

An Efficient Data Acquisition Model for Urban Sensor Networks

Sérgio S. Furlaneto, Aldri L. dos Santos and Carmem S. Hara

Universidade Federal do Paraná

Curitiba-PR, Brazil P.O.Box 19081 81531-980

Email: {sergio, aldri, carmem}@inf.ufpr.br

Abstract—Applications for Wireless sensor networks (WSN) usually take into consideration the specificity of the environment in which they are deployed in order to save the sensors' limited resources. In particular, the sensing task in urban environments requires hundreds and even thousands of sensors to be spread over the monitored area. Moreover, in environmental monitoring applications, sensors that are closely located usually provide similar readings. That is, spatial proximity is related to data similarity. In this paper we propose SIDS (Spatial Indexing Based on Data Similarity for Sensor Networks), a data model that explores this characteristic in order to provide scalability and efficient query processing on urban WSNs. Scalability is achieved by grouping sensors with similar readings, while efficiency for processing queries relies on two strategies: the concept of *repositories*, which consist of sensors that act as datacenters, and an indexing structure designed for speeding up both spatial and value-based queries. We have implemented the proposed model and results from simulations on a variety of scenarios show that SIDS provides scalability and it outperforms CAG and Peer-tree, which are models that have been proposed for processing data and spatial queries, respectively.

I. INTRODUCTION

Wireless sensor networks (WSNs) are composed of small devices with low processing power and data storage. In general, sensor devices present limited energy and communicate with each other by short-range radios [1]. WSNs have been employed for monitoring and controlling real world data in several indoor and outdoor environments [2]. Systems to support WSNs have been an area of intense investigation in the past years, from low-level protocols [3], [4] to application-level databases [5], [6].

The sensing task usually requires hundreds and even thousands of sensors to be spread over the monitored area. In order to fully explore the sensors' limited resources, applications for WSNs usually take into consideration the specificity of the environment in which they are deployed. In this paper, we consider environmental monitoring of a scalar field, such as temperature, humidity, and intensity of light, in *urban environments*. They are characterized by extreme temperature variation due to heat islands that result from the reduction of green areas, traffic, and pollution. Further, network coverage deserves special attention given that communication may be hindered by obstacles created by buildings and interference from other devices that communicate via radio waves [7]. As opposed to other WSN applications such as wildlife

monitoring, energy saving is not as critical in urban scenarios given that energy can be obtained from alternative sources like lampposts and ambient energy harvesting [8]. Thus, in this paper we consider that sensor devices are supported by power scavenging units.

Due to the large number of sensors that may be involved in urban sensor applications, an important requirement of a data model to support data acquisition in this context is scalability. A common approach for reaching this goal is to group sensors into *clusters*, and electing one or more of them to serve as *cluster-heads* (CHs). CHs are then responsible for storing information of all its members and answering query requests. In certain applications, exploring spatial similarity of data results on a reduction of the number of clusters required to cover the network. This approach is followed by several works, like DACH [9], CAG [10] and DEDAC [11]. However, none of them propose an indexing mechanism to reduce the number of cluster-heads to be contacted during query dissemination to only the ones holding relevant information. The goal of an index is to prune the search space, reducing the number of messages on the network, and thus providing search results faster and with lower cost. Among data models that provide an indexing mechanism are GHT [3] and Peer-tree [12].

In this paper we propose a model and indexing structure for urban WSNs called SIDS (Spatial Indexing Based on Data Similarity for Sensor Networks). SIDS explores both data similarity for clustering sensors, and an indexing mechanism for speeding up in-network query processing. We also introduce the notion of *repositories*, which are regions in the monitored area that concentrate information. They can be viewed as datacenters in WSNs. Both model and index structure are inspired by an urban environmental sensing scenario, where sensors closely located are likely to present similar readings. In this context, sensor readings change gradually, and thus they can be characterized as read-intensive applications. The index structure supports both spatial queries, for obtaining readings on a given geographical location, and value-based queries, for retrieving the location of sensors with readings in a given range. To the best of our knowledge, it is the first model that combines spatial similarity of readings with an indexing structure for speeding up both types of queries in the context of urban WSNs.

We have conducted a performance evaluation, based on simulations, comparing SIDS with two other models: CAG,

which has been proposed to support value-based queries, and Peer-tree, which supports spatial queries. Our results show the efficacy of our approach. For processing spatial queries, SIDS performs better than Peer-tree. For value-based queries, the cost for processing a single query on CAG is better than for SIDS. However, when the query set is of size at least two than SIDS outperforms CAG. We have also conducted simulations for evaluating the scalability of the proposed model.

The paper is organized as follows. Section II presents related work. Section III describes the SIDS model and index structure. Section IV presents a performance evaluation, and we conclude in Section V highlighting future works.

II. RELATED WORK

Data gathering from WSNs, which in the past was based on simple commands, has shifted to more complex languages for query processing. Existing data models for WSNs can be classified as being based on database, data similarity, or spatial proximity. Among database models, we can mention two: Cougar [5] and TinyDB [6]. Both provide a relational view of sensor readings, which are stored at the sensors that collect them. This limits their applicability to small to medium size WSNs. In order to reduce the number of sensors to be contacted for data acquisition, several data models for clustering sensors based on readings similarity have been proposed, such as CAG, DEDAC, and DACH. CAG [10] clusters sensors while disseminating a query. This approach provides high data accuracy and decreases the volume of data sent back to the base station, since only one message per cluster is transmitted. However, the time spent for clustering at each new query may be costly. DEDAC [11] was proposed for reducing this cost by storing cluster members information at cluster-heads. However, the model does not support any indexing mechanism, such as the one proposed by DACH [9].

Techniques for cluster-head election have been proposed in the literature, based on several criteria, such as rotation among cluster members [13], residual energy [11], and proximity to other nodes [14]. Spatial proximity has also been used as a criterion for indexing mechanisms for WSNs, such as GHT, CBI, Peer-Tree, and DQT. GHT [3] has been proposed for efficiently answering value-based queries. It is based on a hash function that maps attribute names to a coordinate in the monitoring area. Then the sensor closest to this coordinate is responsible for storing all readings of the given attribute. CBI [15] considers a graph view of the WSN and builds an indexing structure based on the dominating set of the graph. Sensors in the dominating set are considered cluster-heads that store information on all sensors connected to them. An indexing structure is defined on these CHs for processing value-based queries. Both Peer-tree [12] and DQT [16] propose in-network clustering formation and consider that a query can be injected to the network at any node. That is, they do not rely on the processing power of a base station. Peer-tree targets spatial queries by constructing an index similar to R-trees [17]. On the other hand, DQT partitions the WSN into grid cells, which are in turn indexed by a tree structure.

SIDS, the model we propose in this paper, borrows the idea of clustering sensors based on the similarity of their readings from CAG. This approach ensures high accuracy of query results without contacting all sensor devices involved, but only their CHs. Moreover, giving that in urban scenarios readings have spatial similarity, this clustering strategy yields a smaller number of clusters than the grid model adopted by DQT. Similar to some of the models described, SIDS adopts a tree-based indexing structure in which each node keeps the value intervals of their children. However, in order to be used for processing spatial queries, the tree hierarchy in SIDS is determined by the spatial correlation among clusters. The concept of repositories as datacenters, which we believe is novel in the context of WSNs, has the same goal as GHT, but their location and the set of data they hold are determined by the clusters' reading and location. In short, SIDS is a WSN data acquisition model that takes into consideration some ideas from previous works while extending them for urban monitoring scenarios.

III. SIDS DATA MODEL

A wireless sensor network (WSN) is represented as a graph $G = (V, L)$, where $V = \{s_1, \dots, s_n\}$ is a set of sensors spread over a monitored area M and L is a set of links such that (s_i, s_j) is in L if s_i and s_j are within the radio communication range of each other. We say that the distance between s_i and s_j is *one-hop* and that s_i and s_j are *neighbors*. Communication between two arbitrary sensors requires the existence of links $\{(s_1, s_2), (s_2, s_3), \dots, (s_{n-1}, s_n)\}$ such that s_1 is the sensor that originates the message and s_n is the message final destination. That is, WSNs are based on *multi-hop* communication, and rely on a *routing* protocol for determining paths that establish communication between sensors. In our model, we assume that sensors are static, and thus have a fixed geographic coordinate. To simplify our discussion, we also assume that each sensor is responsible for monitoring a single measurement from the environment.

Spatially close sensors are grouped into *clusters* based on a *characteristic reading* (C_r) and a user-defined threshold τ . That is, sensors in a cluster c have readings between the *minimum* and *maximum cluster limits*, given by $C_r(c) \times (1 - \tau)$ and $C_r(c) \times (1 + \tau)$, respectively. Moreover, clusters are defined on contiguous spatial areas. That is, for every sensor s_i in a cluster c , there exists a sensor $s_j \in c$ such that s_j is at most one-hop distant from s_i .

Every cluster have one or more sensors that serve as cluster-heads (CHs), at which data from all sensors that belong to the cluster are stored. In our model, CHs are devices in border regions of a cluster and are one-hop distant from at least one neighbor CH. As an example, sensors s_3 and s_{72} are CHs for the cluster identified by G in Figure 1. The number of CHs for each cluster is determined by our concept of *repository*. A repository is a group of at least two CHs on a border region, each of them belonging to different clusters, and one-hop distant from each other. Intuitively, given that CHs contain sensing data for every cluster member, a repository can act as

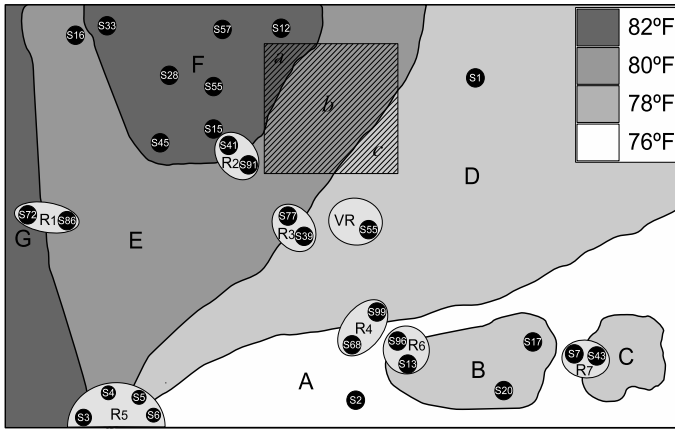


Fig. 1. Sensor Field

datacenters to answer queries referring to any of the clusters that compose it. As an example, in Figure 1, repository R_5 is composed of CHs $s_3, s_4, s_5,$ and s_6 , while repository R_2 is composed of CHs s_{41} and s_{91} .

The number of CHs and repositories is minimum in the following sense: every CH must belong to a repository, and there exist no two repositories that refer to the exact same set of clusters. In the example of Figure 1, there are seven repositories, each of them with CHs from different sets of clusters, such as R_1 for clusters $\{E, G\}$, R_2 for clusters $\{E, F\}$, R_5 for clusters $\{A, D, E, G\}$, and so on. Observe that sensors s_{16} and s_{33} could have been chosen as CHs, since they are sensors in border regions. However, this would violate our minimality rule by defining two repositories for the same set of clusters $\{E, F\}$. Thus, only repository R_2 , composed of CHs s_{91} (for E) and s_{41} (for F) is defined. Our approach for choosing $R_2 = \{s_{41}, s_{91}\}$ over $\{s_{16}, s_{33}\}$ is the following: repositories, and consequently CHs, are placed as close to the center of the sensing field M as possible. Intuitively, since we consider that queries can be injected to the WSN at any sensor node (and not at a centralized access point), it is likely that messages towards the center of the field reach the query target in smaller number of hops.

Although we do not consider a single access point for query injection, we assume the existence of a commodity computer for coordinating the sensors' clustering process at the beginning of the WSN operation. Although not detailed in this paper, the maintenance of the clustering model and indexing structure, presented in the next section, can be executed in the network without relying on the base station.

A. Indexing Structure

Given that sensors have been grouped based on their spatial locality and data similarity, we are now ready to define our indexing structure. The index has been designed for speeding up one-shot spatial and value-based queries. In our model, both sensing data and the index structure are stored in the sensor network. That is, once the index structure has been built, search queries can be processed in the WSN, without forwarding all sensing data to the base station.

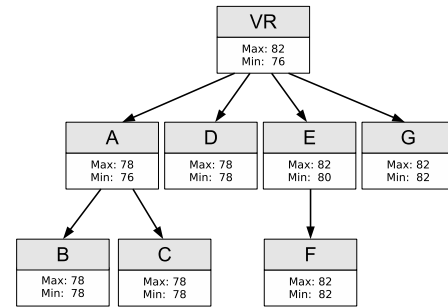


Fig. 2. SIDS index structure for sensor field in Figure 1.

The index has been motivated by an urban sensing scenario in which “heat islands”, or small regions inside larger areas, have distinct characteristics compared to their surroundings. We thus propose a tree structure in which the hierarchy of nodes is determined by the spatial containment relation among clusters. That is, for each cluster c , we determine its minimum bounding box ($mbb(c)$), which corresponds to the smallest rectangle within which all sensors in c lie. The tree hierarchy is then built by setting a cluster c_i as a child of cluster c_j if $mbb(c_j)$ is the “smallest” region that contains $mbb(c_i)$. That is, the outer object contains the inner object and does not touch the boundaries of the inner object. The complete coverage of one area by another characterizes “islands” of sensing data, as opposed to adjacent areas with distinct readings. As an example, in Figure 1, cluster A is the smallest one that completely covers clusters B and C . On the other hand, although $mbb(D)$ and $mbb(E)$ do intersect, they are not related by an “island” relationship, but are adjacent areas with distinct readings.

Figure 2 shows the cluster hierarchy of the example depicted in Figure 1. Each cluster corresponds to a node in the tree such that parent-child links correspond to “island” relationships. Moreover, there exists a distinct root node which points to all top-level clusters on the sensing area. In Figure 1 this root node is depicted as repository v_r , showing its placement at the sensor closest to the center of the monitored area. Next, we describe the placement of other elements of the data model and indexing structure.

B. Storage Model

Consider a cluster c , consisted of a set of sensors S and with a set of CHs H . Each sensor $s \in S$ stores the following information, which we denote as $sensorData(s)$: its geographical position, current reading, the minimum and maximum cluster limits and a list of CHs in H ordered by their distance from s . By ordering the elements in H , a sensor is able to forward a query message to its closest CH, while informing all of them of its reading updates. The limit values play an important role in the maintenance of the clustering model. That is, whenever a sensor obtains a new reading, it may locally check whether it must leave its current cluster and look for a new one in its neighborhood.

In addition to these information, a sensor $h \in H$ that plays the role of a CH stores information both on the clustering

($clusterData(h)$) and on the indexing structure ($indexData(h)$). Among $clusterData(h)$, the following is stored at h : the cluster's limit values, its minimum bounding box (mbb), set of CHs for cluster c , set of cluster members, and set of CHs (of distinct clusters) in the same repository to which h belongs. Now let n be the node in the indexing tree T that corresponds to cluster c . The $indexData(h)$ stored at h includes the CH sensor of its parent that is located closest to h . In addition, for each child n' of n in T , $indexData(h)$ includes the following: a set of CHs for n' , associated with their repositories, the number of CHs in the same repository that correspond to its descendants, the minimum and maximum readings of all sensors in the subtree rooted at n' , and its mbb .

Observe that in the proposed index structure, information on all sensors that compose a cluster are kept at the CH. In some cases, the volume of data can be large, and CHs may only keep a view of the base data, which can be on the form of a histogram, or an aggregate value. Reducing the storage volume of raw sensing data has been an area of intense investigation in the past years [18], [19], [4], but this subject is outside the scope of this paper. It is important to point out, though, that any of the existing approaches can be used in conjunction with our indexing structure.

It is worth noticing that given the distributed and failure-prone nature of WSNs, our approach defines two levels of replication. The first consists of the replication of cluster's information at every sensor that play the role of a CH. The second consists of the replication of a subset of $clusterData$ at the parent CH. Both have been designed to minimize the number of inter-sensor communication for processing both spatial and data queries as described in Section III-C. In addition, our replication and storage models may constitute the basis for fault-tolerant clustering with incremental updates. Although one can advocate that the cost of replication is too high to be practical in the context of WSNs, we argue that this is not the case for urban sensing environments, which are read-intensive, and where sensing data are likely to remain stable for periods of time.

C. Query Processing

In this section, we present a search algorithm based on our index structure. Recall that value-based queries are defined as queries that return the set of sensors (and their associated information) with readings in a given interval. Similarly, spatial queries return the set of sensors in a given geographical area. Based on this observation, SIDS has been designed to support both types of queries by placing CHs on border regions such that repositories may hold information on as many sensors in the result set as possible.

The search algorithm explores this idea when choosing a repository to forward a query in a top-down traversal of the tree. That is, suppose that the algorithm is visiting a cluster-head and the query must be forwarded to a subset D of its children. Given that a cluster does not have a single CH, but a set of them, we choose the ones that are located in the same repository. As a result, a single message can be sent

Algorithm $search(queryId, entryPoint, upDown, cSet, searchRange)$
Input: a $searchRange$ sent to a sensor $entryPoint$
Output: set of sensors' locality (or readings) in $searchRange$

1. **if** s is not a CH
2. **then send to** closest CH c of s :
3. $search(queryId, entryPoint, up, cSet, searchRange)$;
4. $resultSet \leftarrow$ responses from c ; **return** $resultSet$;
5. let s be a CH for cluster c in repository r ;
6. **if** s has already executed $queryId$ or $c \in cSet$
7. **then** discard query; **return**;
8. (* forwards query upwards*)
9. **if** s is not the index root node v_r and $upDown = up$
10. **then send to** parent of s :
11. $search(queryId, s, up, cSet, searchRange)$;
12. (* collects cluster members in $searchRange$ *)
13. $resultSet \leftarrow \{\}$;
14. **if** mbb (or subtree's readings) of cluster c intersects $searchRange$
15. **then** $resultSet \leftarrow resultSet \cup$ sensors in c with position (or reading) in $searchRange$;
16. (* forwards query to CHs in the same repository ($rSet$ *)
17. **for each** CH h in the same repository as s
18. **do if** h is CH for cluster c' and $c' \notin cset$
19. **then send to** h :
20. $search(queryId, s, down, cSet \cup rSet, searchRange)$;
21. $cSet \leftarrow cSet \cup rSet$;
22. (* finds children with sensors in $searchRange$ *)
23. $childCluster \leftarrow \{\}$;
24. **for each** cluster c' that is child of c
25. **do if** mbb (or subtree readings) of c' intersects with $searchRange$
26. **then** $childCluster \leftarrow childCluster \cup \{c'\}$;
27. $childCluster \leftarrow childCluster \setminus cSet$;
28. (* finds repositories with largest number of CHs *)
29. $destRep \leftarrow \{\}$;
30. **while** $childCluster \neq \{\}$
31. **do for each** repository r'
32. **do** $rep[r'] \leftarrow$ set of clusters c' in $childCluster$ such that r' contains a CH for c' ;
33. $destRep \leftarrow destRep \cup \{r'\}$, where r' is the repository with $rep[r']$ with the largest cardinality or largest number of descendent CHs;
34. $childCluster \leftarrow childCluster \setminus rep[destRep]$;
35. **for each** repository r' in $destRep$
36. **do send to** r' : $search(queryId, s, down, cSet, searchRange)$;
37. $resultSet \leftarrow$ responses from parent, clusters in the repository and children;
38. **send to** $entryPoint$: $resultSet$;

Fig. 3. Algorithm for data and spatial search

to the repository, which can then forward it to the CHs that compose it. An algorithm that implements this idea is given in Figure 3. Observe that Algorithm $search$ supports both types of queries. For spatial queries, parameter $searchRange$ refers to a geographical region, while for value-based queries, it is a value interval.

The search algorithm considers that the query can be injected to the WSN from any sensor on the field, which in the algorithm is denoted as $entryPoint$. If $entryPoint$ is not a CH, but a member of cluster c , then it forwards the query to its closest CH (Lines 1-4). If $entryPoint$ is a CH then the query is forwarded upwards until it reaches the index root node, since its ancestors may contain sensors in the $searchRange$ (Lines 9-15). The query is also forwarded to CHs in the same repository as the current CH (Lines 17-21). Before forwarding the query downwards in the tree, the algorithm first computes in $childCluster$ the subset of its children that may contain sensors in $searchRange$ (Lines 23-27). Observe that here the *intersection* operation refers to value intersection for

value-based queries and geometry overlap for spatial queries. For determining the repository to which forward the query, we first associate with every repository r' , a set denoted as $rep[r']$ containing all sensors c' such that c' is CH for a cluster in $childCluster$ (Line 32). We then insert in $destRep$ the repositories containing the *largest* number of clusters in the search space (Line 33). The query is then forwarded to repositories in $destRep$. The sensor waits for result sets from its children, parent, and clusters from the same repository before forwarding them to the sensor from which it received the query (its previous *entryPoint*) (Lines 35 to 38).

As an example, consider a spatial query to obtain sensor readings at the square region that consists of subareas a, b and c depicted at Figure 1. Suppose the query is injected to the WSN at sensor s_2 . Since s_2 is not a cluster-head, it forwards the query message to s_{68} , its closest CH for cluster A . From s_{68} the message is forwarded to s_{99} because both are members of repository R_4 . s_{99} is a CH for D and thus it already contains sensor readings for subregion c . From s_{68} query messages are not forwarded to cluster A 's children because neither $mbb(B)$ nor $mbb(C)$ intersect the query target. Following the upward traversal on the tree, the query message is forwarded to A 's parent, which is the root node v_r at sensor s_{55} . At v_r it is determined that its descendents D, E and F contain mbb that intersect the area of interest. Since sensors in D have already been obtained from s_{99} (D is in $cSet$), then v_r chooses repository R_2 to forward a single query message. This is because R_2 contains CHs for both E (s_{91}) and F (s_{41}).

IV. PERFORMANCE EVALUATION

We have conducted simulations in order to determine SIDS efficacy and compare its cost for processing queries with other existing models. In order to determine SIDS efficacy, we have considered three metrics. The first determines the effect of our similarity-based clustering strategy on the accuracy of the results returned by spatial queries. The other two tackle the question of the model's scalability by determining the cost of the clustering algorithm and also the storage requirement on the CHs. In other words, our goal is to determine how the size of the WSN affects the network deployment, and if the model is compatible with current sensor devices' storage capacity. We have also conducted simulations to compare the cost of processing queries on SIDS with Peer-tree [12], which has been proposed for processing spatial queries, and CAG [10], proposed for supporting value-based queries. They have been chosen based on their similarities with SIDS.

A. Simulation Settings

SIDS, Peer-tree, and CAG have been implemented on network simulator NS2 version 2.34, with Destination Sequenced Distance Vector routing protocol (DSDV). The radio range of every sensor on the field was set to 30 meters. We have generated synthetic scenarios where sensors were randomly placed on the field. Sensors' readings were generated by a Matlab tool [20], which has been especially designed to produce spatially correlated sensor data. The tool receives

as input a correlation coefficient (h) and the size of the monitoring area (m). It generates as output a matrix D of dimension $m \times m$, used to determine sensors' readings as follows: each sensor s at position (x_s, y_s) gets as reading the value at $D[[x_s], [y_s]]$. The correlation coefficient determines the level of similarity. That is, $h = 0$ generates data with no spatial similarity, and higher values of h induces higher spatial correlation. All simulations described in this section have been executed on scenarios generated with high correlation ($h = 9$). The cost measure is given in number of transmissions (or hops). Although this metric differs from energy consumption on absolute values, there is a direct correspondence between them, since communication cost is usually much higher than in-place processing cost [21].

Simulation studies for determining precision, clustering and storage costs considered a 500×500 square meter monitored area and five synthetic data sets have been generated with $h = 9$ and $m = 500$. The network density (d), that is, the number of sensors on the field, varied from 200 to 1000, with steps of 200, and for each density we've generated five different snapshots of the sensing area, based on each data set. The following similarity thresholds (τ) have been considered for clustering sensors: 0%, 1%, 2%, 4%, 6%, 8%, and 10%. Recall that τ is a user-defined value that in conjunction with a characteristic reading C_r determine the minimum and maximum cluster limits.

B. Precision

This simulation has been conducted in order to determine the accuracy of spatial query results. This is an important measurement both for analyzing if the clustering strategy is appropriate for a given application and also for guiding the choice of the similarity threshold τ . For each combination of τ and network density (d), 10 spatial queries were injected to obtain the sensor reading at a given coordinate $c = (x, y)$. Thus, the total number of queries of the simulation was 350. The relative error (ϵ) of the result is given by $\epsilon = |1 - (r_s/r_c)|$, where r_c is the actual reading at coordinate c and r_s is the characteristic reading of the cluster that contains the coordinate, returned as the query result.

Relative errors shown in Figure 4(a) are the average for all network densities obtained for each value of τ . When $\tau = 0\%$ the precision is almost absolute ($\epsilon = 0.0045\%$) since in most cases clusters consist of a single sensor and thus the query result is obtained from the sensor itself. It can be observed that the relative error increases with higher values of τ . However, in most cases, the relative error is at most half of the value of τ . Indeed, for $\tau = 4\%$ we have $\epsilon = 1.6388\%$ and for $\tau = 10\%$, $\epsilon = 3.5952\%$. This shows that the effect of increasing the similarity threshold, which may be necessary for coping with larger sensor density or monitoring area, may impact the accuracy of the results by only half of the increase on τ . It is important to point out that the reported relative errors in these simulations are the maximum that may result from our clustering model, since we consider that the query result is based solely on the cluster characteristic reading, without the

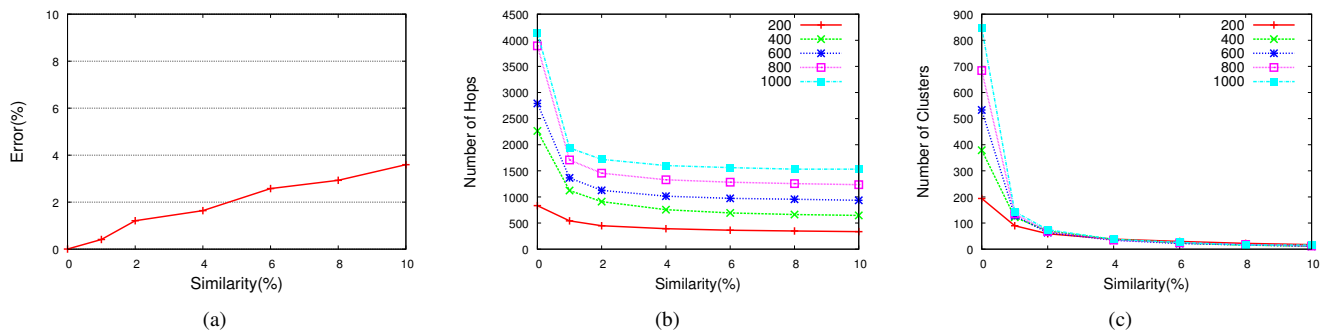


Fig. 4. (a) Relative error of the characteristic reading (b) Cost of clustering (c) Number of clusters

use of any prediction or aggregate operation on actual cluster members readings.

C. Clustering

In this section we examine the cost of SIDS for sensor clustering at network deployment. To this end, we have collected the number of hops needed for disseminating a clustering message on the WSN, collecting all sensors' readings at the base station and disseminating the clustering information back to every sensor. The results shown in Figure 4(b) is the average number of hops obtained from five simulations for each combination of τ and d .

When the similarity threshold τ is set to 0% almost all sensors on the field constitute a singleton sensor cluster. Indeed, the number of clusters is almost the network density, as shown in Figure 4(c). Moreover, the number of transmissions for clustering formation is more than three times the number of sensors on the field. This is because when collecting information for clustering, only the sensor with the cluster characteristic reading sends the cluster members information back to the base station. Thus, when $\tau = 0\%$ almost all sensors send distinct messages back to the base station in reply to the clustering message. The decrease on the number of transmissions for larger clusters (and consequent smaller number of clusters) is also due to the decrease on the messages sent back to the base station. Since we are considering a scenario with high level of similarity on sensor readings, this decrease is considerable even for a similarity threshold of 1%. Indeed, for a density of 200 sensors and $\tau = 0\%$, the number of hops is 833 and the number of clusters is 194. These values reduce to 543 and 90 for $\tau = 1\%$. A steeper drop can be observed for a density of 1000 sensors: the number of hops reduces from 4145 to 1938, and the number of clusters decreases from 848 to 143 for $\tau = 1\%$. For larger values of τ the decrease on these values are not so substantial. These results show that the expected impact on these two measures for scenarios with lower levels of similarity on sensor readings than the one considered in our simulations is the following: both the number of hops and the number of clusters reduce with the increase on the similarity threshold τ ; however, the drop is abrupt for high levels of similarity and easier for lower levels. Given that smaller number of clusters result in larger number of cluster members, the question of storage cost on

Density	Maximum (bytes)	Average (bytes)
200	1882	108,506
400	4828	212,742
600	7150	308,199
800	9678	369,758
1000	11910	414,987

TABLE I
STORAGE COST FOR SIMILARITY THRESHOLD OF 10%.

CHs have an impact on the model's scalability. Next section reports simulation results on this issue.

D. Storage

Recall that in SIDS we do not consider the application of any aggregation operation on data collected from cluster members. Thus, the results reported in this section consist of the cost on the CHs for storing both the clustering and indexing structure without any data compression or aggregation technique. As detailed in Section III-B, the information stored at a CH h can be divided into two groups: $clusterData(h)$ and $indexData(h)$. The size of $clusterData(h)$ is 34 bytes in addition to 12 bytes for each cluster member. The size of $indexData(h)$ is 8 bytes plus 32 bytes for each child node.

Given that with the similarity threshold of 0% most of clusters are singleton sets and there is no "island" relationship to create an index hierarchy, the average storage cost ranges from 52 bytes for a network of 200 sensors and 54 bytes when the density increases to 1000. The storage cost increases with higher similarity thresholds, and the average and maximum values when $\tau = 10\%$ is given in Table I. The average cost on CHs in number of bytes is roughly half the number of sensors on the field. The maximum storage cost, on the other hand, ranges from 9 to 12 times the number of sensors. This is because with the similarity threshold of 10%, almost all sensors belong to a single cluster, as shown in Figure 4(c), in addition to several peripheral smaller clusters. As an example, the maximum cost for a density of 1000 sensors is a cluster composed to 949 devices, which requires almost 12 Kb of storage. Although this cost is significant, the required amount of storage is feasible for current sensor devices capacity even for the worst case scenario considered in these simulations.

E. Query Processing

Simulations described in this section compare the cost of processing queries on SIDS with two existing models: Peer-tree [12] and CAG [10]. We have considered a 200×200

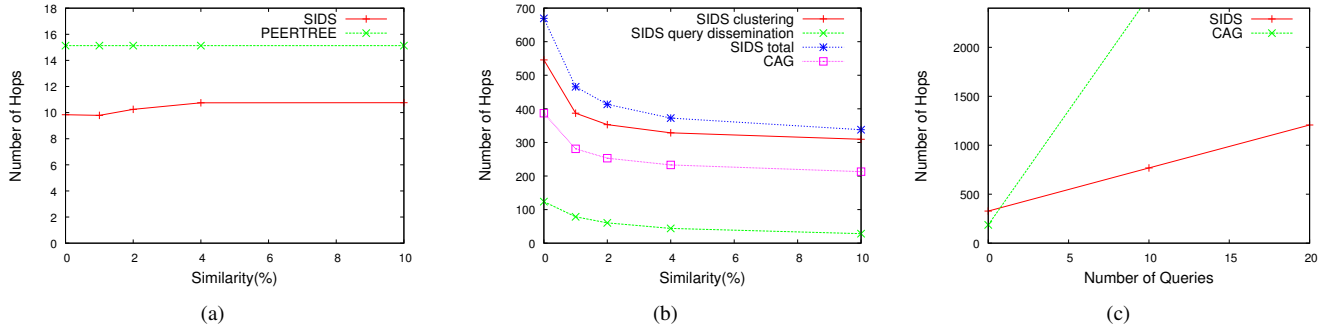


Fig. 5. Number of hops for processing queries (a) Spatial queries (b) Value-based queries (c) Cumulative cost for value-based queries

square meter sensing area with 100 sensors. Ten different synthetic data sets have been generated using the tool described in Section IV-A. The values reported in this section correspond to the average query processing cost over all scenarios. The sensing area extension, number of sensors, and other simulation settings were chosen based on the experimental evaluation of CAG, described in [10].

1) *Spatial Queries*: In this section we compare the cost of processing spatial queries on SIDS and on Peer-tree. Peer-tree groups sensors based on their spatial locality and two user-defined parameters: *distance* and *degree*. *distance* determines the maximum distance (in hops) a cluster member can be from its CH, while *degree* determines the minimum (*degree*) and maximum ($2 \times \text{degree}$) number of cluster members. In all experiments, we have set *distance* to 3 and *degree* to 7. Moreover, Peer-tree does not assume the existence of a base station for generating a tree structure in which the *mbb* of a higher-level cluster contains the *mbb* of its descendants. It relies on in-network processing, with each sensor searching for a CH at increasing number of hops.

For determining the cost of processing spatial queries, we have randomly generated 50 (x, y) coordinates in the sensing area, and for each of them a query has been injected for obtaining the reading from the closest sensor to this coordinate. Figure 5(a) presents the average cost collected from 500 executions for each similarity threshold.

Observe that since Peer-tree clustering strategy is not based on spatial similarity correlation, the query processing cost does not vary with the similarity threshold τ . The almost invariant cost for SIDS can be explained as follows. In the simulations we assume that a query can be injected to the system at any sensor, and not necessarily from the base station. For all queries we have chosen the sensor with the smallest identification (*sensor zero*) as the one that first receives the query message. Observe that *sensor zero* can have been placed anywhere on the sensor field by our scenario generator. According to our search algorithm, *sensor zero* forwards the query to its CH, which in turn forwards to its parent, until it reaches the root node, placed at the center of the sensing area. At each CH the message also traverses the tree top-down following every cluster with a minimum bounding box that contains the search coordinate. All these messages are sent in parallel, and the results are sent back to *sensor zero*. Given that CHs are chosen to be closest to the center of the

sensing area, this search path involves a traversal to the center and one or more back to the borders. Moreover, the size of the path does not vary much with the number of CHs on this path. Since the similarity threshold mainly affects the size of clusters and consequently the number of CHs, they do not have a big impact on the performance for processing spatial queries. Observe that although we have chosen to execute queries that involve a single coordinate, similar results are expected for queries involving a geographical area.

The results show that the cost for processing spatial queries on Peer-tree is slightly higher than on SIDS. This is because Peer-tree chooses CHs randomly, based on the message dissemination pattern in the clustering phase. For SIDS, on the other hand, we induce a traversal towards the center of the sensing area, by placing the index root at this point, and choosing repositories with the minimum distance to the root.

2) *Value-based Queries*: For comparing the cost of processing value-based queries on SIDS and on CAG, we have generated 10 range queries, one for each sensing area scenario. For determining the maximum cost of this type of query, in each of them the range interval was set to be the minimum and maximum sensing values on the entire network. The average number of hops required for processing a single query is presented in Figure 5(b).

The clustering strategy adopted by SIDS and CAG are similar. They are both based on the spatial correlation of sensor readings, but differ on when clusters are built. In CAG sensor clustering is executed on demand, at every query injected to the system. Thus, sensors do not store any clustering information. When a query is submitted, sensor data are grouped at CHs based on their current readings and in order to reduce the number of messages sent back to the base station only CHs are responsible for this task. SIDS, on the other hand, groups sensors at the base station at network deployment and stores clustering and indexing information at CHs pro-actively. SIDS follows an in-sensor maintenance process whenever a cluster member reading falls outside the cluster value limits. In the graph of Figure 5(b), the cost for CAG consists of the number of transmissions required for clustering sensors and for all CHs to send the clustering / query result back to the base station. For SIDS, on the other hand, the total cost for processing a single query, represented as “SIDS total”, is divided into two parts: the cost for clustering sensors (“SIDS clustering”), and the cost for disseminating a query from any sensor

device and sending the result back to this point (“SIDS query dissemination”). The cost of clustering consists of the number of transmissions for collecting sensors’ information at the base station, and sending the clustering and indexing information back to all devices. Observe that the number of transmission for clustering alone is higher than the cost for processing a query on CAG. This is because on CAG, no clustering information is sent back to the sensors, given that sensors do not store any clustering information. The query dissemination cost on SIDS consists of the number of transmissions for contacting all required repositories, and sending the results back to the query entry point.

As shown in the graph, the similarity threshold of 0% yields the highest cost on both models. The number of hops is 387 for CAG and 669 for SIDS, where 546 hops corresponds to clustering, and 123 hops to the dissemination cost. With the increase on the similarity threshold, the number of CHs decreases and so does the total cost on both models. As an example, when $\tau = 4\%$, the average number of clusters is 12 for both models, and the number of hops for processing a query on CAG is 233, while for SIDS it is 372 hops (328 for clustering plus 44 for query dissemination).

The total cost for processing a single query is higher on SIDS than on CAG. However, if a *set* of queries on the same sensing attribute is submitted to the network in a time interval within which sensor readings updates do not incur changes on cluster membership, the cost for processing this *set* is lower on SIDS. This characteristic is illustrated in Figure 5(c), which shows the average cost for processing sets of increasing number of range queries when the similarity threshold has been set to 4%. In this graph, the cost presented when the set of queries is zero corresponds to the number of hops for sensor clustering in each of the models (187 for CAG and 328 for SIDS). Given that CAG follows a reactive approach for sensor clustering, this cost is replicated for processing each query in the sequence. Thus, the addition of a new query in the set increases the total cost by 233 hops. For SIDS, if no clustering updates are considered, the cost of an additional query in the set is 44 hops. Thus, the cost for processing a set of 2 queries on CAG is 466 (233×2) while on SIDS, it is 416 ($328 + 44 \times 2$). This shows that for read intensive applications, SIDS presents a better performance than CAG.

V. CONCLUSION AND FUTURE WORK

In this paper we propose SIDS, a model and indexing structure to support query processing on WSNs. Our proposal has been inspired on characteristics usually found in urban sensing applications, such as spatial correlated data and readings that change gradually over time. SIDS can efficiently support both spatial and value-based queries. Results from simulations show that SIDS presents better performance than Peer-tree for processing spatial queries. A comparative analysis between SIDS and CAG shows that SIDS presents better performance if at least two queries are injected to the system on the same sensing area snapshot. There are still a number of issues to investigate in the future. Some of them are related to moving

towards self management and self configuration capabilities, consider heterogeneity of nodes on the WSN, as well as experiments with real data sets.

Acknowledgment. This work is supported by Fundação Araucária CP 14/2008. The authors would like to thank Angelo Brayner for his valuable suggestions.

REFERENCES

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, Aug. 2002.
- [2] L. B. Ruiz, J. M. Nogueira, and A. A. F. Loureiro, “MANNA: a management architecture for wireless sensor networks,” *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116–125, 2003.
- [3] S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu, “Data-centric storage in sensornets with ght,” *Mobile Networks and Applications*, vol. 8, no. 4, pp. 427–442, 2003.
- [4] A. C. Frery, H. Ramos, J. Alencar-Neto, and E. Nakamura, “Error estimation in wireless sensor networks,” in *Proc. of the Symp. on Applied Computing*. New York, NY, USA: ACM, 2008, pp. 1923–1928.
- [5] Y. Yao and J. Gehrke, “The cougar approach to in-network query processing in sensor networks,” *SIGMOD Record*, vol. 31, no. 3, pp. 9–18, 2002.
- [6] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “Tinydb: an acquisitional query processing system for sensor networks,” *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, 2005.
- [7] R. N. Murty, G. Mainland, I. Rose, A. Chowdhury, A. Gosain, J. Bers, and M. Welsh, “Citysense: An urban-scale wireless sensor network and testbed,” in *IEEE Conf. on Technologies for Homeland Security*, 2008.
- [8] J. A. Paradiso and T. Starner, “Energy scavenging for mobile and wireless electronics,” *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 18–27, Mar. 2005.
- [9] X. Wu, P. Wang, W. Wang, and B. Shi, “Data-aware clustering hierarchy for wireless sensor networks,” in *Proc. of the Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, 2008, pp. 795–802.
- [10] S. Yoon and C. Shahabi, “The clustered aggregation (cac) technique leveraging spatial and temporal correlations in wireless sensor networks,” *Transactions on Sensor Networks*, vol. 3, no. 1, pp. 3–41, 2007.
- [11] Y. Zuang, H. Wang, and L. Tian, “Energy and data aware clustering for data aggregation in wireless sensor networks,” in *Proc. of the IEEE Int. Conf. on Mobile Adhoc and Sensor Systems Conf.* Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 1–6.
- [12] M. Demirbas and H. Ferhatosmanoglu, “Peer-to-peer spatial queries in sensor networks,” in *Proc. of the IEEE Int. Conf. on Peer-to-Peer Computing*, 2003, p. 32.
- [13] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” *IEEE Trans. on Wireless Comm*, vol. 1, no. 4, pp. 660–670, 2002.
- [14] S. Ghiasi, A. Srivastava, X. Yang, and M. Sarrafzadeh, “Optimal energy aware clustering in sensor networks,” *Sensors*, vol. 2, pp. 258–269, 2002.
- [15] Y. Wu and Y. Li, “Distributed indexing and data dissemination in large scale wireless sensor networks,” in *Proc. of the Int. Conf. on Computer Communications and Networks*, 2009, pp. 1–6.
- [16] M. Demirbas and X. Lu, “Distributed quad-tree for spatial querying in wireless sensor networks,” in *Proc. of the IEEE Int. Conf. on Communications*, 2007, pp. 3325–3332.
- [17] A. Guttman, “R-trees: A dynamic index structure for spatial searching,” in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 1984, pp. 47–57.
- [18] A. Brayner, A. Lopes, D. Meira, R. Vasconcelos, and R. Menezes, “An adaptive in-network aggregation operator for query processing in wireless sensor networks,” *J. Syst. Softw.*, vol. 81, pp. 328–342, 2008.
- [19] I. A. Reis, G. Câmara, R. A. ao, A. Miguel, and V. Monteiro, “Data-aware clustering for geosensor networks data collection,” in *Simpósio Brasileiro de Sensoriamento Remoto*, 2007.
- [20] A. Jindal and K. Psounis, “Modeling spatially correlated data in sensor networks,” *Transactions on Sensor Networks*, vol. 2, pp. 466–499, November 2006.
- [21] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, “A taxonomy of wireless micro-sensor network models,” *ACM SIGMOBILE Mobile Computing and Communication Review*, vol. 6, no. 2, pp. 29–36, 2002.