# An Autonomic in-Network Query Processing for Urban Sensor Networks

Marcos A. Carrero*, Rone I. da Silva†, Aldri L. dos Santos*, and Carmem S. Hara*
*DINF – Universidade Federal do Paraná – UFPR – Paraná, Brazil
†DTECH – Universidade Federal de São João del-Rei – UFSJ – Minas Gerais, Brazil
Email: {macarrero, aldri, carmem}@inf.ufpr.br, rone@ufsj.edu.br

***Abstract*—The sensing of urban environments usually takes into account the deployment of a large number of devices to measure their environmental attributes, such as temperature, pressure, humidity, luminosity and pollution. In such applications, nearby sensors usually produce similar readings due to their spatial and temporal correlation. In the era of big data, management of collected data requires autonomous and scalable Wireless Sensor Network (WSN) structures. In this paper, we propose an in-network data storage model, called AQPM, that provides efficient processing of both *spatial* and *value-based* queries. AQPM is *autonomous* and *scalable*. That is, it does not rely on any central entity for neither managing data storage on sensor devices nor for processing queries. Scalability is achieved by grouping sensors with similar readings into *clusters*, while efficient query processing relies on the concept of *repositories*. *Repositories* are sensors that store readings of a *set* of clusters, and are the only ones that have to be contacted for answering queries. AQPM has been implemented on NS2 simulator and experimental results show that it is more effective than existing approaches.**

## I. Introduction

Wireless sensor networks (WSNs) have provided an infrastructure for leveraging a wide range of applications to various fields of interest, such as environmental monitoring, military surveillance and health care [1]. Sensor nodes usually have limited capabilities including low bandwidth, and limited storage, energy and processing power. In the context of urban monitoring, due to the city's irregular topography, variations of pollutant concentration differ even on opposite sides of the street [2]. Thus, monitoring such complex environments requires deployment of dense WSNs [3] [4]. Moreover, sensor devices may store the monitored data enabling applications to execute queries in the network, without relying on a central server. In particular, for large-scale WSNs, keeping the whole data centralized for further processing increase communication costs [5] and is less scalable than decentralized approaches [6].

Query dissemination is a hard task due to high communication and energy costs. However, they can be reduced by exploring some characteristics found in WSNs, such as the spatial and temporal correlations among readings [7]–[9]. Spatial correlation refers to the similarity among data collected by nearby devices, whereas temporal correlation refers to the similarity among consecutive readings from each sensor node [10]. Given that in some applications readings among nearby sensors show redundant data, scalability can be reached by arranging in groups sensors with similar readings [11].

In addition to explore spatio-temporal similarity, a suitable data storage strategy can be a key feature for reducing energy consumption and minimizing query processing delay [12]. In general, sensor readings can be locally stored in the sensor itself, in an external database or in distributed *repositories* within the network. Although the best strategy depends on the application context it has been noticed that the repository approach offers an interesting tradeoff between query and storage costs [13]. In such approach, communication costs depends on *repositories* placement, aiming at minimizing update and query processing delays.

Inspired by these challenges some works have emerged in order to address the query processing task. The majority of current clustering algorithms, such as SIDS [14], consider only one-dimension sensor reading, such as temperature or humidity. Among those that consider readings similarity of multidimensional data, we can cite DCSSC [7] and DCASC [15]. They all group sensors into clusters with a designated sensor for storing the group members' readings, denoted as cluster-head (CH). A node that plays the role of CH is responsible for storing the relevant data about the entire cluster. DCSSC and DCASC rely only on CHs for answering query requests. That is, they do not support the concept of a *repository* to reduce the number of cluster-heads to be contacted during query dissemination, and thus query processing may be costly. Another limitation of DCSSC is that queries are issued only from one specific external entry point. SIDS does support the concept of *repositories*. Nevertheless, it is not an autonomous model, relying on an external entity to coordinate the sensors clustering process.

In this paper we propose the Autonomic Query Processing Model for Sensor Networks (AQPM). The system works as follows. At the lowest level, each sensor node keeps locally the gathered data. In addition, nodes with high correlation are organized in clusters. At the *cluster* level, a node named as cluster-head (CH) is elected to be the group representative. At the *repository* level, specific nodes serving as data *repository* store data transmitted from the cluster level. The goal of the *repository* is to reduce the number of CHs contacted during query processing by concentrating information of clusters with intersecting boundaries in a single location. Our model 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 autonomic model that combines spatial similarity of readings with *repository* placement for reducing the network communication during the query processing in the context of urban WSNs. The system evaluation was conducted by simulations taking into account

an urban scenario. Results shows that AQPM can effectively support autonomic query processing, as well as reducing the communication overhead.

The remainder of this paper is organized as follows. Section II presents related work. Section III describes our proposal and Section IV reports the performance evaluation. We conclude in Section V highlighting future works.

## II. Related Work

Recent research efforts have been made to deliver efficient in-network query processing in WSNs. Data storage and clustering are common approaches used in the development of efficient query processing applications. Scoop [16] and GHT [17] are systems that follow repository-based approaches for storing sensing data. Scoop maps ranges of collected data to sensor devices. However, unlike our work, Scoop is not scalable and requires a base station to operate the network; that is, it is not an autonomous system. GHT, on the other hand, uses a hashing function to map attribute names to a geographical location. Queries and updates in GHT may need to contact storage sensors that may be located far from the collecting devices; thus, these operations can be costly. Additionally, none of them support both spatial and value-based queries. Models that tackle the scalability problem with similarity-based clustering techniques include CAG [18], DCSSC [7] and SIDS [14]. However, none of them is fully-autonomous.

IBIS [19] proposes an efficient mechanism for processing spatial queries that reduces the energy consumption based on finding itineraries for irregular regions of interest. It creates a route for forwarding the request among sensors and for determining where the sensed data should be aggregated. However, IBIS does not take into consideration data similarity and clustering.

To the best of our knowledge none of the existing sensor models provides an in-network query processing model that meets the requirements found in urban WSNs. This model should be able to combine data similarity, node clustering strategies and data *repositories*, in order to support a dense and large-scale WSN autonomously.

## III. The AQPM Model

We introduce AQPM (*Autonomous Query Processing Model*), a hierarchical and distributed in-network model for efficiently answering spatial and value-based queries. AQPM organizes sensors in clusters by exploiting the spatial data similarity of their readings. Once clusters are defined, some sensors selected as data *repositories* store information of a set of distinct clusters. Intuitively, given that cluster heads represent sensing data for every cluster member, a *repository* can act as a datacenter to answer queries referring to any of the clusters that compose it. In the following sections, we present the network model, the concept of spatial correlation, and the query processing algorithm.

### A. The network Model

A wireless sensor network (WSN) is represented as a graph $G = (V, L)$, where $V = \{s_1, \ldots, s_n\}$ is a set of sensors spread over a monitored area 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 $\{(s_1, s_2), (s_2, s_3), \ldots (s_{n-1}, s_n)\}$ 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. In our model, we assume that sensors are static and aware about their location, and thus have a fixed geographic coordinate (*position*$(s_i)$). We can thus rely on a geographic *routing* protocol for determining paths that establish communication between sensors. We also assume that each sensor is responsible for monitoring a multidimensional measurement from the environment. We denote the current readings of a sensor $s$ as a tuple $X = (x_1, x_2, \ldots, x_n)$, where each $x_i$ corresponds to one type of reading such as temperature, humidity, luminosity and air pollution.

As an example, consider a WSN deployed over an urban area collecting several environmental features. Figure 1(a) shows sensors densely placed over regions of the city, such as parks, gardens, streets and avenues. In AQPM, spatially close sensors are grouped into clusters based on measuring their spatial correlation, as shown in Figure 1(b). Every cluster has one sensor that serves as cluster-head (CH). Moreover, clusters are defined on contiguous spatial areas. Consequently, data sampled within each cluster have very high spatial correlation among its members. Thus, based on these observations, in our model we consider that only CH information is relevant for answering queries, instead of collecting and aggregating at the CH data sensed by cluster members. Thus, AQPM clustering approach reduces the intracluster communication overhead, saving network resources.

The developed CH selection is inspired by the spatial correlation clustering algorithm proposed in [15]. Let $X = (x_1, x_2, \ldots, x_n)$ be the readings of sensor $s_i$, $Y = (y_1, y_2, \ldots, y_n)$ the readings of sensor $s_j$ and $N(i)$ the set of neighbors within one-hop of sensor $s_i$. The algorithm uses a weight-based clustering approach, which is divided in four steps. In the first step, the Euclidean distance between two sensor measurements, $d_{ij}$, is computed as follows.

$$d_{ij} = \sqrt{|x_1 - y_1|^2 + |x_2 - y_2|^2 + \ldots + |x_n - y_n|^2} \tag{1}$$

Based on all neighbors of a sensor $s_i$, its mean value $\overline{d_i}$ and average absolute deviation $D(d_{ij})$ are computed by the following equations.

$$\overline{d_i} = \frac{1}{|N(i)|} \sum_{j \in N(i)} d_{ij} \tag{2}$$

$$D(d_{ij}) = \frac{1}{|N(i)|} \sum_{j \in N(i)} (d_{ij} - \overline{d_i})^2 \tag{3}$$

Then in the last step, the spatial correlated weight $w(s_i)$ $(0 \leq w(s_i) \leq 1)$ of node $s_i$ is given by

$$w(s_i) = \frac{\left[\sum_{j \in N(i)} |d_{ij} - \overline{d_i}|\right]^2}{|N(i)|^2 D(d_{ij})} = \frac{\left[\sum_{j \in N(i)} |d_{ij} - \overline{d_i}|\right]^2}{|N(i)| \sum_{j \in N(i)} (d_{ij} - \overline{d_i})^2} \tag{4}$$

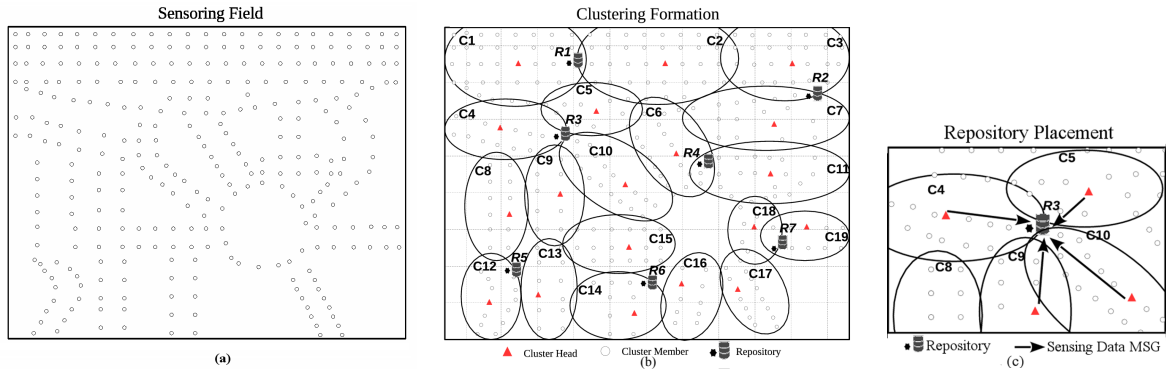



Fig. 1. An overview of AQPM scenario, proposed model and data storage model.

According to the equation above, each sensor calculates its weight $w(s_i)$ that determines the degree of correlation among measurements of $s_i$ and its neighbors $N(i)$. A value close to 1 indicates that readings of $s_i$ and nearby sensors are highly correlated. Indeed, in our model, a sensor is selected as *CH* if its correlated weight is above a user-defined threshold.

AQPM is a three-tier hierarchical model that consists of a sensor-level $s$, a cluster-level $h$ to which $s$ belongs to ($CH(s)$) and a repository level that stores information on $h$ ($R(h)$). Observe that the same sensor can assume different roles simultaneously. That is, in addition to the role of a cluster member, it can be a cluster head and/or a *repository*. We say that a sensor is an isolated cluster head if it is a CH with only itself as a member. We now briefly describe the proposed AQPM's algorithms for clustering and repository placement.

In the clustering algorithm, each sensor $s$ requests its neighbors to send their readings in order to calculate its spatial correlated weight ($w(s)$). Then, it announces itself as leader ($CH$) if $w(s)$ is above the user-defined *threshold* and broadcasts the decision to its neighbors. Nodes with low weight maintain a set structure *candidateCHs* for storing each announced *leader*. Later, these nodes (called *members*) join the cluster with the most similar readings among all $CHs$ stored in *candidateCHs* and sends an ACK message to the chosen $CH$ with its coordinate. Thus, each $CH$ maintains the geographical position of its cluster members, and can compute its minimum bounding rectangle (MBR). If a $CH$ does not receive any ACK, it updates its role as an *isolated cluster head*. To illustrate, consider scenario in Figure 1(b). Each cluster has one leader and sensors become a member of the leader with highest spatially correlated readings.

Given that sensors have been grouped based on their spatial data similarity, we are now ready to define our *repository* placement algorithm. A *repository* aims at minimizing the number of hops during query processing by storing information of nearby CHs, such that only *repositories* may have to be contacted in order to answer queries. Thus, we select as *repositories* sensors in a border region that can concentrate the maximum number of clusters in their neighborhood. The algorithm works as follows. First, a sensor broadcasts a *repository-announcement* if the number of neighbor nodes in distinct clusters exceeds an user-defined threshold. Sensors that play the role of cluster-heads or repositories that receive an announcement message maintain a set structure *knownRepo* for storing each announced *repository* and re-broadcast this message if the distance to the new repository is smaller than the distance to all others. It guarantees a stop criteria for the repository-announcement broadcast. Later, each $CH$ joins the closest repository candidate and sends an ACK informing its readings and cluster members positions. Repositories store these information and calculate their MBR, which is then send to its neighbor repositories recorded in *knownRepo*. At the end of this process, each repository contains information of its neighbor repositories and their MBR, besides the cluster readings. As an example, consider the scenario in Figure 1(c). *Repository* $R3$ is in a shared border region, composed of clusters $C4$, $C5$, $C9$ and $C10$.

### B. Query Processing

AQPM in-networking query processing mechanism supports both spatial and value-based queries, which can be injected at any node in the WSN, without relying on a single entry point. A single query parameter defining the *range* of the query is given as input. For *value-based* queries, the range is an interval, and the result consists of the sensors in the monitored area with values within the range. For *spatial* queries, the range is a rectangular region, and the result consists of the readings of sensors located in the area of interest. Recall that AQPM considers that CHs readings are representative of readings of all sensors in a cluster and that *repositories* serve as data storage for nearby CHs. Thus, processing queries consist of finding within the *repositories* the set of sensors and their associated information that match the query criteria and then return the information to the entry point. The routing protocol used to forward query messages and retrieve query answers relies on the GPSR protocol [20]. Algorithm 1 details the query processing strategy.

The *query processing* algorithm explores AQPM's hierarchical model. First, each query is assigned a unique identifier *qryId*. The algorithm collects the results sent from different repositories in a variable *qryRes[qryId] (l.1-3)*. An entry point sensor receives a query and forwards spatial queries to the region of interest, and value-based queries to its $CH$ *(l.5-7)*. Spatial queries follows AQPM hierarchy only after the first sensor in the region is reached *(l.16-24)*. In the region, the query is forwarded to a cluster-head, and then to its corresponding repository *(l.19-24)*. Procedure *RECEIVING('COLLECT')*) is executed by every repository in the query range for collecting readings. First, data from clusters

**Algorithm 1** Query Processing

1: **procedure** SEARCH($range, type$) by $s_i$
2: $qryId \leftarrow new(id)$
3: $qryRes[qryId] \leftarrow \{\}$
4: **if** $type =$ SPATIAL **then**
5: send ('SPATIAL', $qryId, s_i, range, \{\}$) in *range* direction
6: **else**
7: send ('VALUE', $qryId, s_i, range, \{\}$) to $CH(s_i)$
8: WAIT($\Omega$ time units) for response
9: output $qryRes[qryId]$
10: **procedure** RECEIVING('RESULT', $qryId$, $entryP$, $res$) by $s_i$
11: **if** $s_i = entryP$ **then**
12: $qryRes[qryId] \leftarrow qryRes[qryId] \cup res$
13: **else**
14: forward ('RESULT', $qryId$, $entryP$, $res$) to $entryP$
15: **procedure** RECEIVING('SPATIAL', $qryId$, $entryP$, $range$, $res$) by $s_i$
16: **if** *position*($s_i$) not in *range* **then**
17: forward ('SPATIAL', $qryId$, $entryP$, $range$, $res$, *reached*) in *range* direction
18: **else**
19: **if** $s_i$ is a repository **then**
20: execute ('COLLECT', $qryId$, $entryP$, $range$, $res$)
21: **else if** $s_i$ is a cluster-head **then**
22: forward ('SPATIAL', $qryId$, $entryP$, $range$, $res$) to $R(s_i)$
23: **else**
24: forward ('SPATIAL', $qryId$, $entryP$, $range$, $res$) to $CH(s_i)$
25: **procedure** RECEIVING('COLLECT', *qryId, entryP, range, res*) by $Rep$
26: **if** $Rep$ has already processed $qryId$ **then**
27: drop message; return;
28: $res \leftarrow$ GetDataThatOverlapsRange()
29: send ('RESULT', $qryId$, $entryP$, $res$) to $entryP$
30: **for all** repository $r$ in $knownRepo(Rep)$ **do**
31: **if** $MBR(r)$ overlaps $range$ **then**
32: send ('COLLECT', $qryId$, $entryP$, $range$, $res$) to $r$
33: **procedure** RECEIVING('VALUE', $qryId$, $entryP$, $range$, $res$) by $s_i$
34: **if** $s_i$ has already processed $qryId$ **then**
35: drop message; return;
36: **if** $s_i$ is a repository **then**
37: $res \leftarrow$ GetDataThatIntersectRange()
38: send ('RESULT', $qryId$, $entryP$, $res$) to $entryP$
39: send ('VALUE', *qryId, entryP, range, res*) to *knownRepo*($s_i$)
40: **else if** $s_i$ is a cluster-head **then**
41: forward ('VALUE', *qryId, entryP, range, res*) to $R(s_i)$
42: **else**
43: forward ('VALUE', *qryId, entryP, range, res*) to $CH(s_i)$

in the repository that overlaps the query region are obtained by function *GetDataThatOverlapsRange*, and sent back to the entry point in a *RESULT* message *(l.28-29)*. Then, a *COLLECT* message is sent to neighbor repositories which also overlaps the query region *(l.30-32)*. Regarding value-based queries (procedure *RECEIVING('VALUE')*) *(l.33)*, query results are gathered from repositories by checking whether the stored values intersect the query range, which are then sent back to the entryPoint *(l.36-38)*. As opposed to spatial queries, value-based queries should be forwarded to all repositories in order to collect the query result *(l.39)*. The forwarding strategy follows the AQPM hierarchy model *(l.40-43)*.

As an example of a spatial query, consider the scenario in Figure 2, where *repositories* $R_3$ and $R_4$ are composed of cluster $\{C_4, C_5, C_9, C_{10}\}$, and $\{C_6, C_7, C_{11}\}$, respectively. The region of interest consists of the rectangle $Q$ around repository $R_4$, which overlaps clusters $C_6, C_7, C_{10}$ and $C_{11}$. Suppose the query is injected in the WSN at a sensor $s_e$ in $C_8$. The query execution starts by forwarding the message from the entry point to the query region $Q$. Suppose the first sensor in $Q$ to be reached is a sensor $s_q$ in $C_{10}$. After reaching $s_q$, the

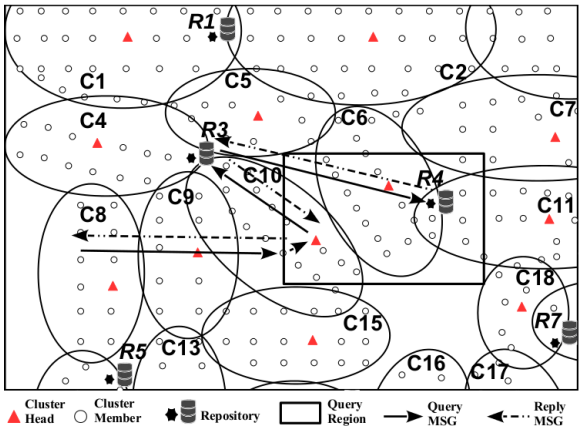



Fig. 2. Repository placement and query processing

message is forward to its CH, and then to its repository $R_3$. Values from cluster $C_{10}$ are collected from $R_3$ and sent back to the entry point $s_e$. Among $R_3$'s neighbor repositories only $R_4$ overlaps $Q$. Thus the *COLLECT* message is sent to $R_4$, from which the values from the remaining clusters that overlap $Q$ are collected and sent back to $s_e$.

## IV. EVALUATION

In this section, we present the performance evaluation of AQPM. We have conducted simulations in order to validate the model and for determining its query efficiency. We consider the following metrics: processing time, energy consumption and error rate. In order to determine AQPM efficiency, we have considered the effect of our similarity-based clustering strategy on determining the number of CHs and repositories. We have also conducted simulations to compare the cost of processing spatial queries on AQPM and IBIS [19]. Both systems have been implemented on NS2 network simulator version 2.34.

### A. Simulation Settings

We have generated synthetic scenarios in which 140 sensors were statically placed on a $1400 \times 1000$ square meter monitored area. The distance between sensor nodes were around 90 meters, with symmetric links, and MAC protocol (802.11). The radio range of every sensor on the field was set to 100 meters. Nodes were equipped with GPS devices allowing them to be aware of their geographical position over the monitored area. The simulation duration time is 40 minutes. For running the simulation, first the algorithms for clustering and repository selection were executed. Then, a query were injected at node 28 to get the average value of five measurements types collected by sensors located at a rectangle delimited by $v_1(300, 50)$ upper-left corner and $v_2(950, 500)$ bottom-right corner, as illustrated in Figure 3. All the results presented in this section correspond to the average of 35 simulations, with a confidence interval of 95%.

We have conducted two experiments. In the first, we analyzed the AQPM model by determining the number of repositories, cluster heads and isolated cluster heads generated by our clustering and repository selection algorithms. For each simulation, we have generated new readings for the monitored area. In the second experiment, AQPM was compared to IBIS. We considered three cost metrics: energy consumption (Joules),

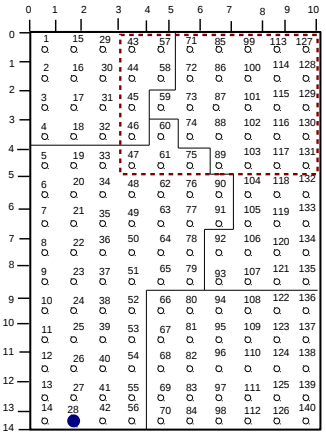


Fig. 3. Simulation scenario

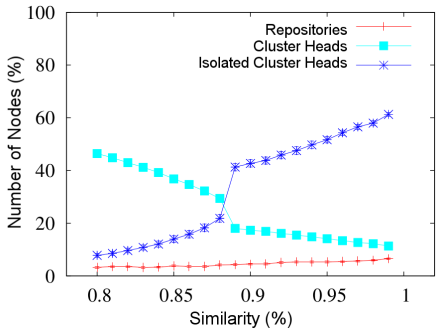



Fig. 4. Variations in the degree of correlation

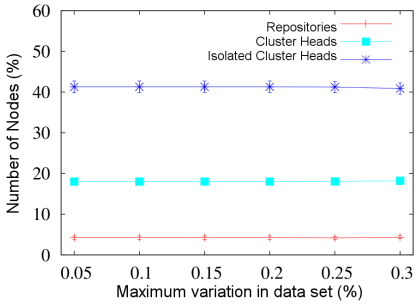



Fig. 5. Maximum variation of data readings

processing time (seconds) and query results error rate. The energy consumed by the network to transmit a packet ($E_{PT}$) consists of the energy for transmitting a packet ($E_{TX}$), plus the sum of the energy consumed in the reception by each of the $n$ neighbors ($\sum_{x=1}^{n} E_{RE}$). This consumption can be modeled by the equation: $E_{(i \to j)PT} = E_{(i)TR} + \sum_{x=1}^{n} E_{(x)RE}$.

AQPM proposes a storage model and query mechanism for applications in which sensor readings have spatial similarity. Thus, we generated the sensors' readings as follows. The monitored area was divided in four asymmetric regions, as illustrated in Figure 3. Each sensor collected data of five distinct measurement types, such as temperature and humidity. We considered that all readings were within $[0, 10]$ interval. Initially, four data sets were created, each set consisting of $5$ values generated randomly. Each set were associated with one region and their values correspond to their value seeds. That is, the sensor readings in the same region are values with random variation of no more than a percentage $V$ from the seed value. Thus, the maximum difference between the readings inside the same region were $V\%$.

### B. *Node Clustering and Data* Repositories

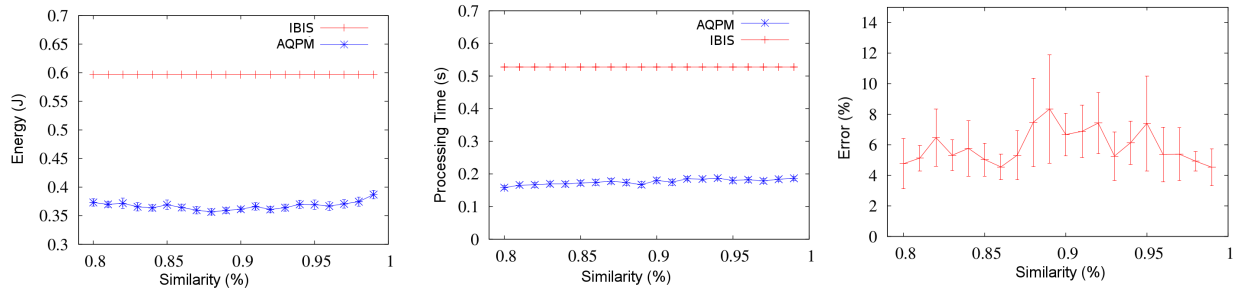
In this experiment, the goal was to determine the number of repositories, cluster-heads, and isolated cluster-heads generated by AQPM. As shown in Figure 4, the similarity threshold used to choose cluster heads ranged from $0.8$ to $0.99$, $V$ was set to $10\%$ and the threshold to select *repositories* in all simulations were set to $4$.

The experiments showed that the percentage of *repositories* remained stable in all settings. The *repository* selection takes into account the proximity of nearby CHs. Therefore, the variation of similarity for choosing CHs, as shown in Figure 4, has low impact in the number of *repositories* in the network. This result shows that the AQPM's strategy of concentrating readings in *repositories* rather than keeping them only in the CHs indeed meets its purpose. Since only repositories have to be contacted for answering queries, this strategy potentially leads to low communication costs for query processing, even on large monitored areas. As shown in the graph, by increasing the similarity threshold for choosing CHs, the number of isolated CHs grows, and the number of CHs is reduced. This result were as expected, since the variance $V$ within nodes in the same quadrant were randomly chosen and thus the number of neighbor nodes with higher similarity than the threshold is lower, generating more isolated CHs. In real scenarios, however, with higher similarity among neighboring sensors, the number of isolated CHs is most probably lower. Figure 5 shows that the percentage $V$ for generating the data set does not have a big impact on the number of clusters and *repositories*. In these experiments, the threshold for choosing CHs was set to $0.88$.

### C. *Query Processing*

This simulation has been conducted in order to compare the cost of processing spatial queries on AQPM and on IBIS. IBIS defines an itinerary within the query's area of interest so that all sensors contained in this region are neighbors of the itinerary. It is worth noticing that IBIS modifies its query execution plan only when there are changes in the network topology. Since in our experiments the network topology remains unchanged, IBIS behavior was constant in all scenarios. We have chosen to compare AQPM with IBIS because among the existing works for processing spatial queries we have analyzed, IBIS is the one that reported the best performance.

As illustrated in Figure 6(a) and Figure 6(b), AQPM showed better performance than IBIS with respect to energy consumption and response time for processing a query request. In these experiments we considered $V = 10\%$. AQPM low energy consumption results from the fact that requests are disseminated only to *repositories* containing sensors that overlap the query region. Moreover, AQPM shows better performance regarding response time, due to the fact that IBIS creates multiple $delays$ during query dissemination, which does not occur with AQPM. However, AQPM does not return exact results, since it considers that the CH readings are representative for the entire cluster. IBIS, on the other hand, gets the readings directly from the sensor devices of the region of interest. The difference between the exact average value of the readings and the value output by AQPM is presented in Figure 6(c). The mean error is around $6\%$ in all scenarios. On the other hand the trade-off between energy versus accuracy is known in the literature [21]. It is worth noticing that this error would be much lower if we had considered a single measurement type, instead of five. This is because for determining the degree of similarity among cluster members, all five measurements are




(a) Energy consumption for query processing (b) Response time for query processing (c) Error rate for query processing

Fig. 6. Cost of Query Processing between AQPM and IBIS

considered. The similarity accuracy is much higher with small number of measurement types.

In addition to the presented metrics, it is important to analyze the cost of node clustering and *repositories* selection. In the experiments with AQPM, this cost was around 0.35 Joules. IBIS energy consumption is lower than AQPM, about 0.054 Joules, for flooding the network. However, this difference is compensated for processing further requests, since AQPM consumes less energy to process queries.

All results show that AQPM meets the requirements of **scalability** and **autonomy** to support the storage of data sensing on urban WSNs, as well as an effective query processing mechanism. In the future, we plan to to analyze and extend the model with alternative approaches for minimizing the relative query result error.

## V. Conclusion

This paper proposed AQPM, an autonomic and scalable storage model, for supporting query processing of sensing data on urban WSNs. AQPM has been inspired on characteristics usually found in urban sensing applications, such as correlated spatial data and readings that change gradually over time. Moreover, AQPM focuses on improving the query processing by storing sensing data on data repositories in the network. We have run experiments, based on simulations, showing that our data placement strategy can significantly reduce the number of messages required for query processing.

In the future, we intend to analyze the impact of the dynamics aspects of sensor readings on the AQPM data model maintenance. The expected functionality is an autonomous reclustering mechanism that can be executed whenever readings updates exceed cluster value limits.

**Acknowledgments.** This work was partially supported by the following Brazilian Agencies: CNPq, CAPES, and Fundação Araucária.

## References

[1] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer networks*, vol. 52, no. 12, pp. 2292–2330, 2008.

[2] B. Resch, M. Mittlboeck, F. Girardin, R. Britter, and C. Ratti, "Live geography–embedded sensing for standarised urban environmental monitoring," 2009.

[3] C. L. Muller, L. Chapman, C. Grimmond, D. T. Young, and X. Cai, "Sensors and the city: a review of urban meteorological networks," *International Journal of Climatology*, vol. 33, no. 7, pp. 1585–1600, 2013.

[4] N. Thepvilojanapong, T. Ono, and Y. Tobe, "A deployment of fine-grained sensor network and empirical analysis of urban temperature," *Sensors*, vol. 10, no. 3, pp. 2217–2241, 2010.

[5] Z. Can and M. Demirbas, "A survey on in-network querying and tracking services for wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 1, pp. 596–610, 2013.

[6] A. Coman, J. Sander, and M. A. Nascimento, "Adaptive processing of historical spatial range queries in peer-to-peer sensor networks," *Distributed and Parallel Databases*, vol. 22, no. 2-3, pp. 133–163, 2007.

[7] T. D. Le, N. D. Pham, and H. Choo, "Towards a distributed clustering scheme based on spatial correlation in wsns," in *Wireless Communications and Mobile Computing Conference, 2008. IWCMC'08. International*. IEEE, 2008, pp. 529–534.

[8] B. Cheng, Z. Xu, C. Chen, and X. Guan, "Spatial correlated data collection in wireless sensor networks with multiple sinks," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*. IEEE, 2011, pp. 578–583.

[9] F. Gielow, G. Jakllari, M. Nogueira, and A. Santos, "Data similarity aware dynamic node clustering in wireless sensor networks," *Ad Hoc Networks*, vol. 24, pp. 29–45, 2015.

[10] M. C. Vuran, Ö. B. Akan, and I. F. Akyildiz, "Spatio-temporal correlation: theory and applications for wireless sensor networks," *Computer Networks*, vol. 45, no. 3, pp. 245–259, 2004.

[11] C.-C. Hung, W.-C. Peng, and W.-C. Lee, "Energy-aware set-covering approaches for approximate data collection in wireless sensor networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 11, pp. 1993–2007, 2012.

[12] Z. Yu, B. Xiao, and S. Zhou, "Achieving optimal data storage position in wireless sensor networks," *Computer Communications*, vol. 33, no. 1, pp. 92–102, 2010.

[13] L. Xie, S. Lu, Y. Cao, and D. Chen, "Towards energy-efficient storage placement in large scale sensor networks," *Frontiers of Computer Science*, vol. 8, no. 3, pp. 409–425, 2014.

[14] S. S. Furlaneto, A. Dos Santos, and C. S. Hara, "An efficient data acquisition model for urban sensor networks," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*. IEEE, 2012, pp. 113–120.

[15] Y. Ma, Y. Guo, X. Tian, and M. Ghanem, "Distributed clustering-based aggregation algorithm for spatial correlated sensor networks," *Sensors Journal, IEEE*, vol. 11, no. 3, pp. 641–648, 2011.

[16] T. M. Gil and S. Madden, "Scoop: An adaptive indexing scheme for stored data in sensor networks," in *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, R. Chirkova, A. Dogac, M. T. Özsu, and T. K. Sellis, Eds. IEEE, 2007, pp. 1345–1349.

[17] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-centric storage in sensornets," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 137–142, 2003.

[18] S. Yoon and C. Shahabi, "The clustered aggregation (cag) technique leveraging spatial and temporal correlations in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, p. 3, 2007.

[19] R. I. da Silva, D. F. Macedo, and J. M. S. Nogueira, "Contornos irregulares no processamento de requisições espaciais para redes de sensores sem fio," in *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2011.

[20] B. Karp and H.-T. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," in *Proceedings of the 6th annual international conference on Mobile computing and networking*. ACM, 2000, pp. 243–254.

[21] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: an energy–accuracy trade-off," *Ad hoc networks*, vol. 1, no. 2, pp. 317–331, 2003.