**Notice of Violation of IEEE Publication Principles**

**"Latency Optimized Workload-based Query Routing Tree Algorithm in Wireless Sensor Networks,"**
by Yingchi Mao, Xiaofang Li, Yi Liang,
in the Proceedings of the 2nd International Conference on Information Science and Engineering (ICISE), 2010, December 2010, pp.2159-2162

After careful and considered review of the content and authorship of this paper by a duly constituted expert committee, this paper has been found to be in violation of IEEE's Publication Principles.

This paper was found to be a near verbatim copy of the paper cited below. The original text was copied without attribution (including appropriate references to the original author(s) and/or paper title) and without permission.

Due to the nature of this violation, reasonable effort should be made to remove all past references to this paper, and future references should be made to the following article:

**"ETC: Energy-driven Tree Construction in Woreless Sensor Networks"**
by P. Andreou, A. Pamboris, D. Zeinalipour-Yazti, P.K. Chrysanthis, and G. Samaras,
in the Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, May 2009, pp. 513-518

# Latency Optimized Workload-based Query Routing Tree Algorithm in Wireless Sensor Networks

Mao Yingchi, Li Xiaofang
College of Computer and Information
Hohai University
Nanjing, China
yingchimao@hhu.edu.cn

Liang Yi
Research and Development Center
State Grid Electric Power Research Institute
Nanjing, China
liangy@naritech.cn

*Abstract*— **Constructing an effective Query Routing Tree is the premise for continuous queries in a Wireless Sensor Networks (WSN). The query routing tree structures can provide sensors with a path to the querying nodes. At present, the data acquisition systems for WSN construct the routing structures in an ad-hoc manner, 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 addition, if the path of query routing is too long, it seriously suffers from the increased data delivery delay. The high end-to-end delay is not acceptable in the delay-constrained applications. In this paper, we present a data collection timing model for query results acquisition. Based on the timing model, we propose a delay-optimized and workload-based query routing tree construction algorithm, which balances the workload among nodes and optimizes the data delivery delay, thus reducing energy consumption and the data delivery time in the course of data acquisition. The simulation experiments from Intel Research illustrate that the proposed query routing tree algorithm can significantly reduce energy consumption under a variety of conditions and prolong the lifetime of a wireless sensor network.**

***Keywords-Querying Routing Tree, Workload-based, Latnecy optimization, Wireless Sensor Networks***

## I. INTRODUCTION

Large-scale deployments of WSN 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 that provides each sensor with a path over which query results can be transmitted to the querying node. However, current methods of constructing a query tree are in ad-hoc manner, there is no guarantee that a given query workload will be distributed equally among all sensor nodes. That leads to data collision and energy waste.

To facilitate our description, Figure 1 illustrates the initial ad-hoc query routing tree created on top of a 10-node sensor network with the First-Heard-From (FHF) approach. 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, node s2 is inflicted with a high workload (i.e., 4 child nodes) while other nodes at the same level (i.e., s3 and s4), only have one child nodes, respectively. Notice that both s8 and s9 are within communication range from s3 and s4 (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 into account as it conducts the child-to-parent assignment in a network-agnostic manner.

Therefore, unbalanced workload topologies pose some important energy challenges. (1) Decreased network lifetime and coverage. (2) Increased data transmission collisions.

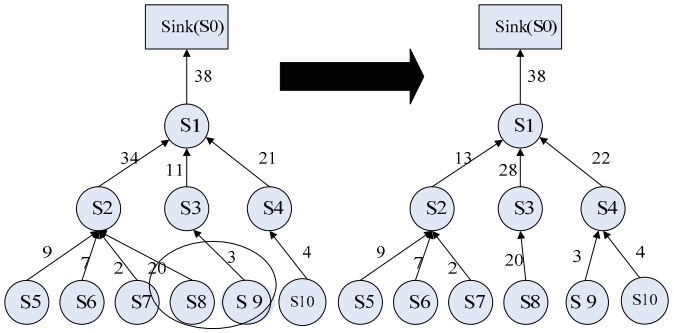


Figure 1. Left: The ad-hoc query routing tree; Right: The optimized workload-based query routing tree constructed using the DWQRT algorithm

In addition, if the path of query routing is too long, it seriously suffers from the increased data delivery delay. The high end-to-end delay is not acceptable in the delay-constrained applications. It is necessary to optimize the waiting time for data delivery from the children within the stipulated time bound. We present a timing model to define how long a

978-1-4244- 7618-3 /10/$26.00 ©2010 IEEE

In this paper, a delay-optimized workload-based query routing tree construction (DWQRT) algorithm for constructing an arbitrary query routing tree into a near-balanced query routing tree is proposed. DWQRT can balance the workload among nodes and optimize the delivery delay to reduce the energy consumption. Finally, the simulation experiments validate the efficiency of DWQRT.

## II. PRELIMINARIES

In this section, we will overview of balanced trees in order to better understand the DWQRT algorithm. 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.

**Definition 1**: Balanced Tree ($\mathbf{T}_{\text{balanced}}$) *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:

$T1 = (V, E) = (\{A, B, C, D\}, \{(B, A), (C, A), (D, B)\})$

where the pairs in the ***E*** set represent the edges of the binary tree. By visualizing $T1$, 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 $T1$ as a balanced tree.

Notice that ***V*** has several balanced tree representations of the same height (e.g., the directed tree

$T2 = (\{A, B, C, D\}, \{(B, A), (C, A), (D, A)\})$

Similarly, ***V*** has also many balanced tree representations of different heights. e.g., the directed tree

$T3 = (\{A, B, C, D\}, \{(B, A), (C, B), (D, C)\})$ which has a height of one rather than two.

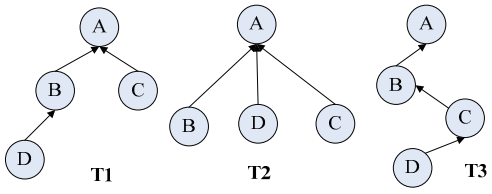


Figure 2. Illustration of Balance Tree

Finally, in a balanced tree every node has approximately $\beta$ children, where $\beta$ is equal to $\sqrt[d]{n}$ (the depth of every balanced tree is $d = \log_\beta n$). The DWQRT algorithm presented in this section focuses on the subset of balanced trees which have the same height to $\mathbf{T}_{input}$ as this makes the construction process more efficient. In order to derive a balanced tree ($\mathbf{T}_{balanced}$) in a centralized manner, we could utilize the respective balancing algorithms of AVL Trees, B-Trees and Red-Black Trees. However, they assume that all nodes are within communication range from each other which is not realistic in WSN. Thus, the DWQRT algorithm seeks to construct a *Near-Balanced Tree* ($\mathbf{T}_{near\_balanced}$), defined as follows:

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

The objective of $\mathbf{T}_{near\_balanced}$ is to yield a structure similar to $\mathbf{T}_{balanced}$ 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 $\mathbf{T}_{near\_balanced}$ and the optimal $\mathbf{T}_{balanced}$ structures.

## III. THE TIMING MODEL

In the course of query routing, each node calculates the initial time for data collection, based on the hop distance from the sink and the number of children it has in its sub tree rooted from it.

Let $M_i$ is the middle node $S_i$ and $L_j$ is the leaf node $S_j$. Let $Neigh(S_i)$ is the number of children nodes for the middle node $M_i$. i.e $Neigh(S_i) = degree(M_i)$. The path cost from the from any leaf node $L_j$ to a middle node $M_i$ is given by

$$path\_cost(L_j, M_i) = \sum_{Mk \in N} degree(M_k)$$

Where $N$ is the set of middle nodes in the path from $L_j$ to $M_i$. The maximum path cost from any leaf node to a middle node $P_i$ is the maximum path cost from leaf node to the middle.

$$P_i = \max\{Path\_cost(L_j, M_i)\} \quad \forall j$$

The maximum path cost from any node to sink $P_{\text{sink}}$ is the maximum path cost from leaf node to the middle.

$$P_{\text{sink}} = \max\{Path\_cost(L_j, \text{sink})\} \quad \forall j$$

The initial staggered time out is calculated as follows. Let $T_i$ is the stagger time out for the middle node $S_i$, which is equal to $T_i = T_{ci} + T_{mi}$. where $T_{ci}$ is the cascading time out, which depends on the level in which the middle is in. This gives the initial timeout as in cascade time out for each node and it is same for all the nodes in the same level. $T_{mi}$ is the collection time out of the node, which depends on the number of children it has.

$$T_{ci} = 2(T - (T_{TD} * hop))$$

$$T_{Mi} = (P_i / P_{\text{sink}}) * (T - T_{TD} * depth)$$

where *hop* denotes the hop distance of the node $M_i$, *depth* is the depth of the tree, $T$ is the query generation period and $T_{TD}$ is the one hop delay between the levels.

## IV. THE DWQRT ALGORITHM

### A. DWQRT Phase 1: Discovery

The first phase of the DWQRT 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., $depth(si)$) from the sink. Notice that $depth(si)$ 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 $si$ 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 $(\beta)$.

### B. DWQRT Phase 2: Balancing

The second phase of the DWQRT algorithm reorganizes of the query routing tree $\mathbf{T}_{input}$ such that this tree becomes near-balanced. In particular, the sink disseminates the $\beta$ value to the $n$ nodes using the reverse acquisition tree. When a node $si$ receives the $\beta$ value from its parent $sp$, it initiates the execution of DWQRT balancing algorithm in which $si$ will order parent re-assignments for its children. The balancing algorithm includes two main steps: the first step is node si's connection to its newly assigned parent *newParent*; and the second step is the transmission of parent reassignment messages to children nodes, in which the given nodes are instructed to change their parent.

In the first step, each node si ( $\forall si \in S - s_0$ ) waits in blocking mode until an incoming message interrupts the *receive()* command. When such a message has arrived, $si$ obtains the $\beta$ value and the identifier of its *newParent*. If its *newParent* is equal to NULL, node $si$ does not need to change its own parent. On the contrary, if *newParent* has a specific node identifier then node $si$ will attempt to connect to that given node. Notice that if *newParent* can not accommodate the connect request from $si$ then the procedure has to be repeated until completion or until the alternative parents are exhausted.

In the second step, node si's children might be instructed to change their parent node. We choose to do such a reassignment at $si$, rather than at the individual child $sj$, because $si$ can more efficiently eliminate duplicate parent assignments (i.e., two arbitrary children of $si$ will both not choose *newParent*). if the number of children is less than $\beta$, node $si$ need not do anything. In the contrary, node $si$ eliminates $|children(si)| - \beta$ children from $si$. Thus, the algorithm iterates through the child list of $si$ and attempts to identify a child $sj$ that has at least one alternate parent. If an alternative parent can not be determined for node $sj$ then it obviously not meaningful to request a change of si's parent.

**Algorithm 1: DWQRT balancing algorithm**

**Input:** A node $si$ and its children (i.e., *childen(si)*); The alternate parent list for each child of $si$ (i.e., *APL(sj)*, where $sj \in children(si)$ ); The optimal Branching Factor $\beta$; The new parent $si$ should select (denoted as *newparent(si)*).
**Output:** A Near-Balanced Query Routing Tree $\mathbf{T}_{near\_balanced}$
**Execute these steps beginning at** $s_0$ **(top-down):**

```
1: procedure Balance_Tree(si; children(si); ∀sj ∈ children(si) APL(sj);)
2:    (β, newParent) = receive(); ▷ Get info from parent.
3:    ▷ Step1: Connet to new parent if needed
4:    while (newParent != NULL) do
5:       if (! connect(newParent)) then
6:          newParent = getNewParent(parent(si))
7:       end if
8:    end while
9:    ▷ Step2: Adjust the parent of the children nodes.
10:   if (|children(si)|<=β) then
11:      for j=1 to |children(si)| do
12:         send(β, NULL, sj);
13:      end for
14:   else ▷ Ask |children(si)|-β nodes to change their parent.
15:   while (|children(si)|>β) do
16:      sj = getNext(children(si));
17:      if (|APL(sj)|>1) then
18:         newParent = AlternParent(APL(sj), si);
19:         send(β, newParent, sj);
20:         children(si) = children(si) − sj;
21:      else
22:         send(β, NULL, sj);
23:      end if
24:   end while
25: end if
26: end procedure
```

## V. Experimental Evaluation Results

### A. Datasets

*1) 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.

*2) 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.

### B. Energy Model

The energy model of Crossbow's research TelosB [1] sensor device is used to validate our ideas. Our performance measure is Energy, in Joules, that is required at each discrete time instance on to resolve the query.

### C. 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 DWQRT algorithm. Thus, we measure the balancing error of the generated trees. The Balancing Error of a query routing tree is defined as follows:

$$Balancing\_Error(\mathrm{T}near\_balance) = \sum_{i=0}^{n} \left| \beta - \sum_{j=0}^{n} PMij \right|$$

where $\beta = \sqrt[d]{n}$ and $PMij = 1$ denotes that node $i$ is a parent of node $j$ and $PMij = 0$ the opposite.

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 $\mathbf{T}_{input}$ without any specific properties; ii) The DWQRT algorithm, which transforms $\mathbf{T}_{input}$ into the best possible near-balanced tree $\mathbf{T}_{near\_balanced}$ in a centralized manner; and iii) The DWQRT algorithm, which transforms Tinput into a near-balanced tree $\mathbf{T}_{near\_balanced}$ in a distributed manner.

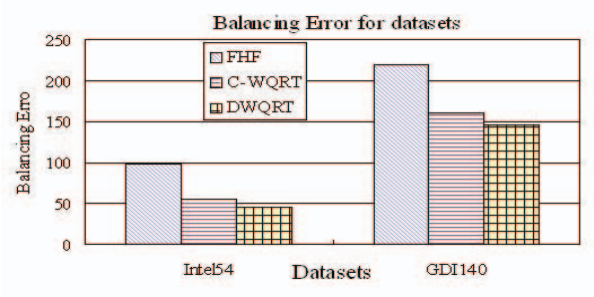


Figure 3. Measuring the Balancing Error

As shown in Figure 3, we can see that: i) All three approaches feature some balancing error, which indicates that in all cases it is not feasible to construct a fully balanced tree $\mathbf{T}_{balanced}$. This is attributed to the inherent structure of the sensor network where certain nodes are not within communication radius from other nodes. ii) The 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 DWQRT algorithm; iii)The distributed DWQRT algorithm is only 11% less accurate than the centralized DWQRT algorithm. Therefore, even though the distributed DWQRT algorithm does not utilize any global knowledge, it is still able to create a near-balanced topology in a distributed manner.

### *D. Energy Consumption of DWQRT*

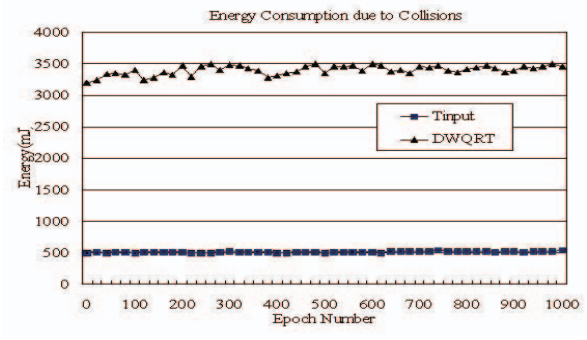


Figure 4. Energy Consumption $\mathbf{T}_{input}$ and DWQRT ($\mathbf{T}_{near\_balanced}$).

In order to translate the effects of balancing the querying routing tree into an energy cost, we conduct the experiment using the Intel54 dataset. Specifically, we generate two query routing trees: *i)* $\mathbf{T}_{input}$, constructed using the First-Heard-From approach, and *ii)* $\mathbf{T}_{near\_balanced}$ constructed using the DWQRT algorithm. We will measure the energy required for re-transmissions due to collisions in order to accurately capture the additional cost of having an unbalanced topology.Figure 4 displays the energy consumption of the two structures. We can observe that the energy required for re-transmissions using $\mathbf{T}_{input}$ is much more than $\mathbf{T}_{near\_balanced}$ requires. The reason why DWQRT 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.

## VI. CONCLUSIONS

In this paper, a delay optimized workload-based query routing tree construction algorithm, DWQRT is presented, which can balances the workload among nodes and optimizes the delivery delay, thus reducing energy consumption in the course of data acquisition. The simulation experiments from Intel Research illustrate that the proposed DWQRT can significantly reduce energy consumption under a variety of conditions and prolong the lifetime of a wireless sensor network.

## ACKNOWLEDGMENT

This research is partly supported in part by the Nature Science Fund of Hohai University under grant No.2008428911, and the Fundamental Research Funds for the Central Universities under grant No. 2009B20714.

## 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 UsingWireless Sensor Networks”, In ACM IPSN, 2007.

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

[5] Madden S.R., Franklin M.J., Hellerstein J.M., HongW., 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 Sen-Sys, 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 innetwork 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.