On the Use of Histograms for Processing Exact Aggregate Queries in Wireless Sensor Networks

Khaled Ammar Mario A. Nascimento
Department of Computing Science,
University of Alberta, Canada
{ kammar, mn }@cs.ualberta.ca

ABSTRACT

Wireless Sensor Networks (WSNs) are typically used to collect values of some phenomena in a monitored area. In many applications, users are interested in statistical summaries of the observed data, e.g., histograms reflecting the distribution of the collected values. In this paper we discuss how to leverage on a histogram query processing algorithm in order to process other aggregate queries within a WSN exactly. Our experimental results using both synthetic and real data sets show that our proposed solutions are able to substantially decrease the overall amount of bytes sent, thus saving energy and increasing the WNS lifespan, when compared to other classical approaches.

1. INTRODUCTION

A typical Wireless Sensor Network (WSN) consists of nodes, equipped with sensors, distributed in an area and connected, via a tree-like topology, to a base station. WSNs are typically used to observe some phenomena about a monitored area and are becoming common in many application domains [4, 5, 11]. Typically, WSN nodes have limited resources in terms of energy, CPU and memory. The base station, on the other hand, is a full-fledged computer system with light limitations on memory, CPU, or bandwidth. Battery lifetime is considered the most important resource in WSN nodes as the required energy for transmission is significantly higher than the required energy for data processing in a node [7]. Thus, it is very important that query processing algorithms for WSN are energy-efficient.

We assume that a WSN has N nodes $s_j \in S$ $(1 \leq j \leq N)$ spread in a monitored area. Each node s_j in S periodically measures a value v_j. In fact, every measured value has an associated timestamp, however in order to lighten the notation we do not denote it unless necessary. Nodes are connected to the base station by a routing tree, where the base station is the root and, they can reach the base station by multi-hop routing through other nodes. Typically, this tree is constructed to minimize the number of hops between any node and the base station. We assume that the connections between nodes are reliable (no link failure), and we focus on the data aggregation problem only. Finally, users are connected to the WSN through the base station where they can submit queries and collect the respective answers.

A *Histogram* query is defined as: $Q(Lb, Ub, b_1, b_2, ..., b_B, \tau)$, where τ is the time lapse, or epoch, between any two consecutive *Histogram* answers. The lower and upper bound values of the measured phenomena are Lb and Ub. Each b_i is one of B bins in the *Histogram* query and it is defined as $b_i = [Lb_i, Ub_i)$ for , where $1 \leq i < B$ and $b_B = [Lb_B, Ub_B]$, furthermore, $Ub_i \leq Lb_j \ \forall i < j$ and $\bigcup_{1 \leq i \leq B}\{b_i\} = [Lb, Ub]$ and $Lb_1 = Lb$ and $Ub_B = Ub$. The answer for a *Histogram* query is $H = (h_1, h_2, ..., h_B)$, where $h_i = \mid \{(s_j, v_j) \mid Lb_i \leq v_j < Ub_i,\ s_j \in S\} \mid$.

This paper extends our previously proposed algorithm for continuous *Histogram* query HIU [1]. HIU is an efficient distributed algorithm to answer continuous *Histogram* queries in WSNs and it requires less than the half the amount of energy used by the classical TAG algorithm [6] to construct a histogram. Our contribution in this paper is a set of algorithms that use the *answer* of a *Histogram* query, at the end of the each round, to answer other aggregate queries. The proposed approaches can find approximate as well as exact answers. Approximate answers can be obtained with no overhead, whereas exact answers require a very small overhead on the WSN.

The rest of this paper is organized as follows: Section 2 introduces the HIU algorithm and shows how to compute approximate answers for several aggregate queries. Next, we show how to compute the exact answers for aggregate queries using a *Histogram* computed by HIU as a starting point. Section 3 presents our experiments and Section 4 discusses briefly the related work. Finally, Section 5 concludes the paper and presents a few directions for further research.

2. HISTOGRAM BASED AGGREGATE QUERY PROCESSING

A *Histogram* provides a broad picture for values in the WSN and is a good starting point for more statistical analysis. Before presenting how to obtain exact aggregate queries using a previously obtained *Histogram*, we discuss briefly how to compute approximate answers for those queries. These approximate solutions have bounded accuracy levels and are computed with no overhead on the histogram query.

In [1], we proposed the HIU algorithm to compute the answer of a *Histogram* query as defined above. In HIU, each intermediate node constructs its local histogram answer based

Query	Approximate answer	Error bound
Max	$\frac{Ub_m+Lb_m}{2} \mid h_m \neq 0, \forall i > m, h_i = 0$	$\frac{Ub_m-Lb_m}{2}$
Min	$\frac{Ub_m+Lb_m}{2} \mid h_m \neq 0, \forall i < m, h_i = 0$	
Median	$\frac{Ub_m+Lb_m}{2} \mid \Sigma_{i=1}^{m}\{h_i\} \geq \frac{Count}{2}, \Sigma_{i=m}^{B}\{h_i\} \geq \frac{Count}{2}$	
Count	$\Sigma_{i=1}^{B}\{h_i\}$	0
Sum	$\Sigma_{i=1}^{B}\{h_i \times \frac{Ub_i+Lb_i}{2}\}$	$\Sigma_{i=1}^{B}\{\frac{Ub_i-Lb_i}{2} \times h_i\}$
Average	$\frac{Sum}{Count}$	$\frac{1}{Count}\Sigma_{i=1}^{B}\{\frac{Ub_i-Lb_i}{2} \times h_i\}$

Table 1: Approximations for aggregate queries using HUI's response

on its subtree and stores it as a partial state. In the following round, an intermediate node compares the cached and newly constructed partial states, and builds the most efficient message to communicate this change to its parent. A leaf node communicates its value if and only if its value is changed and if this change can influence the local histogram in its parent.

Table 1 shows how to compute approximate answers for some aggregate queries in the base station using HIU's response to a *Histogram* query. Because all computations are made on the base station, there is absolutely no overhead on the WSN. This table also shows the error bound for each approximate aggregate answer. All bounds depend on the bin size ($Ub_i - Lb_i$). Decreasing the bin size in the *Histogram* query will lead to answers with higher accuracy. However, this will increase the overall cost of the histogram result because it increases the number of sent bytes.

The HIU algorithm can also be extended to obtain exact answers for several aggregate queries. In each round, we can obtain exact answers using two strategies: (1) adding some per-message overhead but *no extra messages*, or (2) sending refinement queries, thus extra messages, but imposing *no per-message overhead*. We discuss both strategies next. However, it is important to note that obtaining the aggregate value is a post-processing "overhead" at the end of each one of HIU's round. After the aggregate value is obtained, this post-processing effort is "forgotten" and the next round's histogram can be computed by HIU, in a continuous manner, as detailed in [1].

2.1 Exact answers using per-message overhead

Communication devices in some WSN mandate the sensor to send messages of fixed size only [8]. In this case, sending less information will not decrease the energy consumption because all buckets should have the same size. These idle bytes can be used to send some extra information, at no extra cost, to facilitate computing the exact answer. We can use this strategy to compute exact answers for *Max*, *Min*, *Sum*, and *Average* queries.

While processing a *Histogram* query, a node can send more information attached to its message. The exact type of information depends on the required aggregate query. In case of *Max* (or *Min*) queries, all intermediate nodes who construct a partial status (for the *Histogram* query) should report information about the maximum (or minimum) value in their subtrees.

Because the base station (and all intermediate nodes) already has an exact answer for *Count*, it can compute the exact result for *Average* if the exact *Sum* is available. The exact answer for *Sum* can be computed if each intermediate node sends the total sum of its subtree while leaf nodes send their own values.

If the communication device allows variable message size, then every bit counts when calculating the energy consumption. We can decrease the size of the overhead using the information we havefrom the histogram. For instance, instead of reporting the real value of the Maximum (Minimum), nodes select a bin id (m) that includes the maximum value and send the difference between the bin's lower bound (LB_m) and the *Max* value. In the worst case, the overhead will be number of bits required to represent the bin size ($Ub_m - Lb_m$) which is $log_2(Ub_m - Lb_m)$ bits per node, every epoch τ. Following the same idea, instead of sending the real sum of all values in a node's subtree which might need a large representation, each the node will send the difference $Sum - \Sigma_{i=1}^{B}\frac{Ub_i+Lb_i}{2} \times h_i$. The maximum possible value of *Sum* in any node is $\Sigma_{i=1}^{B}Ub_i \times h_i$, if all values in all bins are equal to the upper bound of this bin. The maximum possible overhead per node per round is $log_2(\Sigma_{i=1}^{B}\frac{Ub_i-Lb_i}{2} \times h_i)$.

2.2 Exact answers using extra messages

Some queries cannot be answered by sending a single value attached to the *Histogram* 's query answer as describe above. For instance, *Median* and K^{th}*value* queries require sending all values to the base station [6]. For the sake of simplicity we focus on *Median* queries in the remainder of this section, but we observe that the proposed ideas can be easily generalized to process K^{th}*value* queries; indeed, the *Median* queries is a special case of the former where $K = \lceil N/2 \rceil$.

Using a histogram answer, we can narrow the requirements from collecting all values in the WSN to collecting all values in the median bin. A bin that contains the median value can be easily identified based on the histogram answer in the base station. It is the bin where the total number of values in all preceding and all following bins in the histogram are both less than half the number of values in the whole histogram. That is, it is the bin b_m, where $m = arg\{b_m \mid \Sigma_{i=1}^{m}\{h_i\} > \frac{Count}{2}$, and $\Sigma_{i=m}^{B}\{h_i\} > \frac{Count}{2}\}$. Instead of reporting the average of the median bin as an approximate answer, the base station may send another query to the WSN requesting all values in this bin. The query will be guided only to the relevant nodes by using the cached partial states. The full algorithm is presented in Algorithm 1. It has two parameters: the WSN's logical routing tree (T) and the histogram answer (H) in the base station –obtained using the HIU algorithm for instance– and it returns the exact median.

Function GetVALUES returns all values in the median bin as an array. After sorting the returned values and using the number of total values in the WSN, the ExactMEDIAN algorithm computes and returns the exact *Median*. The algorithm for function GetVALUES is represented in Algorithm 2. It collects all values between values L and U in the given tree T. However, instead of sending the request to all

Algorithm 1 ExactMEDIAN(Logical routing tree T, histogram answer H)

1: $Count \leftarrow \Sigma_{i=1}^{B}\{h_i\}, \forall h_i \in H$ {Histogram query Q has B bins}
2: $b_m \leftarrow$ A bin in query Q of index $m \mid \Sigma_{i=1}^{m}\{h_i\} \geq \frac{Count}{2}$ and $\Sigma_{i=m}^{B}\{h_i\} \geq \frac{Count}{2}$,$\forall h_i \in H$ {Find the median bin to use its boundaries}
3: V = GetVALUES(T, Lb_m, Ub_m) {V is an array of all values in T within b_m boundaries (Lb_m and Ub_m)}
4: V = SORT (V) {Sort V values in ascending order}
5: $C \leftarrow \frac{Count}{2} - \Sigma_{i=1}^{m-1}\{h_i\}$
6: **return** $Median \leftarrow C^{th}$ value in V.

sensors in the tree, it uses the cached partial states to prune branches leading to subtrees that have no values within the requested boundaries. In the worst case, all values would be in leaf nodes in the maximum tree's level L. The maximum possible overhead per round is the number of bits to retrieve these values which is $log_2(h_m \times L \times max(v_j))$ where $1 \leq j \leq N$ and m is the bin id that contains the median value.

Algorithm 2 GetVALUES(Tree T, lower bound L, upper bound U)

1: {This function uses the partial state P_j in each node s_j to collect required values efficiently.}
2: V = { } {It will contain a set of collected values}
3: $s_j \leftarrow$ The root of tree T
4: **if** $v_j \in [L, U]$ **then**
5: Insert v_j in V {v_j is the value of sensor s_j}
6: **if** $\Sigma_{i=1}^{B}\{h_i\} = 1 \mid b_i \cap [L, U] > 0$, $\forall h_i \in P_j$ and $\forall b_i \in$ *Histogram* query Q **then**
7: **return** V {The single value in the tree is already found}
8: **end if**
9: **end if**
10: **if** $\Sigma_{i=1}^{B}\{h_i\} > 0 \mid b_i \cap [L, U] \neq \phi$, $\forall h_i \in P_j$ and $\forall b_i \in$ *Histogram* query Q **then**
11: **for** each children c of s_j **do**
12: Insert $GetVALUES(c$'s Tree, $L, U)$ to V {Insert returned values to the array of values V}
13: **end for**
14: **end if**
15: **return** V

Figure 1 illustrates how HIU works, in particular how the task of obtaining the reduced set of candidate values, if executed carefully as proposed above, may be fairly inexpensive.

Assuming that all values in the WSN are in the range [0,100], the query $Q(0, 100, [0, 25), [25, 50), ..., [75, 100], \tau)$ is issued at the outset. Note that (1) we use four bins in this example only for the sake of illustration –we investigate the effect of this parameter in the next section– and (2), the epoch τ is not relevant for in this single-round example, thus we left it unspecified. HIU's algorithm will proceed and obtain the answer $H = \{4, 1, 1, 5\}$ at the base-station. The median value is obviously in the third bin, namely within the range [50, 75), at this point the base-station requests all values in that range. It is important to note that the whole sub-tree rooted at node B need not be traversed as its local histogram does not contain any value in the median bin, i.e., it fails the test on line 10 in function GetValues() presented above. Likewise, node A satisfies the test at line 6 (there is a single value in the bin and that value is in that node) the query need not be propagated any further down the subtree. At the end only the single value of 72 is returned at the base station, and the query is answered.

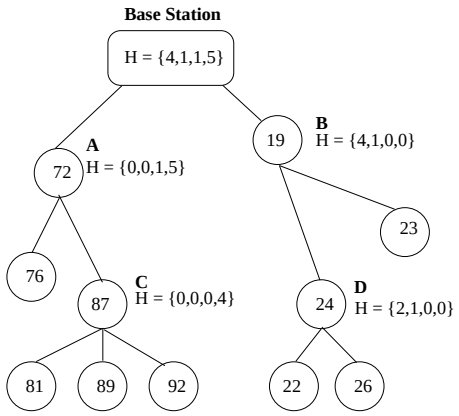


Figure 1: Processing a *Median query.*

3. PERFORMANCE EVALUATION

In our simulation we implemented TAG and HIU assuming both of them are using a Shortest Path (logical) Tree (SPT) for the underlying tree routing T. We make the following assumptions about the required storage: (1) an observed/sensed node value consumes 2 bytes, (2) a complete histogram size depends on the number of bins, i.e., it requires $2 \times B$ bytes, where B is the number of the bins in the histogram, and (3) updating a histogram bin require 3 bytes, 1 for the bin id and 2 for bin's count.

We investigate our algorithms with respect to five parameters (Radio range R, histogram size in terms of number of bins B, average amount of change in sensor's value δ, the probability that a sensor's value change ρ, and number of nodes in the WSN N).

The radio range controls the logical network topology and may increase/decrease network depth. Varying the histogram size shows the scalability of our algorithm from the point of view of the histogram size. Studying δ and ρ shows the sensitivity of our algorithm against the behavior of the values in the WSN field. It is important to show the influence of these two parameters because our algorithm depends on incremental updates which might be very large if many changes happen. Finally, increasing the number of sensors N, shows the algorithm scalability from the point of view of the WSN density.

Table 2 lists all tested values for all parameters. While testing one parameter, we use the default value (denoted in bold) of all other parameters. The reported show the average values obtained over 20 runs. During each run, the sensor locations are randomly distributed and the base station is randomly selected among one of the sensors. In order to ensure a fair comparison, both TAG and HIU use exactly the same setup.

We used two datasets, a synthetic and a real one. Due to space constraints we do not show the results obtained using the real dataset, nonetheless, the results we obtained using

Parameter	Used Values
R (WSN node's radio range)	$20, \mathbf{30}, 40, 50, 60$
B (Histogram size in terms of number of bins)	$5, 10, \mathbf{20}, 40, 60$
δ (Average amount of change)	$1\%, 25\%, \mathbf{50\%}, 75\%, 100\%$
ρ (Probability of change)	$1\%, 25\%, \mathbf{50\%}, 75\%, 100\%$
N (Number of Sensors)	$1000, 2000, \mathbf{3000}, 4000, 5000$

Table 2: Studied parameters and their values (default values are shown in bold)

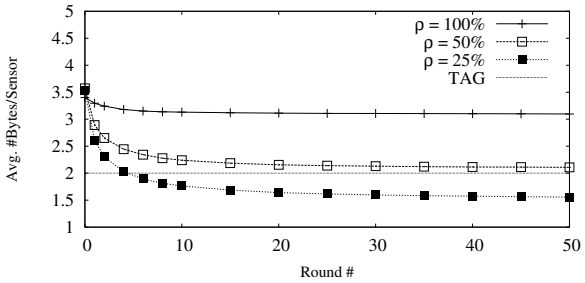
the real dataset are available in an extended version of this paper [2].

A synthetic dataset allows us to discuss how the algorithms behave with respect to some important parameters. For instance, since the observed values in a real dataset are given, one cannot investigate how the algorithm behave when the observed values change more or less frequently. Our synthetic dataset consists of N nodes uniformly distributed in an area of $200m \times 200m$. The values of all nodes use 2 bytes and are initialized uniformly between 1 and 2^{16}. In each round, a sensor's value could change with a probability ρ. In case of change, a sensor value is increased by an exponential random variable (equally likely to be negative or positive). The exponential random variable was chosen to allow very large and very small changes. The average of the exponential random variable is $\delta\%$ of 2^{16}. We assume that all nodes are capable of sensing values between 0 and 2^{16} only. If a value exceeds that range in either direction, it is assumed to be either 0 or 2^{16}, respectively.

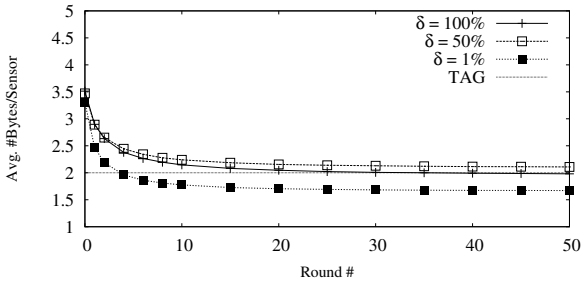
Because the main target of our experiment is to study HIU's cost for computing an exact answer, we use the average accumulated amount of bytes sent per sensor per round as our performance indicator. Every round, the total number of sent bytes from all nodes during all previous rounds are calculated and then divided by number of sensors, thus providing an *amortized* query cost. For brevity we constrain our presentation to two particular types of queries: *Max* and *Median*.

Based on [6], every sensor should send exactly 2 bytes to collect the maximum value using TAG. HIU collects the *Max* information while constructing the histogram. HIU's performance depends on the amount of changes in the network because it uses in-network caching and send data to update this cache. In general, HIU requires more bytes to be sent at the very start, but as time goes, the average total number of sent bytes per round is decreased and eventually reaches a steady state. Recall that the first round in HIU consumes a large amount of energy due to sending the largest amount of bytes comparing to other rounds because there is no cached information.

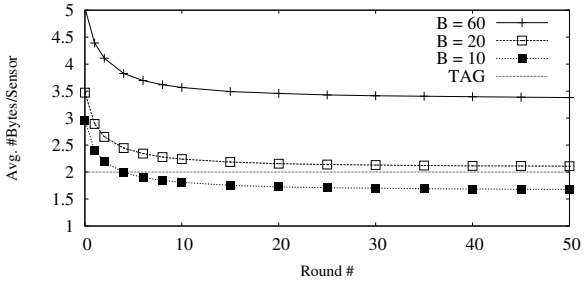
Figure 2(a) shows the influence of change probability on HIU. For exampe if $\rho = 100\%$, and compared to TAG, HIU needs about one additional byte from each sensor (on average) per round. As the probability gets smaller, this overhead decreases. The figure shows that lower values of ρ lead to a smaller HIU cost but TAG's performance stay the same. If $\rho = 1\%$ (not shown in the figure), HIU outperforms TAG by about 1.8 bytes which means 90% less bytes than TAG. It is worth mentioning that HIU's cost also includes constructing an accurate histogram in the base station, hence allowing approximate answers for other aggregate querie at no extra cost, while TAG (in this experiment) computes the maximum value only. The histogram in the base station



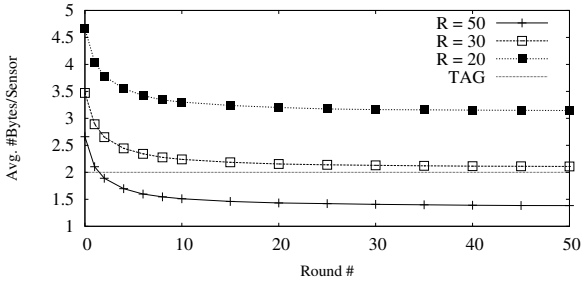
(a) Probability a value changes (ρ)



(b) Average amount of change (δ)



(c) Histogram Size (B)



(d) Radio Range (R)

Figure 2: Cost of processing an exact *Max* query

offers computing approximate answers for many other aggregate queries. This means, if the target is computing the *Max* query only, then HIU is better only if sensors change their values not very often ($\rho \leq 40\%$).

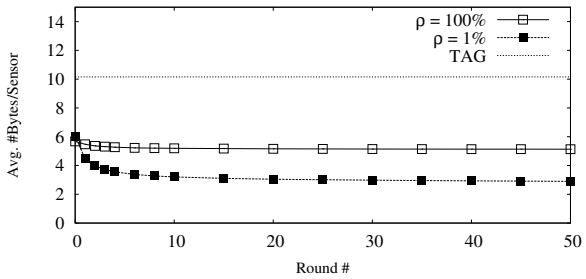
In Figure 2(b) we assume that $\rho = 50\%$ and investigate the influence of the amount of change (δ). If δ is very small (1%) HIU will outperform TAG. In the special case where $\delta = 100\%$, HIU practically ties with TAG in the long term. Recall that a sensor can sense a specific range of values. If the value is bigger than maximum value, a sensor will report its maximum limit. If $\delta = 100\%$ then there is a high probability that all sensors end up detecting only the maximum or minimum limits because the change could be positive or negative. It is clear that the amount of change does not have a significant influence on the results. In fact, regardless of the amount of change, the probability of change (ρ) has the main influence not the amount of change. The only exception is the change of 1% curve because changing a sensor value by 1% on average will unlikely change its bin in the histogram, if the bin's width is reasonably large, and then will unlikely cause a sensor to send any data.

HIU performance depends on the bin size (number of bins) because the number of values in smaller bins is more likely to change between rounds. Although the error bound of all approximate answers worsens when bin size increase, the HIU algorithm performs better while computing the *Max* query. Figure 2(c) shows that decreasing the number of bins can make HIU outperform TAG very early even if the probability of change and amount of change are both 50%. Recall that TAG outperforms HIU in Figures 2(a) and 2(b) when δ or ρ equals 50%. The major fraction of the HIU cost is paid to construct the histogram. Hence, decreasing the histogram size decreases the histogram overhead.

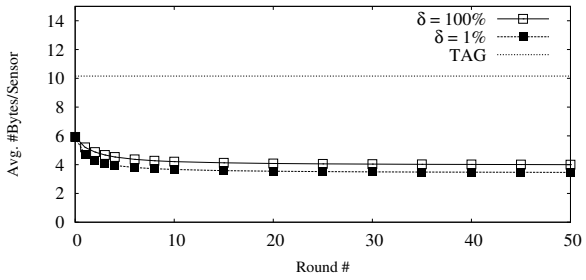
The sensor's radio range influences the logical tree structure. A short radio range requires the WSN to build a logical tree with larger depth than a long radio range. Increasing the average number of hops for sensors to reach the base station does not have any influence on TAG because every sensor will send a single message of fixed size (2 bytes) any way. The shorter the radio range, the more the number of hops which requires HIU to send more bytes. Figure 2(d) shows that increasing the radio range makes HIU's total cost less than TAG's total cost after 3 rounds only.

Even though a graph is not shown we also investigated the influence on the number of nodes in the network. Network density depends on the number of sensors and it has no influence on the TAG algorithm to compute *Max*. In all cases, each sensor should report its value. In the case of HIU, the more sensors available in the area the more opportunities to save and decrease the amount of sent messages.

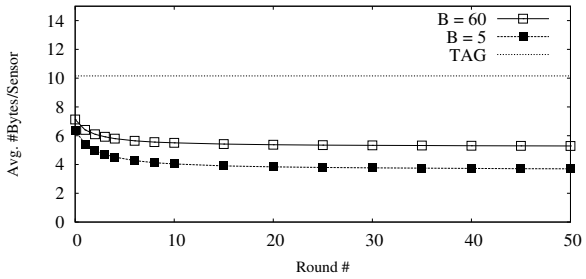
Based on [6], TAG suggests collecting all values to the base station in order to compute an exact answer for *Median* queries. Overall, Figure 3 shows that collecting all values to compute an exact *Median* cannot outperform HIU even in the first round. The cost of the *Median* query using HIU is twofold: histogram's cost and values collection's cost. In the first round, HIU does not reuse any existing information. However, in the consecutive rounds, HIU reduces the number of bytes sent by each sensor per round to construct the histogram. Typically, HIU requests a smaller number of values to be collected than TAG. Instead of sending the collection query to all nodes in the WSN, the cached information (in each node) is used to direct the query only to



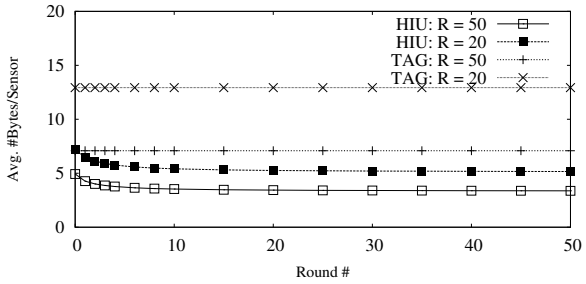
(a) Median-Probability a value changes (ρ)



(b) Average amount of change(δ)



(c) Histogram Size (B)



(d) Histogram Range (R)

Figure 3: Cost of processing an exact *Median* query

relevant nodes as explained earlier in Section 2.2.

In Figures 3(a) and 3(b) we show the influence of changing ρ and δ which affect histogram's cost only with no influence on the values collection's cost. The smaller the ρ, the cheaper the histogram and then the cheaper the Median. Figure 3(b) confirms our earlier finding that δ does not have a significant influence on the query cost. On the other hand, the smaller the ρ, the less expensive the histogram and then less expensive is the *Median*.

Figure 3(c), shows that changing the histogram size affects both the histogram's cost and the values collection's cost. The smaller the number of bins in the histogram, the less expensive the query cost but the more values are required in the values collection phase. However, the overall query cost is reduced for two reasons: (1) All nodes are guaranteed to participate in the histogram construction and then guaranteed to pay its cost. (2) The values collection phase uses the local histogram in each node to prune sub-trees whenever possible.

Figure 3(d) shows the influence of the sensor's radio range (R) on the *Median* query cost. This parameter affects the TAG cost as well as the HIU cost because it changes the logical routing tree. However, TAG cost is fixed regardless to number of rounds. In Figure 3(d) one can see that the larger the sensor range, the smaller the number of hops to reach the base station and then the smaller number of bytes sent for both TAG and HIU.

4. RELATED WORK

To our knowledge this is the first proposal for reusing histograms for processing other aggregate queries exactly. In fact, there has been not much work done to construct a histogram of WSN values since Madden et. al. proposed TAG algorithm in 2002 [6]. Chow et al proposed an algorithm to construct a spatio-temporal histogram [3]. This approximate histogram is used for location monitoring. Since our algorithm constructs an exact histogram using an in-network algorithm and we do not require all values to be sent to the base station, we did not compare their approach with HIU. However, we believe their algorithm for location monitoring can work on top of our exact histogram and still save sensors' energy.

There are a few algorithms proposed in the literature to answer more complex aggregate queries like *Median*. Algorithms proposing approximate answer for Median typically have a bound on the median's rank not the median's value (e.g. [10, 9]). That is, they ensure the rank of an approximate answer is $\lceil N/2 \rceil \pm \varepsilon$. Our proposal, again, provides an exact answer for *Median* queries.

5. CONCLUSION

In this paper we proposed extensions to HIU [1] that uses in-network aggregation and in-node caching to reduce the energy consumption for answering a histogram query. These extensions may be used to compute the exact answer for several other aggregate queries. Obtaining a histogram in the base station helps in computing bounded approximate answers for other aggregate queries as well. Using the *Max* and *Median* queries as examples, our experiments have shown that HIU outperforms the TAG algorithm [6], on average tripling the network's lifetime. In our future work, we would like to explore how to use the proposed technique to reduce the overall cost in the context of multiple/concurrent query optimization.

Acknowledgements

This work was partially supported by NSERC, Canada.

6. REFERENCES

[1] K. Ammar and M. A. Nascimento. Histogram and other aggregate queries in wireless sensor networks. In *Proc. of SSDBM*, 2011. To appear.

[2] K. Ammar and M.A. Nascimento. Histogram and other aggregate queries in wireless sensor networks. Technical Report TR 11-03, Department of Computing Science, University of Alberta, 2011.

[3] C.Y. Chow, M.F. Mokbel, and T. He. Aggregate location monitoring for wireless sensor networks: A histogram-based approach. In *Proc. of MDM*, pages 82–91, 2009.

[4] S. Collins et al. New opportunities in ecological sensing using wireless sensor networks. *Frontiers in Ecology and the Environment*, 4(8):402–407, 2006.

[5] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon. Health monitoring of civil infrastructures using wireless sensor networks. *Proc. of IPSN*, pages 254–263, 2007.

[6] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):131–146, 2002.

[7] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. *Proc. of ACM SIGMOD*, pages 491–502, 2003.

[8] E.D. Pinedo-Frausto and J.A. Garcia-Macias. An experimental analysis of zigbee networks. In *Proc. of LCN*, pages 723 –729, 2008.

[9] S. Roy, M. Conti, S. Setia, and S. Jajodia. Securely computing an approximate median in wireless sensor networks. *Proc. of SecureComm*, (6):1–10, 2008.

[10] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri. Medians and beyond: new aggregation techniques for sensor networks. *Proc. of SenSys*, pages 239–249, 2004.

[11] J. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood. Wireless sensor networks for in-home healthcare: Potential and challenges. *Proc. of HCMDSS*, 2005.