes. We tested our PAO and PAO+ algorithms using a real world data set termed Intel Lab Data. The data set consists of temperature and humidity measurements collected by 48 motes for a period of 633 and 487 epochs, respectively, in the Intel Research, Berkeley lab [5]. To test our methods in harsh conditions, apart from using the original data sets, we produced extra versions (termed as "Noisy" in our experiments) where we increased the complexity of the measurements by specifying for each mote a 6% probability that it will fail dirty at some epoch. Failures were simulated using a known deficiency [5] of the MICA2 temperature sensor according to which a mote that fails-dirty increases its samples until it reaches a maximum value. We set that increment to 1 degree per epoch with a maximum value of 100 degrees. 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 random, spurious reading between 0 and 100 degrees. We organized our network in four clusters and utilized a $minSup$ value of 4 (i.e 1/3 of the total motes in a cluster). Eventually, in each experiment we used $\measuredangle_{thres}$ of 10 and 20 degrees which account for rigid and more relaxed cases of outlier definitions.



(a) Avg Bits Transmitted vs w (b) Avg Bits Transmitted vs p

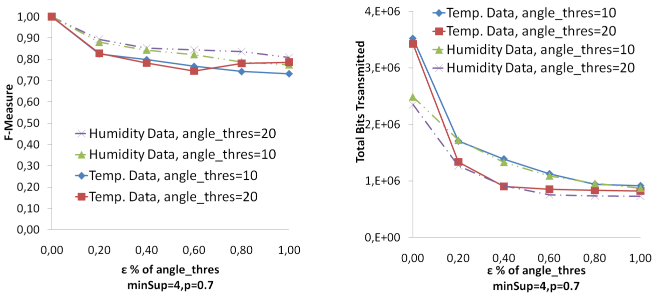
Figure 3: Avg Bits Transmitted per Window varying w and p

Bandwidth Consumption in PAO. We first present a set of experiments regarding the regular operation of our framework using disjoint time windows. We compare the bandwidth consumption of PAO against a centralized method termed "SelectStar" that collects all data in the query source and performs the outlier detection process there, instead of using PAOs in-network paradigm.

Figure 3(a) shows the reduction in bandwidth consumption provided by PAO for the temperature data, when varying the window size w, for a $p = 0.7$ threshold. PAO manages to reduce the average amount of transmitted data per window up to a factor of 1/3.8 for the original and 1/2.6 for the noisy data compared to the SelectStar approach. Additionally, Figure 3(b) depicts the average communication preservation depending on p's strictness for the humidity data using $w = 16$. Please note that setting the window size to the maximum of the previously (Fig. 3(a)) cited windows constitutes a worst case scenario for PAO, as the larger the window the fewer the motes that manage to adapt to the linear model. We can observe that the gains in the average amount of transmitted bits range between 1/1.65 and 1/15 for $p = 1$ and $p = 0$, respectively (SelectStar is the straight line at the very top of the figure). Notice that setting $p = 1$ for the noisy data version results in the transmission of all Vec_{S_u}s since no mote satisfies that threshold in the examined data sets. As a result, the aforementioned lower bound of 1/1.65 expresses the worst case gains solely provided by PAOs in-network outlier detection approach.

Sliding the Window. We now investigate the characteristics attributed to PAO when operating over sliding windows, thus approximately pinpointing outlying values and reducing the communication burden by suppressing messages as described in Section 4. Figures 4(a), 4(b) present the accuracy of our framework and the amount of communicated bits for different ϵ values expressed as a percentage of the specified $\measuredangle_{thres}$. We compute the accuracy of PAO using the $F\!-\!measure = 2/(1/Precision + 1/Recall)$ metric where precision specifies the percentage of reported outliers that are true outliers, while recall specifies the percentage of outliers that are reported by our framework. Notably, PAO exhibits high accuracy with F-measure values $\geq 80\%$ in most of the cases while managing to reduce the total amount of communicated data to $1/3.6$ on average compared to the mere transmission of the latest value in the window (for $\epsilon = 0$).

Eventually, Figure 5 shows the corrected answers that would be obtained upon utilizing PAO on par with a simple aggregate (max) query. The parameters used during the outlier detection are included in the figure. In each epoch, we initially computed the maximum humidity reported by the network using the original data sets (AGGREGATION). Then we posed the same aggregate query and



(a) Accuracy varying ϵ (b) Total Bits Transmitted vs ϵ

Figure 4: Approximate Processing over Sliding Window, w=16

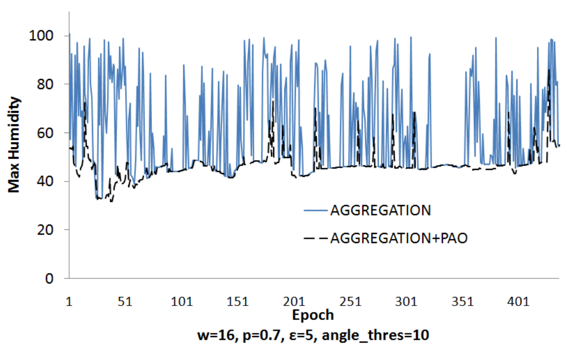


Figure 5: Max Humidity Values after Outlier Removal

		$\measuredangle_{thres}$					
		10			20		
Cluster Size	Buckets	Cmps	Multi-hashes	Hashes Per Bucket	Cmps	Multi-hashes	Hashes Per Bucket
24	1+1	70.45	0	12	70.46	0	12
	2+1	33.72	0.78	8.26	35.88	1.63	8.54
	4+1	16.58	2.21	5.24	18.39	4.65	5.73
36	1+1	160.27	0	18	160.41	0	18
	2+1	77.61	1.22	12.41	81.63	2.12	12.71
	4+1	37.38	3.39	7.88	42.26	7.05	8.61
48	1+1	286.38	0	24	286.80	0	24
	2+1	137.39	1.63	16.54	145.94	3.04	17.01
	4+1	67.39	4.53	10.51	75.33	9.12	11.42

Table 1: Bucket Introduction in PAO+ (w=16, p=0.75)

let it be executed after the outlier detection and removal performed by PAO using $\epsilon = 5$ degrees. We can observe that PAO manages to prevent the distortion on the final results caused by the outliers for the vast majority of the epochs.

PAO+ Leverages. To validate the ability of the hashing mechanism introduced by PAO+ to distribute the intracluster load, as well as to prune unnecessary comparisons, in Table 1 we present the effect of bucket node introduction utilizing disjoint time windows of $w = 16$ size. We used different cluster sizes between 24 and 48 motes while varying the number of buckets from 1 to 4. Moreover, the notation ”+1” used in the number of buckets expresses the utility of an additional bucket for motes that do not succeed in the linearity test. The table provides average results per window. In particular, we include the average number of comparisons (Cmps) that take place in a window along with the average number of motes that sent their data to 2 buckets (Multihashes). Furthermore, we present the average hashes received by a bucket (Hashes Per Bucket). It can easily be deduced that increasing the number of buckets dramatically reduces the number of performed comparisons which validates the usefulness of PAO+ in this particular aspect. On the other hand, the number of multihashes and the number of hashes per bucket regard a transmission cost mainly charged to cluster's regular motes and the load distribution between buckets, correspondingly. The adoption of more buckets, causes an increase in multihashes and a simultaneous decrease in the number of hashes per bucket. This is interpreted as a shift in the energy consumption from clusterhead and bucket nodes to regular cluster motes caused by the increment of bucket nodes' number. Achieving appropriate balance aids in keeping intracluster, uniform energy consumption, which subsequently leads to infrequent network reorganization.

7. CONCLUSIONS

In this paper we presented PAO, an outlier detection framework that manages to perform over multiple window types and allows users to choose between exact or approximate operation. We also devised PAO+'s mechanisms that manage to prune unnecessary comparisons and balance the intracluster load during the outlier detection process. Our experimental evaluation using real world datasets validated that our framework can pinpoint outlier readings ensuring significantly decreased amount of communicated information. It also showed the ability of approximate PAO to provide results of high quality with further reduced bandwidth consumption.

8. REFERENCES

[1] P. Andreou, D. Zeinalipour-Yazti, A. Pamboris, P. K. Chrysanthis, and G. Samaras. Optimized Query Routing Trees for Wireless Sensor Networks. *Information Systems*, to appear, 2010.

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

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

[5] 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.

[6] E. Elnahrawy and B.Nath. Cleaning and querying noisy sensors. In *WSNA*, 2003.

[7] K. Georgoulas and Y. Kotidis. Random Hyperplane Projection using Derived Dimensions. In *MobiDE*, 2010.

[8] N. Giatrakos, Y. Kotidis, A. Deligiannakis, V. Vassalos, and Y. Theodoridis. TACO: Tunable Approximate Computation of Outliers in wireless sensor networks. In *SIGMOD*, 2010.

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

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

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

[12] Y. Kotidis, A. Deligiannakis, V. Stoumpos, V. Vassalos, and A. Delis. Robust Management of Outliers in Sensor Network Aggregate Queries. In *MobiDE*, 2007.

[13] 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.

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

[15] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In *MobiDE*, 2003.

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

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

[18] 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.

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

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

[21] Y. Zhang, N. Meratnia, and P. Havinga. Outlier detection techniques for wireless sensor networks: A survey. *International Journal of IEEE Communications Surveys and Tutorials*, 12(2), 2010.

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