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.

![Image](image-url)

### 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 ($T_{\text{balanced}}$) 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:

$T_1 = (V, E) = (\{A, B, C, D\}, \{B, A\}, \{C, A\}, \{D, B\})$

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., $|\text{height}(B) - \text{height}(C)| = 1$) and that the subtrees of $B$ differ again by at most one (i.e., $|\text{height}(D) - \text{height}(\text{NULL})| = 1$).

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 $T_2 = (\{A, B, C, \text{D}\}, \{(B, A), (C, A), (D, D)\})$

Similarly, $V$ has also many balanced tree representations of different heights. e.g., the directed tree $T_3 = (\{A, B, C, D\}, \{(B, A), (C, B), (D, C)\})$ which has a height of one rather than two.

![Image](image-url)

### 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 $T_{\text{input}}$ into a near-balanced tree $T_{\text{near-balanced}}$ 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 FH 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. WQRT Phase 2: Balancing

The second phase of the WQRT algorithm reorganizes of the query routing tree \( T_{input} \) 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 \( s_i \) receives the \( \beta \) value from its parent \( sp \), it initiates the execution of WQRT balancing algorithm in which \( s_i \) will order parent re-assignments for its children. The balancing algorithm includes two main steps: the first step is node \( s_i \)'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 \( s_i \) ( \( \forall s_i \in S - s_0 \) ) waits in blocking mode until an incoming message interrupts the \texttt{receive()} command. When such a message has arrived, \( s_i \) obtains the \( \beta \) value and the identifier of its \texttt{newParent}. If its \texttt{newParent} is equal to NULL, node \( s_i \) does not need to change its own parent. On the contrary, if \texttt{newParent} has a specific node identifier then node \( s_i \) will attempt to connect to that given node. Notice that if \texttt{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 the second step, node \( s_i \)'s children might be instructed to change their parent node. We choose to do such a realignment 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 \texttt{newParent}). If the number of children is less than \( \beta \), node \( s_i \) need not do anything. In the contrary, node \( s_i \) eliminates \( |\text{children}(s_i)| - \beta \) children from \( s_i \). Thus, the algorithm iterates through the child list of \( s_i \) and attempts to identify a child \( s_j \) that has at least one alternate parent. 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.

Algorithm 1: WQRT balancing algorithm

Input: A node \( s_i \) and its children (i.e., \( \text{children}(s_i) \) ); The alternate parent list for each child of \( s_i \) (i.e., \( \text{APL}(s_j) \) , where \( s_j \in \text{children}(s_i) \) ); The optimal Branching Factor \( \beta \); The new parent \( s_i \) should select (denoted as \texttt{newparent}(s_i)).

Output: A Near-Balanced Query Routing Tree \( T_{near\_balanced} \)

Execute these steps beginning at \( s_0 \) (top-down):

1: \textbf{procedure} \\
2: \hspace{1em} Balance\_Tree(s_i, \text{children}(s_i), \beta, \text{children}(\text{APL}(s_j))) \hspace{1em} \textbf{procedure} \\
3: \hspace{4em} \triangleright \text{Step1: Connet to new parent if needed} \\
4: \hspace{4em} \triangleright \text{Step2: Adjust the parent of the children nodes.} \\
5: \hspace{4em} \triangleright \text{Step3: Get info from parent.} \\
6: \hspace{4em} \triangleright \text{Step4: if (} \text{connect(newParent)} \text{) then} \\
7: \hspace{4em} \triangleright \text{Step5: newParent = getNewParent(parent(s_i))} \\
8: \hspace{4em} \triangleright \text{Step6: if } (\text{children(s_i)} <= \beta) \text{ then} \\
9: \hspace{4em} \triangleright \text{Step7: end if} \\
10: \hspace{4em} \triangleright \text{Step8: end while} \\
11: \hspace{1em} \textbf{end procedure} \\
12: \hspace{1em} \textbf{for } j=1 \text{ to } |\text{children(s_i)}| \text{ do} \\
13: \hspace{4em} \textbf{send}(\beta, \text{NULL}, s_j); \\
14: \hspace{4em} \textbf{end for} \\
15: \hspace{4em} \textbf{else } \triangleright \text{ Ask } |\text{children(s_i)}| - \beta \text{ nodes to change their parent.} \\
16: \hspace{4em} \textbf{while } (|\text{children(s_i)}| > \beta) \text{ do} \\
17: \hspace{4em} \hspace{4em} \textbf{send}(\beta, \text{newParent}, s_j); \\
18: \hspace{4em} \hspace{4em} \textbf{end while} \\
19: \hspace{4em} \textbf{end if} \\
20: \hspace{4em} \textbf{end for} \\
21: \hspace{4em} \textbf{end for} \\
22: \hspace{4em} \textbf{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 to resolve the query. The energy formula is as following:

\[
\text{Energy(Joules)} = \text{Volts} \times \text{Amperes} \times \text{Seconds}.
\]

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:

\[
\text{Balancing\_Error}(T_{near\_balance}) = \sum_{j=1}^{B} \beta - \sum_{j=1}^{PMij} PMij
\]

where \( \beta = \sqrt{n} \) 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 $T_{\text{input}}$ without any specific properties; ii) The WQRT algorithm, which transforms $T_{\text{input}}$ into the best possible near-balanced tree $T_{\text{near,balanced}}$ in a centralized manner; and iii) The WQRT algorithm, which transforms $T_{\text{input}}$ into a near-balanced tree $T_{\text{near,balanced}}$ 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 $T_{\text{balanced}}$. 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 4 displays the energy consumption of the two structures. We can observe that the energy required for re-transmissions using $T_{\text{input}}$ is much more than $T_{\text{near,balanced}}$ 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