The 18th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'07)

A NOVEL QUERY OPTIMIZATION METHOD FOR WIRELESS SENSOR NETWORKS

Jun-Zhao Sun
Academy of Finland
Information Processing Laboratory, Dept. of Electrical and Information Engineering, University of Oulu
Oulu, Finland

ABSTRACT

This paper presents a novel query optimization method for wireless sensor networks. By taking advantage of the delay constraint specified with a query, the method can find the optimal combinations of sending sensor data from nodes to sink, with or without performing data processing. System models for cost and timing of both computation and communication are created. Conditions of performing optimization are described accordingly. Three cases of possible optimization are discussed. Finally, performance of the proposed method is evaluated. It shows that the proposed method achieves better performance than without performing the optimization.

I. INTRODUCTION

Sensor networks represent significant improvement over traditional sensors in many ways [1]. However, sensor nodes have very limited supply of energy, and should be available in function for extremely long time (e.g. a couple of years) without being re-charged. Therefore, energy conservation needs to be one key consideration in the design of the system and applications. Extensive research work has been devoted to address the problem of energy conservation. Examples include energy efficient MAC protocol [2], clustering [3], localization [4], routing [5], data management [6], etc.

A sensor field is like a database with dynamic, distributed, and unreliable data across geographically dispersed nodes from the environment. These features render the database view [7, 8] more challenging, particularly for applications with the low-latency, real-time, and high-reliability requirements. Query processing is employed to retrieve sensor data from the network [9, 10]. A general scenario of querying sensor network is, when user requires some information, he or she specifies queries through an interface at sink (also known as gateway, base station, etc.). Then, queries are parsed and query plans are made. After that, queries are injected into the network for dissemination. One query may eventually be distributed to only a small set of sensor nodes for processing. When sensor node has the sampling data ready, results flow up out of the network to the sink. The data can then be stored for further analysis and/or visualized for end user. Query optimization can be made through out the process in all the stages [9, 10].

This paper presents a novel query optimization method for wireless sensor networks, and in particular, for the last stage of query processing: query result collection. The proposed method is applicable for the optimization of both one-time query and periodical query. The key novelty of the method lies on the careful consideration of timing issue along with energy consumption in both computation and communication. By taking advantage of the delay constraint specified with a query, the method can find the optimal combination of transmitting sensor data to sink, with or without data processing. The paper also takes into account the situation where multiple queries are simultaneously under processing in one single sensor networks.

The remainder of this paper is organized as follows. Section II describes the system models for cost and timing of both computation and communication. Section III describes the proposed method in detail. Performance of the proposed method is evaluated in Section IV. Finally, Section V concludes the paper.

II. SYSTEM MODEL AND PROBLEM FORMALIZATION

A query plan is executed with two components, computation and communication. Computation component concerns data sampling by sensors, as well as local and in-network data processing. Communication component allows a set of spatially distributed sensor nodes to send data to a central destination node. Therefore, the power consumption of the sensor network consists of two types of energy cost, data processing cost and data transmission cost. The system and energy cost are modeled in the section.

A sensor network is modelled as a graph $G = (V, E)$, where V denotes the node set and E the edge set representing the links between node-pairs. An edge $e \in E$ is denoted by $e = (u, v)$, where u is the start node and v is the end node.

Data processing cost, $C_P(u)$ denotes energy consumption for the data processing at the node u. The processing includes all the data manipulations from single data processing (data sampling and signal processing, raw data filtering, compression/decompression) to multiple data processing (packet merge, data compression, and data fusion). Data processing results in one or a series of packets ready to be sent. Energy consumption for single data processing at a specific node u is fixed, and so can be represented by a constant $C_{SDP}(u)$. On the contrary, energy cost for multiple data processing depends on the amount of data to be processed as well as the algorithms utilized. First the unit processing cost on node u is defined as $C_{PU}(u)$. Then, the cost for processing the $D(u)$ amount of data at node u is given by

$$C_P(u) = C_{SDP}(u) + C_{PU}(u)\, D(u). \quad (1)$$

Here $D(u)$ is usually a set of sampling result data for one single or different queries. Note that after the data processing, the amount of data that is significant to be sent, $D'(u)$ (i.e. payload in packet) is usually less than the original amount $D(u)$. Thus, data reduction ration, ru can be defined as

$$D'(u) = D(u)\,(1 - ru) \quad (2)$$

The real value of *ru* is mostly depending on the fusion/compression algorithms utilized as well as the similarity/correlation between data samples.

Similarly, we can define the time for data processing as

$$T_P(u) = T_{SDP}(u) + T_{PU}(u)\, D(u), \quad (3)$$

where $T_{SDP}(u)$ is the time for single data processing at node u and $T_{PU}(u)$ is the unit processing time at this node.

It is worth noting that C_{PU} and T_{PU} are relevant to *ru*, depending on the processing algorithm used. Basically, the higher the *ru*, the higher the C_{PU}/T_{PU}. For example if a simple packet merge is performed, then *ru* will be very small, and the corresponding C_{PU} and T_{PU} will be very low. On the contrary, if some complex data compression algorithm is utilized, then a much better ru will be reached with fairly high C_{PU} and T_{PU}.

By using similar concepts above, data transmission cost and data transmission time can be modeled as well. One packet contains two parts, header and payload. Packet header is the packet overhead that is the same for all the packet, which contains numbering, addressing and error checking information. Transmission cost denotes the cost for transmitting $D(u)$ amount of data (payload) from node u to node v through link $e = (u, v)$. The cost includes the energy consumption at both u and v. Unit cost of the link for transmitting data between two nodes can be abstracted as $C_{TU}(e)$, and thus the transmission cost $C_T(e)$ is given by

$$C_T(e) = C_{PHT}(e) + C_{TU}(e)\, D(u), \quad (4)$$

where C_{PHT} is the constant transmission cost of packet header. Similarly, transmission time $T_T(e)$ is given by

$$T_T(e) = T_{PHT}(e) + T_{TU}(e)\, D(u), \quad (5)$$

where $T_{TU}(e)$ is the unit transmission time, i.e. the reciprocal of bandwidth. We note that all the cost and time parameters are all defined on link e, because different link has different conditions e.g. distances, congestion, and reliability.

Finally, the definition can be easily extended to the transmission cost and time for a *path*. The cost and time for $D(u)$ from one node x to another node y through a multihop path x->y can be represented as

$$C(D(x): x\text{->}y) = \sum_{e \in x\text{->}y} C_T(e) + \sum_{u \in x\text{->}y} C_P(u), \quad (6)$$

$$T(D(x): x\text{->}y) = \sum_{e \in x\text{->}y} T_T(e) + \sum_{u \in x\text{->}y} T_P(u). \quad (7)$$

To apply the model in this paper, some assumptions and approximations are needed to make the model treatable in practice. Here, we assume that data processing occurs only on the start node that is under study. This means all the rest nodes on the path will only relay the packet without performing any manipulations. The assumption in reasonable because only queries with no aggregation operations are considered in this paper. Moreover, the algorithms presented in next section is actually still applicable to the case that there are in-network data fusion performed by media nodes. After this assumption, the second parts of the two formulas above are changed to $C_P(x)$ and $T_P(x)$ respectively, as below.

$$C(D(x): x\text{->}y) = \sum_{e \in x\text{->}y} C_T(e) + C_P(x), \quad (8)$$

$$T(D(x): x\text{->}y) = \sum_{e \in x\text{->}y} T_T(e) + T_P(x). \quad (9)$$

It is worth noting that in the following study the destination node y is always the single sink node (denoted by s hereafter).

Therefore to formalize the problem to be addressed, suppose there is a *delay* constraint specified in the queries, our goal is *to find an optimal query execution plan such that the sum of energy cost can be minimal, while the query results must be reported on time.*

III. Query Optimization Algorithm

This section describes the detail optimization algorithms. It starts with the hypothesises of the algorithms, followed by the analysis of three different cases. Finally, a simplification is presented to loose a strict hypothesis.

Hypothesises

To derive the algorithms for query optimization, following hypothesises are defined in this paper.

1) The topology of the sensor network is organized as a spanning tree, with the sink as the root, and tree-based routing algorithm is adopted.

2) In case of node mobility or link failures, the routing tree will be automatically re-configured by the underlying topology control protocols.

3) Downlink (from sink to leaf nodes) and reverse uplink are symmetrical, meaning that the communication bandwidth is at the same level.

4) A global synchronization is achieved in the network, that is, all the nodes share a common clock.

5) There exists a restriction on the max packet size (MPS) allowed for transmission in the sensor network.

6) The data reduction ratio *ru* and unit processing cost/time C_{PU} & T_{PU} for each node are available.

7) Every node has the full knowledge of the whole network, including the routing tree composition, the C_{PHT}/C_{TU} and T_{PHT}/T_{TU} of every link, as well as all the living queries in the sensor network.

The core of the optimization algorithm is to make a decision about whether to perform the optimization, and if yes, when. Therefore, the description of the optimization algorithms below will concentrate mostly on the conditions when to make the optimization.

Case 1. Multiple snapshot queries.

This is the case where there are multiple living one-time queries at a node. Instead of immediately sending the sampling data directly to the sink node s, the node may choose to wait for other sampling data of other queries and send a set of data together, as long as all the *delay* constraints are obeyed, as illustrated in Figure 1. To make the decision of whether a query optimization should be performed at the current node u, three conditions should be fulfilled, as follows.

Condition 1. Packet size. According to Hypothesis 5, in case of optimization, the final size of the combined data must not be bigger than the specified maximum packet size, i.e.

$$D'(u) = D(u)\,(1\text{-}ru) < \text{MPS}. \quad (10)$$

With this condition, the queries that can be considered for optimization can thus be selected.

Condition 2. Time for transmission. If optimization is performed, there should be sufficient time left in order to report the query results on time (i.e. before the specified *delay*), i.e.

$$T(D'(u)\text{: } u\text{->}s) < \text{FirstReportTime} - \text{LastSamplingTime}. \quad (11)$$

Left side of the formula is give by formula (9). Figure 1 depicts this time features. After checking the condition with the selected queries from Condition 1, a sub-set of queries will be obtained, denoted *Sq*. The number of the queries in this sub-set is denoted by *n*.

Condition 3. Energy saving gain. At last, the effectiveness of the optimization need to be estimated. This is done by comparing the overall cost of *Sq* for optimization with the cost when no optimization is performed, i.e.

$$C(D'(u)\text{: } u\text{->}s) < \sum_{q \in Sq} C(D(q) : u\text{- > } s), \quad (12)$$

where $D(q)$ denotes the data size of query q, the sum of $D(q)$ is $D(u)$. Right side of the formula above is the sum cost of separately sending each individual query result to the sink without the data combination in optimization. By using formula (1), (2), (4) and (8), after derivation, below formula can be achieved.

$$C_{PU}(u)\, D(u) < (n\text{-}1) \sum_{e \in u\text{->}s} C_{PHT}(e) + \sum_{e \in u\text{->}s} C_{TU}(e)\, D(u)\, ru. \quad (13)$$

That is, extra cost for data processing (left side) is less than the gain in transmission cost reduction due to data combination (right side).

Case 2. Periodical query.

This is the case where there is a continuous query at a node. The basic idea is the same as of Case 1, as illustrated in Figure 2. One special feature is that, due to the periodical nature of the queries, the size of all the query result data as well as all the corresponding delays are actually the same. Therefore, the query selection in Condition 1 and 2 in Case 1 can be simplified to the problem of finding the number of sampling data (i.e. loops) that can be combined.

In addition, the calculation and decision need to be carried out only once. After that, the number of data units to be combined is fixed, and therefore can be utilized during the whole query execution without further calculation. The three conditions in Case 1 all hold for this case, thus the detailed derivation is omitted here.

Case 3. Cross-node queries.

Case 1 and 2 consider only the optimization of local queries, that is, a node only optimizes the queries living at the local node. The general situation of a sensor network is that,

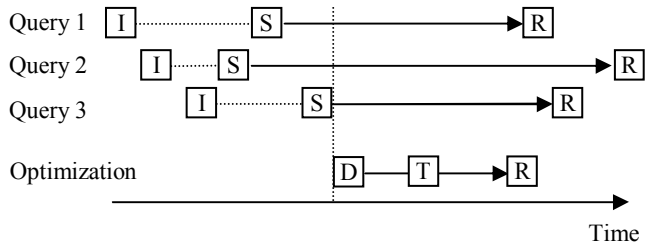


Figure 1: Query optimization for multiple snapshot queries. I – time of query injection; S – sampling time; R – latest report time; S to R – allowed delay; D – starting data processing; T – starting transmission.

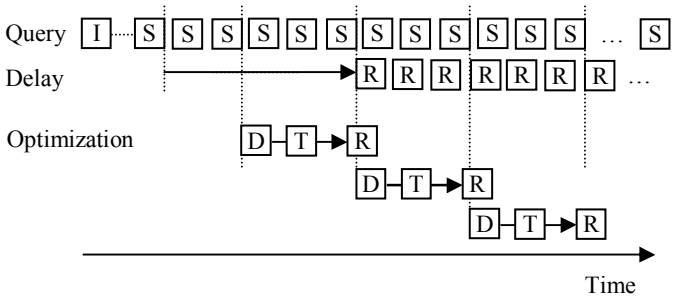


Figure 2: Query optimization for periodical query.

at a moment there are multiple queries executing in the network, with each query executing at multiple nodes simultaneously. On the other hand, nodes in a sensor network work not only as a sensor, but at the same time a router in the routing tree that taking care of receiving and forwarding packets on the path. Therefore it is possible for the median nodes to perform optimization above both the local data as well as the data from other nodes.

The basic idea is similar to Case 1, except that some data may come from other nodes (its offspring nodes). In case of multiple simultaneous queries living in the network, each sensor data will be attached with a query ID to denote the owner of the data. The median node can then intercept, extract, and parse the query ID for every packet to be relayed. According to Hypothesis 7, the corresponding information about the query e.g. the latest report time is available at the local node. Thus, all the necessary information is ready for the median node to make a decision whether or not an optimization will be performed accordingly. The rest procedures are exact the same as in Case 1 above.

Simplification.

The Hypothesis 7 requires every node has the full knowledge of the network: routing, query, and especially each single link in terms of cost and time, which is too strict. Even this is feasible with respect to the node's storage capacity, the overhead of information updating will be very high for the cases of new node coming, node failure, mobile node, or link fluctuations. In order to make the scheme more relevant in the real word, simplification is needed to loose the strict hypothesis, as discussed below.

Before data collection, in order to support the proposed optimization algorithms, there must be some pre-processing during query dissemination stage, as follow.

a. Every query that is injected from the sink into the sensor network for dissemination is associated the moment of time of injection and a hop counter.

b. During the query dissemination, the hop counter is updated with current number of hops (i.e. the level of the routing tree) and sent along the way until completed.

c. On receiving a query, a node stores the query ID, time of injection, value of the hop counter, time of receipt, and the query itself (attributes, delay, sampling time, interval, loop, etc.).

d. The query entry stored expires when either execution is completed (in the case when the query is for this node, according to the *where* clause.) or the latest time to report result elapses (if the query is not for this node).

With this series of pre-processing in query distribution, Hypothesis 7 can thus be simplified so that *each sensor node needs to know only local information*. First, every node needs to have only the knowledge of local routing information, i.e. maintain a record of its parent node and all the next level offspring nodes. Second, every node only needs to know the queries that are on its branch.

Further simplifications can be made by studying formula (8) and (9), by considering the path as a whole from source node to sink instead of every individual links on the path. In formula (8), the first part is the sum transmission cost of all the links from node x to y, and can be approximate by assuming that the average cost of unit data transmission is available. In this case, universal C_{TU} can be defined for all the links, and therefore the first part (i.e. cost for data transmission through the path) can be approximated by

$$C_T(D(x): x->y) = noh\ C_{TU}\ (hs + D(x)), \quad (14)$$

where *noh* (Number of Hop) is number of hops on path $x->y$, and *hs* (Header Size) is header size of the packet.

Similarly, in formula (9), the first part is the sum transmission time of all the links from node x to y, and can be simplified by defining a universal T_{TU} for all the links. Instead, here we present another simplification method based on the estimation of real measurement. The first part of formula (9) is replaced with the transmission time for path $x->y$, given by

$$T_T(D(x): x->y) = T_{TU}(x->y)\ (hs + D(x)). \quad (15)$$

By taking advantage of the parameters collected in pre-processing, the unit transmission time from a node x to the sink s, $T_{TU}(x->s)$ can be estimated by

$$T_{TU}\ (x->s) = \frac{\text{Time\_of\_Receipt} - \text{Time\_of\_Injection}}{\text{Length\_of\_Query}}. \quad (16)$$

Finally, with the same idea above, data processing cost and time in formula (1) and (3) can be simplified by defining universal constant C_{PU} and T_{PU}, meaning the data processing cost and time are only relevant to the data size.

IV. Performance Evaluation

In this section, we compare the performance of queries with optimization with queries without optimization. We denote the query cost with performing optimization by *Cop* and without optimization by *Cnop*. Furthermore, without losing any generality, we assume the data size for all the queries is the same, denoted by *Dq*. Thus, the volume of data before optimization at a node u is given by

$$D(u) = noq\ Dq, \quad (17)$$

where *noq* (Number of Query) is the number of selected queries to be optimized. Therefore, the following cost metrics can be defined according to formula (8) and (14):

$$C_{op} = noh\ C_{TU}\ (hs + noq\ Dq\ (1\text{-}ru)) + C_{PU}\ noq\ Dq, \quad (18)$$

$$C_{nop} = noq\ noh\ C_{TU}\ (hs + Dq), \quad (19)$$

$$\text{Energy\_Saving} = C_{nop} - C_{op}$$
$$= (noq - 1)\ noh\ C_{TU}\ hs + noq\ Dq\ (noh\ C_{TU}\ ru - CPU). \quad (20)$$

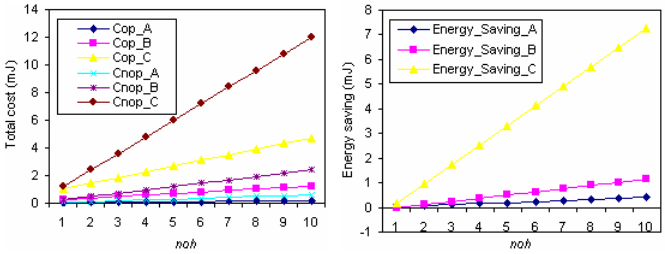
Table 1 shows the parameters, their typical value ranges, and values used in the performance analysis.

Table 1: Parameters, their value ranges, and values used in the performance analysis.

parameter	value range	used value
C_{PU}	0-100nJ/bit	5, 30, 80nJ/bit
C_{TU}	20-400nJ/bit	100nJ/bit
ru	0-1	0, 0.3, 0.5
noh	1-50	8, 10
noq	1-10	5
hs	10-30 bytes	10
Dq	5-500 bytes	5, 50, 100, 200

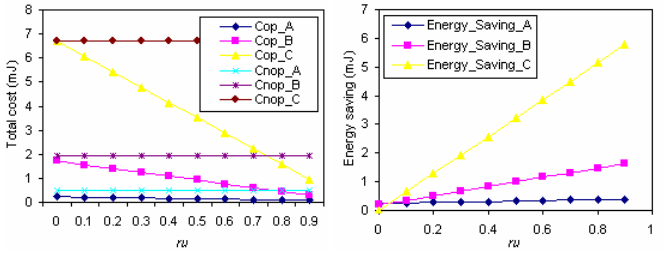
First the impact of *noh* is studied. Figure 3(a) illustrates the total cost of collecting 5 query results from a sensor node with different distance to the sink (i.e. *noh*). The C_{PU} is set to 80 nJ/bit, *ru* is 0.5, *Dq* is set to 5, 50, and 200 bytes for case A, B, and C, respectively. Obviously, total cost increases with the increasing of *noh* and *Dq*. Figure 3(b) shows the performance gain in terms of energy saving. From the figure, it is clear that *noh* plays an important role in the optimization. With the increase of the hops from the node to the sink, the performance gain from the optimization becomes more and more significant. Also, the effectiveness of the optimization becomes better when the size of data reading for the queries increases. This is because the total energy saving resulted from the decrease in packet size is increasing when more data is relayed by more median nodes.

It is worth noting that there exist one negative point in Figure 3(b) in terms of energy saving, where *noh* = 1 for case B. This means that the optimization does not necessarily lead to performance gain in terms of energy saving. When node is very close to the sink, to direct send raw data without processing might be a good choice.



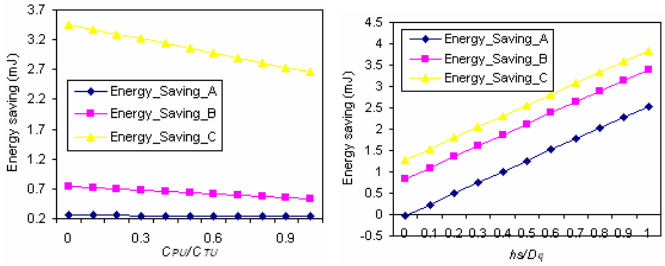
(a) Total cost. (b) Energy saving.

Figure 3: Impact of *noh* to energy consumption.



(a) Total cost. (b) Energy saving.

Figure 4: Impact of *ru* to energy consumption.



(a) C_{PU}/C_{TU}. (b) hs/Dq.

Figure 5: Impact of two ratios to energy consumption.

The impact of *ru* is illustrated by Figure 4. This time C_{PU} is set to 30 nJ/bit, noh is 8, Dq is still 5, 50 and 200 for the three cases. Since *ru* is only used in optimization, the total cost of non-optimization is actually constant, as shown in Figure 4(a). For cases with optimization, the total cost is larger with bigger Dq, and decreased with the increasing of *ru*. Figure 4(b) shows the performance gain. The energy saving for small reading data has no significant increase with the increasing of ru. For big amount of data as case C where Dq = 200, the energy saving becomes more sensitive to the change of *ru*. Therefore, good data compression algorithm should be used when a big amount of data is produced in sensor sampling.

It is interesting to note that two ratios, C_{PU}/C_{TU} and hs/Dq, have crucial impacts to the performance of query optimization. Therefore, the influences of the two ratios to the energy saving metric are studied, as illustrated in Figure 5(a) and (b). The Dq is set to 100 bytes, and *ru* is 0, 0.3, and 0.5 for case A, B, and C respectively. In Figure 5(a), *hs* is set to 10 bytes. From the figure, Energy saving decreases with the increasing of the ratio C_{PU}/C_{TU}. This is because more energy is needed in data processing. However, when *ru* is fairly big, which means a satisfied data processing method is available, then the performance gain is significant even C_{PU} is closing to C_{TU}. In Figure 5(b) the impact of hs/Dq is shown, in which C_{PU} is set to 5, 30, and 80 for the three cases. Obviously, the bigger the header size hs comparing to the sensor reading data Dq, the more effective the optimization in terms of energy saving.

V. CONCLUSIONS

A novel query optimization method is proposed for query processing during data gathering. Performance evaluation shows that the proposed method achieves better performance than without performing the optimization. We believe that consideration of timing issue, like user specified delay in this paper, provides new opportunities for query optimization. Also, energy consumptions in both computation and communication have to be explored. Finally, the paper also attempts to at the first step reveal the possibility of co-optimization of multiple queries.

ACKNOWLEDGMENT

The financial support by the Academy of Finland (Project No. 209570) is gratefully acknowledged.

REFERENCES

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Comunications Magazine*, vol. 40, no. 8, 102-114, Aug. 2002.

[2] Matthew J. Miller and Nitin H. Vaidya, "A MAC Protocol to Reduce Sensor Network Energy Consumption Using a Wakeup Radio," IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 4, NO. 3, 228-242, MAY/JUNE 2005.

[3] Y. Fukushima, H. Harai, S. Arakawa, and M. Murata, "Distributed clustering method for large-scaled wavelength routed networks," *Proc. Workshop on High Performance Switching and Routing*, 416-420, May 2005.

[4] Lingxuan Hu and David Evans, "Localization for Mobile Sensor Networks," Proc. *Tenth Annual International Conference on Mobile Computing and Networking* (MobiCom 2004). Philadelphia, 45-57, Sept.-Oct. 2004.

[5] J.N. Al-Karaki and A.E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, 6-28, Dec. 2004.

[6] Alan Demers, Johannes Gehrke, Raimohan Rajaraman, Niki Trigoni, and Yong Yao. Energy-Efficient Data Management for Sensor Networks: A Work-In-Progress Report. 2nd IEEE Upstate New York Workshop on Sensor Networks. Syracuse, NY, October 2003.

[7] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. Towards Sensor Database Systems. In *Proceedings of the Second International Conference on Mobile Data Management*. Hong Kong, January 2001.

[8] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, "TinyDB: An Acquisitional Query Processing System for Sensor Networks," ACM Transactions on Database Systems, vol.30, no.1, 122-173, Mar. 2005.

[9] J. Gehrke and S. Madden, "Query processing in sensor networks," *IEEE Pervasive Computing*, vol. 3, no. 11, pp. 46-55, 2004.

[10] Y. Yao and J. Gehrke, "Query processing for sensor networks,", In Proceedings of the *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, Asilomar, California, January 2003.