Notice of Violation of IEEE Publication Principles

"Workload-Based Query Routing Tree Algorithm in Wireless Sensor Networks,"
by Yingchi Mao, Xiaofang Li, Yi Liang,
in the 2010 International Conference on Computational Intelligence and Software Engineering (CiSE), December 2010, pp.1-4

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

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 this paper we present a Workload-based Query Routing Tree construction (WQRT) algorithm, which balances the workload among nodes and minimizes the data collisions, thus reducing energy consumption in the course of data acquisition. The simulation experiments from Intel Research illustrate that the proposed WQRT 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, Query, Wireless Sensor Networks

I. INTRODUCTION

A wireless sensor network (WSN) is composed of a large number of low-cost and low-energy sensor nodes that communicate with each other through multi-hop wireless links. A basic operation of WSN is data gathering and querying. Because sensor nodes are often deployed in remote or inaccessible environments, where replenishing sensor's energy is usually impossible, an important issue is conserving energy and maximizing sensor lifetime [1].

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 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. Since the majority of the energy consumption is transmitting 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), sensor s2's energy will be depleted faster than s3 and s4. In addition, if s2's energy is depleted and no alternate parents are available for sensors s5, s6, and s7, then the coverage quality of the network will be reduced dramatically. (2) Increased data transmission collisions. An unbalanced workload topology increases data transmission collisions which results in a major source of energy waste in wireless communication.

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

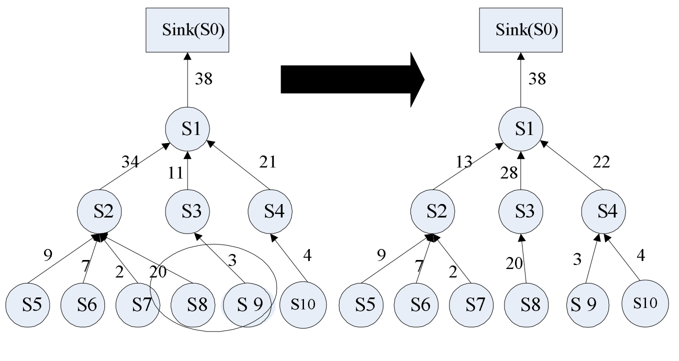

Figure 1. Left: The ad-hoc query routing tree; Right: The optimized workload-based query routing tree constructed using the WQRT algorithm

II. PRELIMINARIES

In this section, we will overview of balanced trees in order to better understand the WQRT 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 $\boldsymbol{E}$ set represent the edges of the binary tree. By visualizing $T1$, we observe that the subtrees of $\boldsymbol{A}$ differ by at most one $(i.e., |height(B) - height(C)| = 1)$ and that the subtrees of $\boldsymbol{B}$ differ again by at most one

$(i.e., |height(D) - height(NULL)| = 1)$.

Thus, we can characterize $T1$ as a balanced tree.

Notice that $\boldsymbol{V}$ 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, $\boldsymbol{V}$ 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.

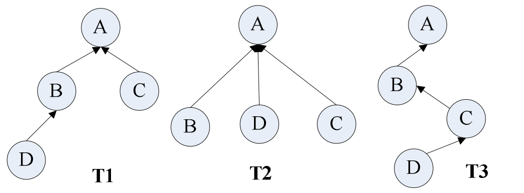

Figure 2. Illustration of Balance Tree

Finally, in a balanced tree every node has approximately β children, where β is equal to $\sqrt[d]{n}$ (the depth of every balanced tree is $d = \log_\beta n$). The WQRT 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 WQRT 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* β .

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 WQRT ALGORITHM

In this section, the discovery and balancing phases of the WQRT algorithm will be discussed. The objective of balance phases is to transform $\mathbf{T}_{input}$ into a near-balanced tree $\mathbf{T}_{near\_balanced}$ in a distributed manner.

A. WQRT Phase 1: Discovery

The first phase of the WQRT 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 (β) .

B. WQRT Phase 2: Balancing

The second phase of the WQRT algorithm reorganizes of the query routing tree $\mathbf{T}_{input}$ 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 si receives the β value from its parent sp, it initiates the execution of WQRT 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 β 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 β, node si need not do anything. In the contrary, node si eliminates |*children(si)*| − β 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: WQRT balancing algorithm

Input: A node si and its children (i.e., $children(si)$); The alternate parent list for each child of si (i.e., $APL(sj)$, where $sj \in children(si)$); The optimal Branching Factor β; 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 $Balance\_Tree(si; children(si); \forall sj \in children(si) APL(sj);)$
2:     $(\beta, newParent)$ = 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 (|$children(si)$|<=$\beta$) then
11:        for $j$=1 to |$children(si)$| do
12:            send($\beta, NULL, sj$);
13:        end for
14:    else ▷ Ask |$children(si)$|-$\beta$ nodes to change their parent.
15:    while (|$children(si)$|>$\beta$) do
16:        $sj = getNext(children(si))$;
17:        if (|APL($sj$)|>1) then
18:            $newParent = AlternParent(APL(sj), si)$;
19:            send($\beta, newParent, sj$);
20:            $children(si) = children(si) - sj$;
21:        else
22:            send($\beta, NULL, sj$);
23:        end if
24:    end while
25: end if
26: end procedure
```

IV. Experimental Evaluation Results

To assess the efficiency of the WQRT algorithm, we have studied the balancing error and the energy consumption of the WQRT algorithm.

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. The dataset includes 2.3 million readings collected from these sensors.

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. The energy formula is as following:

Energy(Joules) = Volts × Amperes × Seconds.

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 WQRT 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 WQRT algorithm, which transforms $\mathbf{T}_{input}$ into the best possible near-balanced tree $\mathbf{T}_{near\_balanced}$ in a centralized manner; and iii) The WQRT algorithm, which transforms Tinput into a near-balanced tree $\mathbf{T}_{near\_balanced}$ in a distributed manner.

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 WQRT algorithm; iii)The distributed WQRT algorithm is only 11% less accurate than the centralized WQRT algorithm. Therefore, even though the distributed WQRT algorithm does not utilize any global knowledge, it is still able to create a near-balanced topology in a distributed manner.

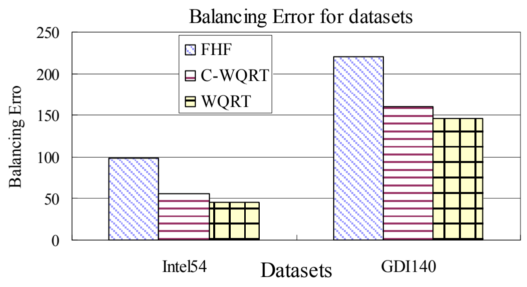

Figure 3. Measuring the Balancing Error of the FHF ($\mathbf{T}_{\text{input}}$), C-WQRT ($\mathbf{T}_{\text{balanced}}$) and WQRT ($\mathbf{T}_{\text{near\_balanced}}$)

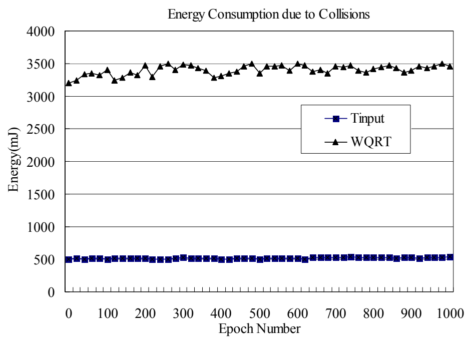

Figure 4. Energy Consumption due to re-transmissions in an unbalanced topology ($\mathbf{T}_{\text{input}}$) and in a near-balanced topology WQRT ($\mathbf{T}_{\text{near\_balanced}}$).

D. Energy Consumption of WQRT

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 WQRT 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 WQRT 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.

V. CONCLUSIONS

In this paper, a workload-based query routing tree construction algorithm, WQRT is presented, which can balances the workload among nodes and minimizes the data collisions, thus reducing energy consumption in the course of data acquisition. The simulation experiments from Intel Research illustrate that the proposed WQRT 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.