ETC: Energy-driven Tree Construction in Wireless Sensor Networks

P. Andreou[#], A. Pamboris[*], D. Zeinalipour-Yazti[#], P.K. Chrysanthis[‡], G. Samaras[#]

[#] Dept. of Computer Science , University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus
[*] Dept. of Computer Science and Engr., University of California - San Diego, CA 92093, USA
[‡] Dept. of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA
{p.andreou,dzeina,cssamara}@cs.ucy.ac.cy, apambori@cs.ucsd.edu, panos@cs.pitt.edu

Abstract

Continuous queries in Wireless Sensor Networks *(WSNs) are founded on the premise of* Query Routing Tree *structures (denoted as T), which provide sensors with a path to the querying node. Predominant data acquisition systems for WSNs construct such structures in an ad-hoc manner and therefore there is no guarantee that a given query workload will be distributed equally among all sensors. That leads to data collisions which represent a major source of energy waste. In this paper we present the* Energy-driven Tree Construction (ETC) *algorithm, which balances the workload among nodes and minimizes data collisions, thus reducing energy consumption, during data acquisition in WSNs. We show through real micro-benchmarks on the CC2420 radio chip and trace-driven experimentation with real datasets from Intel Research and UC-Berkeley that* ETC *can provide significant energy reductions under a variety of conditions prolonging the longevity of a wireless sensor network.*

1 Introduction

Large-scale deployments of WSNs have already emerged in environmental and habitant monitoring [8, 7], structural monitoring [3] and urban monitoring [6]. A decisive variable for prolonging the longevity of a WSN is to minimize the utilization of the wireless communication medium. It is well established that communicating over the radio in a WSN is the most energy demanding factor among all other functions, such as storage and processing [10, 4, 5, 11, 9]. The energy consumption for transmitting 1 bit of data using the MICA mote [1] is approximately equivalent to processing 1000 CPU instructions [5].

In order to process continuous queries in *Wireless Sensor Networks* (WSNs), predominant data acquisition frameworks typically organize sensors in a *Query Routing Tree* (denoted as T) that provides each sensor with a path over which query results can be transmitted to the querying node. We found that current methods deployed in predominant data acquisition systems sub-optimality construct T which leads to an enormous waste of energy. In particular, since T is constructed in an ad-hoc manner, there is no guarantee that a given query workload will be distributed equally among all sensors. That leads to data collisions which represent a major source of energy waste.

To facilitate our description, consider the example depicted on Figure 1 (left), which illustrates the initial ad-hoc query routing tree T created on top of a 10-node sensor network with the *First-Heard-From (FHF)* approach [1]. Assume that the weight on each edge of T represents the workload (e.g., the number of transmitted tuples) that incurs when a child node communicates its partial results to its designated parent node. In the example, we observe that node s_2 is inflicted with a high workload (i.e., 5 child nodes) while other nodes at the same level (i.e., s_3 and s_4), only have zero and one child nodes, respectively. Notice that both s_8 and s_{10} are within communication range from s_3 (i.e., the dotted circle), thus these nodes could have chosen the latter one as their parent. Unfortunately, the FHF approach is not able to take these semantics into account as it conducts the child-to-parent assignment in a network-agnostic manner.

Unbalanced topologies pose some important energy challenges which are summarized as follows:

Decreased Lifetime and Coverage: Since the majority of the energy capacity is spent on transmitting

[1]In FHF, each sensor s_i selects as its parent the first node from which s_i received the query.

978-0-7695-3650-7/09 $25.00 © 2009 IEEE
DOI 10.1109/MDM.2009.90

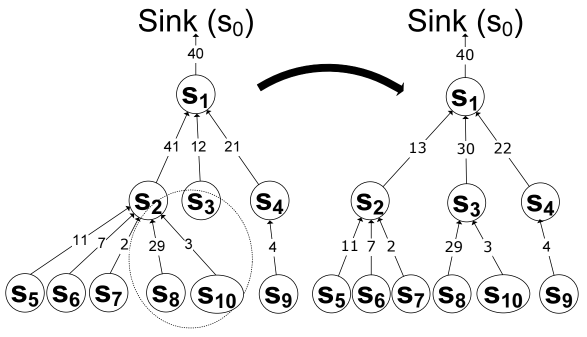


Figure 1. Left: The ad-hoc query routing tree; Right: The optimized workload-aware query routing tree constructed using the in-network ETC balancing algorithm.

and receiving data, the available energy of sensors with a high workload will be depleted more rapidly than the others. For example, in Figure 1 (left), we observe that sensor s_2's energy will be depleted 93/12=7.75 faster than s_3, that is $((\sum_{i=0}^{children(s_2)}(s_i, s_2) + (s_2, s_1))$ / $(\sum_{i=0}^{children(s_3)}(s_i, s_3) + (s_3, s_1))$, and 3.72 times faster than s_4 (i.e., 93/25). In addition, if s_2's energy is depleted and no alternate parents are available for sensors s_{5-7} then the coverage of the network will be reduced dramatically.

Increased Data Transmission Collisions: An unbalanced workload increases data transmission collisions which represent a major source of energy waste in wireless communication. Our micro-benchmarks on the CC2420 radio transceiver, presented in Section 4.1, unveil that crowded parent hubs like s_2 might yield loss rates of up to 80%, thus inflicting many retransmissions to successfully complete the data transfer task between nodes.

In Section 4.1, we show that the execution of a query over a node with 10 children will lead to a 48% loss rate of data packets, while executing the same query over a node with 100 children will lead to a 77% loss rate. These figures translate into an approximately threefold increase in energy demand due to inevitable retransmissions of data packets. Consequently, unbalanced trees can severely degrade the network health and efficiency.

Contributions: We present ETC, a distributed algorithm for transforming an arbitrary query routing tree into a near-balanced routing tree in which data collisions are minimized, thus Wireless Devices have the capability to power down their transceiver much earlier conserving valuable energy. We also validate experimentally the efficiency of our propositions with an experimental study that utilizes real sensor readings and micro-benchmarks.

2 Preliminaries and Background

In this section, we will provide an overview of balanced trees in order to better frame the problem the ETC algorithm seeks to improve. Balanced trees can improve the asymptotic complexity of `insert`, `delete` and `lookup` operations in trees from $O(n)$ time to $O(log_b n)$ time, where b is the branching factor of the tree and n the size of the tree. We will next formalize our discussion by providing some definitions:

Definition 1: Balanced Tree ($T_{balanced}$)
A tree where the heights of the children of each internal node differ at most by one.

The above definition specifies that no leaf is much farther away from the root than any other leaf node. For ease of exposition consider the following directed tree: $T_1 = (V, E) = (\{A, B, C, D\}, \{(B, A), (C, A), (D, B)\})$, where the pairs in the E set represent the edges of the binary tree. By visualizing T_1, we observe that the subtrees of A differ by at most one (i.e., $|height(B) - height(C)|$=1) and that the subtrees of B differ again by at most one (i.e., $|height(D) - height(NULL)|$=1). Thus, we can characterize T_1 as a balanced tree.

Notice that V has several balanced tree representations of the same height (e.g., the directed tree $T_2 = (\{A, B, C, D\}, \{(B, A), (C, A), (D, C)\})$). Similarly, V has also many balanced tree representations of different heights (e.g., the directed tree $T_3 = (\{A, B, C, D\}, \{(B, A), (C, A), (D, A)\})$ which has a height of one rather than two). Finally, in a balanced tree every node has approximately β children, where β is equal to $\sqrt[d]{n}$ (the depth of every balanced tree is $d = \log_\beta n$, thus $\beta^d = n$ and $\beta = \sqrt[d]{n}$). The ETC algorithm presented in this section focuses on the subset of balanced trees which have the same height to T_{input} as this makes the construction process more efficient.

In order to derive a balanced tree ($T_{balanced}$) in a centralized manner we could utilize the respective balancing algorithms of AVL Trees, B-Trees and Red-Black Trees. However, that would assume that all nodes are within communication range from each other which is not realistic. Thus, the ETC algorithm seeks to construct a *Near-Balanced Tree* ($T_{near\_balanced}$), defined as follows:

Definition 2: Near-Balanced Tree ($T_{near\_balanced}$)
A tree in which every internal node attempts to obtain a less or equal number of children to the optimal branching factor β.

The objective of $T_{near\_balanced}$ is to yield a structure similar to $T_{balanced}$ without imposing an impossible network structure (i.e., nodes will never be enforced to connect to other nodes that are not within their communication range). We shall later also define an error metric for measuring the discrepancy between the yielded $T_{near\_balanced}$ and the optimal $T_{balanced}$ structures. We will additionally show in Section 4.2.1 that constructing $T_{near\_balanced}$ with the ETC algorithm yields an error of 11% on average for the topologies utilized in this paper.

3 The ETC Algorithm

In this section, we describe the discovery and balancing phases of the ETC algorithm whose objective is to transform T_{input} into a near-balanced tree T_{ETC} in a distributed manner.

3.1 ETC Phase 1: Discovery

The first phase of the ETC algorithm starts out by having each node select one node as its parent using the FHF approach. During this phase, each node also records its local depth (i.e., $depth(s_i)$) from the sink. Notice that $depth(s_i)$ can be determined based on a *hops* parameter that is included inside the tree construction request message. In particular, the hops parameter is initialized to zero and is incremented each time the tree construction request is forwarded to the children nodes of some node.

A node s_i also maintains a child node list *children* and an alternate parent list *APL*. The APL list is constructed locally at each sensor by *snooping* (i.e., monitoring the radio channel while other nodes transmit and recording neighboring nodes) and comes at no extra cost. Such a list could also be utilized to find alternate parents in cases of failures.

The sink then queries the network for the total number of sensors n and the maximum depth of the routing tree d. Such a query can be completed with a message complexity of $O(n)$. When variables n and d are received, the sink calculates the Optimal Branching Factor (β).

3.2 ETC Phase 2: Balancing

The second phase of the ETC algorithm involves the top-down reorganization of the query routing tree T_{input} such that this tree becomes near-balanced. In particular, the sink disseminates the β value to the n nodes using the reverse acquisition tree. When a node s_i receives the β value from its parent s_p it initiates the execution of Algorithm 1 in which s_i will order parent re-assignments for its children. The presented algorithm is divided into two main steps: i) lines 3-8: s_i's connection to its newly assigned parent *newParent*; and ii) lines 9-25: the transmission of parent reassignment messages to children nodes, in which the given nodes are instructed to change their parent.

In line 2 of Algorithm 1, each node s_i ($\forall s_i \in S - s_0$) waits in blocking mode until an incoming message interrupts the *receive*() command. When such a message has arrived, s_i obtains the β value and the identifier of its *newParent*. The next objective (line 4) is to identify whether *newParent* is equal to NULL, in which case s_i does not need to change its own parent (i.e., we proceed to line 9). On the contrary, if *newParent* has a specific node identifier then s_i will attempt to connect to that given node (lines 4-8). Notice that if *newParent* can not accommodate the connect request from s_i then the procedure has to be repeated until completion or until the alternative parents are exhausted.

In line 9 we proceed to the second step of the algorithm in which s_i's children might be instructed to change their parent node. We choose to do such a reassignment at s_i, rather than at the individual child s_j, because s_i can more efficiently eliminate duplicate parent assignments (i.e., two arbitrary children of s_i will both not choose *newParent*). In line 10 we skip s_i if the number of children is less than β. In the contrary case (line 14), we have to eliminate $|children(s_i)| - \beta$ children from s_i. Thus, we iterate through the child list of s_i (line 16) and attempt to identify a child s_j that has at least one alternate parent (line 17). If an alternative parent can not be determined for node s_j then it obviously not meaningful to request a change of s_i's parent (line 22).

Let us now simulate the execution of the ETC algorithm using the illustration of Figure 1. In particular, Figure 1 (left) displays $n = 10$ sensors arranged in an ad-hoc topology T_{input} with a depth $d = 2$. In order to transform T_{input} into a near-balanced topology each node has to obtain approximately $\beta = 3.16$ children (i.e., $\sqrt[2]{10}$). To simplify our discussion, but w.l.o.g., let us assume that the only sensors with multiple entries in their alternate parent list (APL) are s_8 and s_{10}. In particular, assume that we have the following values: APL(s_8)={s_3} and APL(s_{10})={s_3}.

The ETC algorithm is initiated at the sink node s_0. Since s_0 has less than $\beta = 3.16$ children it transmits β and `newParent=NULL` to its only child s_1. Similarly, s_1 transmits β and `newParent=NULL` to its children s_2, s_3 and s_4. Let us now consider s_2 which receives the above parameters in line 2 of Algorithm 4. Since `newParent =NULL`, s_2 does not need to change its parent (lines 3-8).

Algorithm 1 : ETC balancing algorithm

Input: A node s_i and its children (i.e., $children(s_i)$); The alternate parent list for each child of s_i (i.e., $APL(s_j)$, where $s_j \in children(s_i)$); The Optimal Branching Factor β; The new parent s_i should select (denoted as $newparent(s_i)$).
Output: A Near-Balanced Query Routing Tree T_{CETC}.
Execute these steps beginning at s_0 (top-down):

```
1: procedure $Balance\_Tree(s_i;\ children(s_i);\ \forall_{s_j \in children(s_i)} APL(s_j);\ )$
2:     $(\beta, newParent)$=receive(); ▷ Get info from parent.
3:     ▷ Step 1: Connect to new parent if needed
4:     while ($newParent$ != $NULL$) do
5:         if (!connect($newParent$)) then
6:             newParent = getNewParent(parent($s_i$)) ).
7:         end if
8:     end while
9:     ▷ Step 2: Adjust the parent of the children nodes.
10:    if ($|children(s_i)| <= \beta$) then                ▷ Skip $s_i$.
11:        for $j = 1$ to $|children(s_i)|$ do
12:            send($\beta, NULL, s_j$);        ▷ Send $\beta$ to child.
13:        end for
14:    else▷ Ask $|children(s_i)| - \beta$ nodes to change their parent.
15:        while ($|children(s_i)| > \beta$) do
16:            $s_j = getNext(children(s_i))$;
17:            if ($|APL(s_j)| > 1$) then
18:                $newParent$=$AlternParent(APL(s_j), s_i)$;
19:                send($\beta, newParent, s_j$);    ▷ Send to $s_j$.
20:                $children(s_i) = children(s_i)$ - $s_j$        ▷
21:            else
22:                send($\beta, NULL, s_j$);
23:            end if
24:        end while
25:    end if
26: end procedure
```

It has to however instruct some of its children to change their parents as `|children(`s_2`)|>`β. Thus, it processes its children nodes in sequential order, starting at s_5 and ending at s_{10}, instructing some of them to change their parent. In particular, s_{5-8} are instructed to retain their initial parent while s_8 and s_{10} are instructed to change their parent to s_3 (i.e., they receive the messages `send(`$3.16, s_3,\ s_8$`)` and `send(`$3.16, s_3, s_{10}$`)` respectively. In our example s_3 can accommodate s_8's and s_{10}'s request as `|children (`s_3`)|=0`. Under different conditions however, satisfying such requests might not be possible. Thus, each node might request from its parent another alternative parent (i.e., lines 5-7). The updated near-balanced tree T_{ETC} is presented in Figure 1 (right).

3.3 Experimental Methodology

Datasets: We utilize a three realistic traces in our experiments:

i) **Intel54:** Sensor readings that are collected from 54 sensors deployed at the premises of the Intel Research in Berkeley [2] between February 28th and April 5th, 2004. The sensors utilized in the deployment were equipped with weather boards and collected time-stamped topology information along with humidity, temperature, light and voltage values once every 30 seconds (i.e., the epoch). The dataset includes 2.3 million readings collected from these sensors.

ii) **GDI140:** This is a medium-scale dataset from the habitat monitoring project deployed in 2002 on the Great Duck Island which is 15km off the coast of Maine [8], USA.

iii) **Intel540:** This is a set of 540 sensors which is randomly derived from the of Intel54 dataset. In particular, we randomly replicate nodes from the Intel54 dataset until we get the defined sample.

Sensing Device: We use the energy model of Crossbow's research TelosB [1] sensor device to validate our ideas. TelosB is an ultra-low power wireless sensor equipped with a 8 MHz MSP430 core, 1MB of external flash storage, and a 250Kbps RF Transceiver that consumes 23mA in receive mode (Rx), 19.5mA in transmit mode (Tx), $7.8mA$ in active mode (MCU active) with the radio off and $5.1\mu A$ in sleep mode. Our performance measure is *Energy*, in *Joules*, that is required at each discrete time instance on to resolve the query. The energy formula is as following: $Energy(Joules) = Volts \times Amperes \times Seconds$.

4 Experimental Evaluation Results

To assess the efficiency of the algorithms presented in this paper we have conducted two series of experiments. In the first series we have conducted a micro-benchmark on the CC2420 radio transceiver in order to quantify the transmission inefficiencies in a real setting. In the second series we have studied the balancing error and the energy consumption of the ETC algorithm.

4.1 Micro-benchmarks

In the first experimental series we have conduct a micro-benchmark on the CC2420 radio chip (using the TelosB [1] energy model in TOSSIM) to justify why data transmission inefficiencies have to be optimized in current data acquisition systems. In particular, we

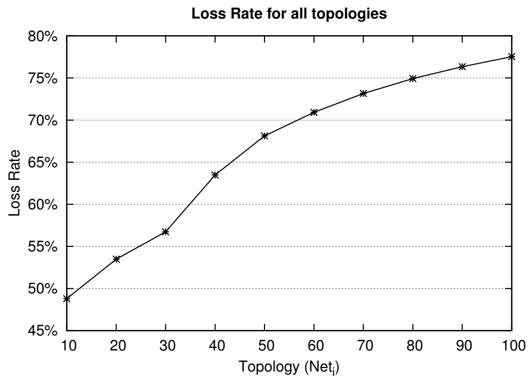


Figure 2. Micro-benchmarks using the CC2420 communication module.

justify why a sensor network should minimize the number of hub nodes (i.e., nodes with several children) as these increase collisions during data transmission and thus also increase energy consumption.

For this purpose we construct ten star topologies with 10 to 100 nodes respectively (i.e., $\{Net_i : 1 \leq i \leq 10, |Net_i| = 10 * i\}$), and evaluate the loss rate when all nodes attempt to transmit data packets to a given sink node. In particular, each node attempts to transmit a 16-byte packet to a given sink node for 60 seconds (that accounts to approximately 250 messages in our setting). Since this experiment requires a large number of sensors we utilized the TOSSIM environment along with its LossyBuilder module that created "lossy" radio models for each topology.

For each topology Net_i we measure: i) the *Total Packets Sent* from all sensors to s_0 (denoted as P_i^T) and ii) the *Total Packets Received* from s_0 (denoted as P_i^0). We next evaluate each topology's loss rate by using the formula: $LossRate(Net_i) = 1 - (\frac{P_i^0}{P_i^T})$

Figure 2 illustrates the loss rate for the ten presented topologies. We can observe an almost linear increase in the loss rate with a 77% packet loss when 100 nodes transmit concurrently to a single parent. Consequently, many data packets did not reach their designated destination in the first attempt and had to be re-transmitted (the energy cost will be documented in the subsequent experiments). While somebody might argue that one hundred nodes will not transmit concurrently in a given geographic region, our results indicate that the high loss rate applies even to smaller numbers. In particular, even when a node has 10 children then the loss rate is as high as 48%. The ETC algorithm presented in this paper distributes the children of overloaded nodes to neighboring nodes and different wake-up times decreasing in that way data transmission collisions.

4.2 Evaluation of the ETC algorithm

In the second experimental series we assess the efficiency of the ETC algorithm. We start out by assessing the construction quality of the ETC algorithm and then proceed with an energy evaluation of our algorithm.

4.2.1 Measuring the Balancing Error

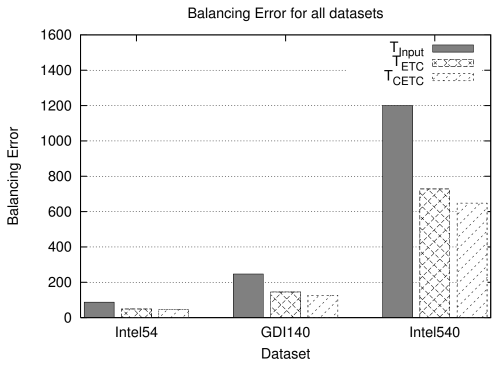
Our first objective is to measure the quality of the tree, with regards to the balancing factor, that is generated by the ETC algorithm. Thus, we measure the balancing error of the generated trees. The Balancing_Error of a query routing tree is defined as follows:

$$Balancing\_Error(T_{CETC}) := \sum_{i=0}^{n} |\beta - \sum_{j=0}^{n} PM_{ij}|$$

where $\beta = \sqrt[d]{n}$ and $PM_{ij} = 1$ denotes that node i is a parent of node j and $PM_{ij} = 0$ the opposite. Notice that this table is fragmented vertically but can be obtained easily with a message complexity of O(n), where each message has a size of $O(n^2)$ in the worst case.

For this experiment we generated one query routing tree per dataset using the three described algorithms: i) The First-Heard-From approach, which constructs an ad-hoc spanning tree T_{input} without any specific properties; ii) The CETC algorithm, which transforms T_{input} into the best possible near-balanced tree T_{CETC} in a centralized manner using global knowledge; and iii) The ETC algorithm, which transforms T_{input} into a near-balanced tree T_{ETC} in a distributed manner.

Figure 3 (left), presents the results of our evaluation which demonstrates the following properties: i) All three approaches feature some balancing error, which indicates that in all cases it is not feasible to construct a fully balanced tree $T_{balanced}$. This is attributed to the inherent structure of the sensor network where certain nodes are not within communication radius from other nodes. ii) The second observation is that the FHF approach has the worst Balancing_Error, which is an indicator that FHF can rarely produce any proper balanced topology and that increases data transmission collisions and energy consumption (shown in next experiment). In particular, the balancing error of the FHF approach is on average 91% larger than the respective error for the CETC algorithm; iii) The third and most important observation is that the distributed ETC algorithm is only 11% less accurate than the centralized CETC algorithm. Therefore, even though the ETC algorithm does not utilize any global knowledge, it is still able to create a near-balanced topology in a distributed manner.



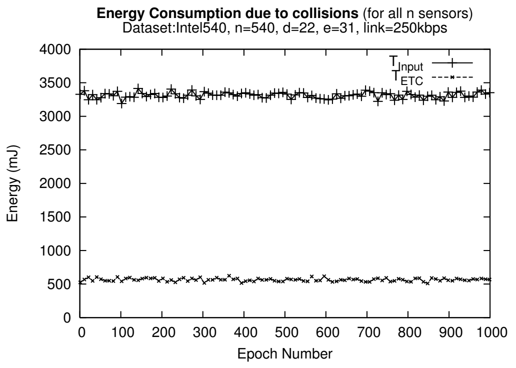


Figure 3. Left: Measuring the Balancing Error of the FHF (T_{input}), CETC (T_{CETC}) and ETC (T_{ETC}) algorithms; Right: Energy Consumption due to re-transmissions in an unbalanced topology (T_{input}) and in a near-balanced topology (T_{ETC})

4.2.2 Energy Consumption of ETC

In order to translate the effects of the Balancing_Error into an energy cost, we conduct another experiment using the Intel540 dataset. Specifically, we generate two query routing trees: a) T_{input}, constructed using the First-Heard-From approach, and b) T_{ETC} constructed using the ETC algorithm. We configure our testbed to measure the energy required for re-transmissions due to collisions in order to accurately capture the additional cost of having an unbalanced topology.

Figure 3 (right) displays the energy consumption of the two structures. We observe that the energy required for re-transmissions using T_{input} is 3,314±50mJ. On the other hand, T_{ETC} requires only 566±22mJ which translates to additional energy savings of 83%. The reason why T_{ETC} presents such great additional savings is due to the re-structuring of the query routing tree into a near balanced query routing tree which ensures that data transmissions collisions are decreased to a minimum.

5 Conclusions

In this paper we have present ETC, a distributed algorithm for balancing sensor network query routing trees to minimize collisions and prolong the longevity of a sensor network. Our experimentation with real micro-benchmarks and trace-driven experimentation shows that ETC offers significant power reductions under a variety of conditions. In the future we plan to study how these ideas can be incorporated into existing data acquisition frameworks.

Acknowledgements: This work was supported by the Open Univ. of Cyprus under project SenseView, the US NSF under project AQSIOS (#IIS-0534531), the EU under projects mPower (#034707) and IPAC (#224395).

References

[1] Crossbow Technology, Inc. http://www.xbow.com/

[2] Intel Lab Data http://db.csail.mit.edu/labdata/labdata.html

[3] Kim S., Pakzad S., Culler D., Demmel J., Fenves G., Glaser S., Turon M., “Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks”, In *ACM IPSN*, 2007.

[4] Madden S.R., Franklin M.J., Hellerstein J.M., Hong W., ”The Design of an Acquisitional Query Processor for Sensor Networks”, In *SIGMOD*, 2003.

[5] Madden S.R., Franklin M.J., Hellerstein J.M., Hong W., “TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks”, In *USENIX OSDI*, 2002.

[6] Murty R.N., Mainland G., Rose I., Chowdhury A.R., Gosain A., Bers J., and Welsh M., “CitySense: An Urban-Scale Wireless Sensor Network and Testbed”, In *IEEE HST*, 2008.

[7] Sadler C., Zhang P., Martonosi M., Lyon S., “Hardware Design Experiences in ZebraNet”, In *ACM SenSys*, 2004.

[8] Szewczyk R., Mainwaring A., Polastre J., Anderson J., Culler D., “An Analysis of a Large Scale Habitat Monitoring Application”, In *ACM SenSys*, 2004.

[9] Yao Y., Gehrke J.E., ”The cougar approach to in-network query processing in sensor networks”, In *SIGMOD Record*, Vol.32, No.3, pp.9-18, 2002.

[10] Zeinalipour-Yazti D., Andreou P., Chrysanthis P.K., Samaras G., “MINT Views: Materialized In-Network Top-k Views in Sensor Networks”, In *MDM*, 2007.

[11] Zeinalipour-Yazti D., Lin S., Kalogeraki V., Gunopulos D., Najjar W., “MicroHash: An Efficient Index Structure for Flash-Based Sensor Devices”, In *USENIX FAST*, 2005.