A Query-Based Routing Tree in Sensor Networks

In Chul Song Yohan Roh Dongjoon Hyun Myoung Ho Kim

Division of Computer Science
Department of Electrical Engineering & Computer Science
Korea Advanced Institute of Science and Technology (KAIST)
Daejeon, Korea
{icsong, yhroh, djhyun, mhkim}@dbserver.kaist.ac.kr

ABSTRACT

In sensor networks, energy efficiency has been a primary concern because most sensor nodes have limited power. A well-known technique to reduce energy consumption is in-network processing, which reduces the message cost through partial aggregation and merging of messages. Routing trees used for message transmissions affect the amount of in-network processing. The efficiency of a routing tree varies depending on the continuous query being executed. However, query-specific cost reduction has not been much considered in constructing routing trees. Although there is some work on a query-based routing tree, only a limited type of queries, i.e., aggregation queries with the group-by clause has been considered. In this paper, we propose a query-based routing tree, called the MD-tree. It is separately constructed for each continuous query with the goal of increasing the amount of in-network processing. First, we formally define the MD-tree for a given continuous query, using a measure called the minimum distance. Roughly speaking, the minimum distance of a node indicates how far the node is from some other node that generates a message. Next, we describe how to construct MD-trees in sensor networks. MD-trees are constructed in such a way that messages generated from sensor nodes can be merged more often and earlier. Our experimental results show that MD-trees outperform existing routing trees in terms of the number of message transmissions for data collection.

Categories and Subject Descriptors

H.2.4 [**Systems**]: Distributed Databases, Query Processing; C.2.2 [**Network Protocols**]: Routing Protocols

General Terms

Algorithms, Design

Keywords

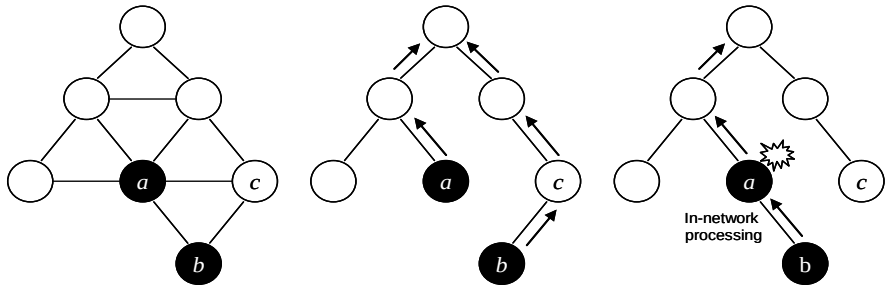
Sensor Networks, In-network Processing, Routing Trees

1. INTRODUCTION

The advent of wireless sensor nodes gives us opportunities unattainable before [3][4]. They can be deployed in many places including buildings, manufacturing plants, and habitats, and can provide timely and accurate information about environmental conditions. There are many applications of sensor networks: builders can detect damage in buildings [7]; engineers in industrial plants can identify machines or processes that need repairing [1]; biologists can trace moving of animals [2]. Wireless sensor nodes communicate each other via wireless multi-hop networking to deliver sensed values to the base station, where users request queries and receive the results of those queries. Because typical sensor network applications use continuous queries that periodically execute and report sensed values for their lifetimes, routing trees are commonly constructed for efficient collection of sensed values.

Energy efficiency has been a primary concern in sensor networks because most sensor nodes have limited power. If used without care, they will deplete their power in only a few days [5]. It is known that message communication among sensor nodes is a main source of energy consumption. Typically, wireless communication consumes 1,000 to 10,000 times more energy than computation [10]. In-network processing is an important technique to reduce energy consumption in sensor network applications [6][9]. The main idea of in-network processing is to reduce volumes of data in the network by partially aggregating sensed values or merging intermediate messages. For aggregation queries (e.g., MAX, COUNT, etc.), an intermediate node may not forward values received from its children. Instead, it may aggregate them and send only a newly computed value. For example, for a MAX query, an intermediate node forwards only the maximum value among the values received from its children. For other queries, an intermediate node may merge multiple messages into one message to reduce the number of messages.

Routing trees used for message transmissions affect the amount of in-network processing. The efficiency of a routing tree varies depending on the query being executed. For example, Figure 1 shows a wireless network of seven sensor nodes and two different routing trees constructed from it. In Figure 1-(a), an edge between two nodes indicates that they can communicate each other. Consider a query, “Report the maximum temperature reading on the second floor in a building every one minute.” Black-colored nodes in the figure are assumed to be on the second floor, and hence will send their sensed values. In Figure 1-(b) and (c), arrows indicate message transmissions for this query. As in the figure, five messages are used in Routing Tree I, whereas only three messages are enough in Routing tree II. This is because the sensed values of nodes *a* and *b* can be merged into one message in Routing tree II. Suppose that another query is requested, and this time nodes *b* and *c* generate messages for the query. Then, Routing tree I will be more beneficial than Routing tree II for this query. As shown in this example, the efficiency of a routing tree



(a) A network of seven sensor nodes (b) Routing tree I (c) Routing tree II

Figure 1. Two different routing trees for a graph of sensor nodes

varies depending on the query being executed. In existing routing trees [6][9], however, query-specific cost reduction has not been much considered. Although [8] discusses a query-based routing tree, it considers only a limited type of queries: aggregation queries with the group-by clause.

In this paper, we propose a query-based routing tree, called the minimum distance tree (MD-tree) that is separately constructed for each query by utilizing the query information. The main objective of the MD-tree is to increase the amount of in-network processing. The MD-tree is constructed in such a way that messages generated from sensor nodes can be merged more often and earlier.

The remainder of this paper is organized as follows. Section 2 discusses the background and related work. Section 3 formally defines the MD-tree and describes how to construct MD-trees in sensor networks. Experimental evaluation of MD-trees is presented in Section 4, and Section 5 concludes the paper.

2. BACKGROUND AND RELATED WORK

Sensor networks can be modeled as a distributed database [5][6][9]. Each node generates tuples for the distributed table, named *sensors*. Users of sensor networks can request queries using an SQL-like query language. Consider the following query:

```
SELECT nodeid, temp, humidity
  FROM sensors
  WHERE temp > threshold
  SAMPLE PERIOD 10 sec
  LIFETIME 30 days
```

This query specifies that each node should report its own identifier (nodeid), temperature (temp), and humidity readings that exceed a user-specified *threshold*, once per 10 seconds for 30 days.

Query processing in sensor networks consists of two phases: the query dissemination phase and result collection phase. In the query dissemination phase, a query is sent from the root node to all the nodes in the network. In the result collection phase, each node reads values from its sensors once per the sample period specified in the query and combines them into a tuple; and if the tuple satisfies the conditions of the query, it generates a message to deliver the tuple to the base station. A routing tree may already exist before this query is posed, or may be constructed during the query dissemination phase for efficient collection of sensed values from sensor nodes in the result collection phase.

Consider Figure 1-(b) and (c) again. Let the level of the root node be zero, the level of the children of the root node be one, and so on. In-network processing proceeds level-by-level starting from the nodes at the highest level, toward the root node. When the nodes at level l are sending messages, the nodes at level l–1 are listening. Before generating messages, each node performs in-network processing: That is, partially aggregates its tuple together with those from its child nodes, or combines them into one message.

Although much work on message routing in ad hoc networks and sensor networks has been made, construction of efficient routing trees has been an important issue in sensor network applications. Routing trees can be classified into either query-independent routing trees, or query-based routing trees, depending on utilization of query information in routing tree construction.

A query-independent routing tree (QIRT) [6][9] is constructed without considering specific queries. In QIRT construction, the routing message floods from the root node down the network. Each node selects the first node it heard from as its parent node. Once constructed, one QIRT is used for all queries.

On the other hand, a query-based routing tree (QBRT) is separately constructed for each query. A semantic routing tree (SRT) [5] is a routing tree used in query dissemination to route a query to the nodes that have a possibility to generate tuples for the query. By sending a query only to the nodes that need to receive the query, the SRT can reduce communication cost in query dissemination. The group-aware network configuration (GaNC) method [8] constructs a routing tree used in result collection of aggregation queries with the group-by clause. The method divides the nodes in the network into separate groups based on the group information in a query, and arranges them in such a way that the nodes in the same group are aligned along the same path. Messages sent from this setting will contain fewer groups: Hence they can be combined into one short message. One disadvantage of the GaNC method is that it can be used only for aggregation queries with the group-by clause. Thus, for other queries it operates in the same way as in QIRTs.

In the next section, we propose a query-based routing tree, called the minimum tree (MD-tree). While the primary goal of SRTs is to reduce query dissemination cost in the query dissemination phase, that of MD-trees is to reduce communication cost in the result collection phase. The goal of the GaNC method is to increase the amount of data reduction whenever in-network processing takes place, whereas that of MD-trees is to make in-network processing itself occur more often and earlier. The combination of their work and ours may further reduce volumes of data in sensor networks.

3. MINIMUM DISTANCE TREES

In this section, we define the minimum distance tree (MD-tree) and describe how to construct MD-trees in sensor networks. We design the MD-tree with the following two goals in mind:

1. in-network processing should occur as *many* as possible,
2. in-network processing should occur as *early* as possible.

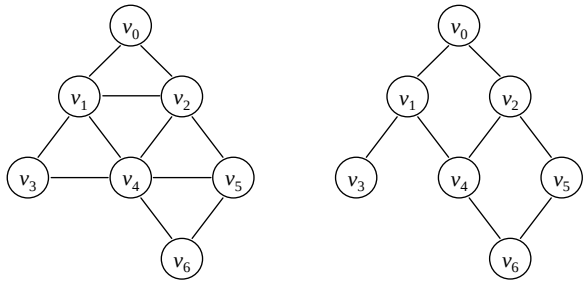
It is clear that the more in-network processing takes place, the more it reduces volumes of data. Early in-network processing also results in more data reduction than late in-network processing. For example, consider two different routing trees of the same height. Assume in both routing trees that two different nodes at level l generate messages containing the tuples that satisfy the conditions of a query, and no other nodes in the network. If in one routing tree those messages are immediately merged at level l–1 (i.e., when the two nodes are siblings under the same parent node), the total messages will be 2+(l–1): 2 message transmissions from each node to the parent node at level l–1, and l–1 message transmissions from the parent node to the root node. In another routing tree, on the other hand, if those messages are never merged on the way to the root node, the total messages will be 2l. The difference in message transmissions of these two routing trees is about l. Thus, early in-network processing is important, especially in large sensor networks where the height of a routing tree is high.

3.1 Definition

We model a sensor network as an undirected graph $G = (V, E)$ where V is a set of nodes and E is a set of edges. There is a distinguished node v_0, called the *root node*, which is an ordinary sensor node or possibly can be a base station. The root node and all other sensor nodes are members of V. An edge (v_i,v_j) is in E if two nodes v_i and v_j can communicate each other. Figure 2-(a) shows a graph for a sensor network with seven nodes.

The *distance* from v_i to v_j in graph G for a sensor network, denoted by $d_G(v_i,v_j)$, is the length of a path from v_i to v_j with the minimum number of edges. The distance from the root node v_0 to v_i is simply called the "distance of v_i" and is denoted by $d_G(v_i)$. $d_G(v_0) = 0$ by default.

Let $G = (V,E)$ be a graph for a sensor network. A *stratified routing graph* $G_S = (V, E')$ of G is a subgraph of G, where an edge $(v_i,v_j) \in E$ is in E' if $|d_G(v_i) - d_G(v_j)| = 1$. Figure 2-(a) and (b) show a graph and the stratified routing graph derived from it, respectively.



(a) (b)

Figure 2. A graph and its stratified routing graph

Next, we define a dependence relationship between two nodes. Let $G_S = (V, E)$ be a stratified routing graph. We define a relation $\rightarrow_{G_S}$ on V by

$$\rightarrow_{G_S} = \{<v_i,v_j> \mid (v_i,v_j) \in E \text{ and } d_{G_S}(v_j) < d_{G_S}(v_i)\} .$$

If a pair $<v_i,v_j>$ is in $\rightarrow_{G_S}$, we write $v_i \rightarrow_{G_S} v_j$. We will omit the subscript G_S in relation $\rightarrow_{G_S}$ when the context is clear. In other words, we will commonly use $v_i \rightarrow v_j$ rather than $v_i \rightarrow_{G_S} v_j$ if there is no ambiguity. The *transitive closure* of $\rightarrow$ is denoted by $\rightarrow^+$. We say that v_i *depends* on v_j if $v_i \rightarrow^+ v_j$ and v_i *directly depends* on v_j if $v_i \rightarrow v_j$. In Figure 2-(b), every node depends on v_0, that is, $v_i \rightarrow^+ v_0$, $1 \le i \le 6$. v_6 depends on v_0, v_1, v_2, v_4, and v_5, whereas it directly depends on v_4 and v_5. There is no dependence relationship between v_3 and v_6. The dependence relationship $v_i \rightarrow^+ v_j$ indicates that a path from nodes v_i to v_j, through which messages are forwarded toward the root node in the result collection phase, can be established in a routing tree for a sensor network.

Given a query, a node is called a *query node* if it satisfies the query qualification, i.e., the conditions in the WHERE clause of the query. Additionally, the root node will be considered a query node for every query regardless of satisfying the qualification of the query. This is only for the convenience of defining the minimum distance of a node described later. For example, consider the following query that reports, every 30 seconds, the maximum temperature reading on the second floor in a building for 30 days:

```
SELECT nodeid, MAX(temp)
  FROM sensors
  WHERE floor = 2
  SAMPLE PERIOD 30 sec
  LIFETIME 30 days
```

For this query, query nodes are the nodes on the second floor and the root node.

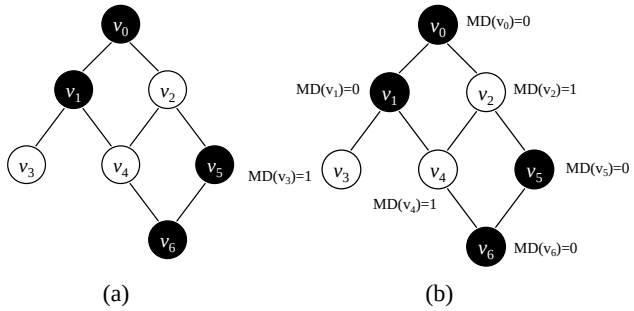


Figure 3. Query nodes and minimum distances

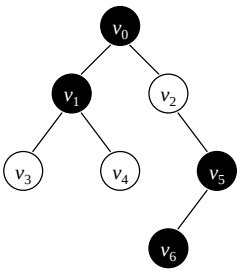


Figure 4. An MD-tree

The minimum distance of node v_i for query q in stratified routing graph $G_S = (V,E)$, denoted by $MD_{q,G_S}(v_i)$ is defined as follows:

(1) $MD_{q,G_S}(v_i) = 0$, if node v_i is a query node for query q,

(2) $MD_{q,G_S}(v_i) = min\{d_{G_S}(v_i,v_j) \mid v_j$ is a query node for query q and $v_i \rightarrow^+ v_j\}$, otherwise.

For convenience, we will use $MD(v_i)$ instead of $MD_{q,G_S}(v_i)$ if there is no ambiguity. $MD(v_i)$ represents the distance to the closest query node on which v_i depends.

For example, Figure 3-(a) shows the query nodes for some query in the stratified routing graph in Figure 2-(b), and Figure 3-(b) shows the minimum distance of each node. In Figure 3-(b), $MD(v_0)$, $MD(v_1)$, $MD(v_5)$, and $MD(v_6)$ are zero because they are query nodes. $MD(v_4)$ is 1 because it is not a query node, and the distance to the closest query node on which it depends, i.e., v_1 is 1.

Finally, we define the MD-tree. The MD-tree for query q in stratified routing graph $G_S = (V,E)$, denoted by $MDT_{G_S}(q) = (V, E')$, is a spanning tree of G_S, where

$E' = \{(v_i,v_j) \mid$ for all v_j such that $v_i \rightarrow v_j$, $MD_{G_S}(v_j,q)$ is the smallest; if there are two or more such v_j, arbitrary one of (v_i,v_j) remains in $E'\}$.

Figure 4 shows the MD-tree constructed from the stratified routing graph in Figure 3.

Discussion For simplicity of explanation, we have only considered the type of queries with a restricted syntax. More general form of queries on the sensor network, commonly addressed in the literature, may have the following form:

```
SELECT list of attributes and aggregates
  FROM sensors
  WHERE list of conditions
  SAMPLE PERIOD sample-period
  LIFETIME lifetime
```

For example, consider the following query:

```
SELECT nodeid, MAX(temp)
  FROM sensors
  WHERE floor = 2 AND temp > 20
  SAMPLE PERIOD 30 sec
  LIFETIME 30 days
```

This query will be executed in any node located on the second floor of the building. However, only the nodes whose temperature readings are greater than 20 will report their sensed values (i.e., generate messages). Because temperature readings in a node may change from time to time, a node on the second floor may or may not generate messages at certain times.

A condition such as "floor = 2" is called a *static condition* in the sense that if a node satisfies this condition at the first execution of a query, it will always satisfy this condition during the lifetime of the query. A condition such as "temperature > 20" is called a *dynamic condition* in the sense that even though a node satisfies this condition at some time, it may not satisfy this condition at another time. Because queries may have a dynamic condition in general, we redefine the definition of the query node as follows: *a node is called a query node for a query if it satisfies all the static conditions of the query*.

3.2 MD-tree Construction

In this section, we describe how to construct MD-trees in sensor networks. The MD-tree construction process consists of the following two steps:

- *Stratified routing graph (SRG) construction step*: In the SRG construction step, the stratified routing graph for a sensor network is constructed. In this step, every node discovers the nodes it directly depends on.
- *Query dissemination and parent selection (QP) step*: Given a query, in the QP step, the MD-tree for the query is constructed.

In what follows, we describe each step in detail.

3.2.1 Stratified routing graph construction step

In this step, to construct the stratified routing graph for a sensor network, the *distance determination (DD) message* floods from the root node down the network. The DD message format is as follows:

<*src_id*, *dist*>,

where *src_id* is the sender identifier, and *dist* is the distance of the sender from the root node.

As the DD message floods, each node discovers the nodes it directly depends on and records the identifiers of these nodes in its *dependent set D*.

This step proceeds as follows.

1. The root node prepares the DD message. It records, in the message, its identifier to *src_id*, its distance (the distance of the root node is 0) to *dist*, and broadcasts the message.
2. When a node receives the DD message for the first time, it sets its distance to *dist* in the message plus one and adds *src_id* to its dependent set *D*. Then, it broadcasts the message with its identifier and distance.
3. When a node receives the DD message, but it has already decided its distance, there are three cases to consider: 1) If *dist* in the message is equal to its distance minus one, then it adds *src_id* to *D*; 2) If *dist* in the message is less than its distance minus one, then it first empties its dependent set *D*. Next, it sets its distance to *dist* in the message plus one and adds *src_id* to *D*. It also broadcasts the message with its identifier and distance; 3) If *dist* in the message is greater than or equal to its distance, it ignores the message.
4. This process is repeated until all the nodes in the network decide their distances and dependent sets.

Figure 5 shows how the SRG construction step proceeds in a sensor network with seven nodes. In the figure, arrows indicate DD message transmissions, and the number over an arrow indicates the value of the *dist* field in the DD message. The message transmissions ignored in the receivers are not shown. The dependent set of each node is presented next to it.

In the figure, the root node v_0 first prepares the DD message, records its identifier and distance (which is zero) in it, and broadcasts the message. Note that the dependent set of the root node is empty because there are no nodes on which it directly depends. Node v_1 first receives the DD message from the root node v_0. It sets its distance to the distance of the root node plus one and adds node v_0 to its dependent set. It also broadcasts the DD message with its identifier and distance. Later, node v_1 ignores messages from its other neighbors—node v_2, v_3, and v_4 because the distances of those nodes are greater than or equal to its distance. Node v_4 first receives the DD message from node v_1. It sets its distance to 2, adds node v_1 to its dependent set, and broadcasts the DD message with its identifier and distance. Next time it receives the DD message from node v_2, it adds node v_2 to its dependent set because the distance of node v_2 is one less than its distance. In this way, every node in the network discovers the nodes it directly depends on.

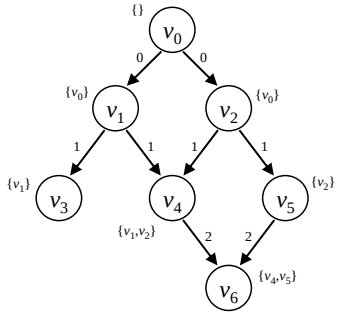


Figure 5. An SRG construction step

To reflect the change of network topology, we can initiate the SRG construction step periodically, or whenever we receive less than a user-specified number of tuples within the specified sample period, as suggested in [9].

3.2.2 Query dissemination and parent selection step

When a user requests a query, the MD-tree for the query is constructed through the query dissemination and parent selection (QP) step. In this step, a *query message* containing query information and the minimum distance of a sender floods from the root node down the network. The format of query messages is as follows:

<*src_id*, *md*, *query*>,

where *src_id* is the sender identifier, *md* is the minimum distance of the sender, and *query* is the query information that contains the query identifier, query, and so on.

As the query message for query *q* floods, each node maintains its *minimum distance set* MD_q for query *q*, to keep track of the minimum distances of the nodes on which it directly depends. The minimum distance set of a node contains pairs of the node identifier, on which it directly depends, and the minimum distance of the node, i.e., (node identifier, minimum distance). After a node selects its parent for a query, it records pair (query identifier, parent node identifier) in its *parent set P*.

This step proceeds as follows.

1. When a user requests query *q*, the root node prepares the query message for query *q*. It records, in the message, its identifier to *src_id*, its minimum distance (which is 0) to *md*, and the query information to *query*, and broadcasts the message.
2. When a node receives the query message for query *q*, there are two cases to consider: 1) If the sender identifier is in its dependent set *D*, it adds pair (*src_id*, *md*) to its minimum distance set MD_q for query *q*; 2) Otherwise, it ignores the message.
3. When a node has received the query messages for query *q* from all of the nodes on which it directly depends, i.e., from all the nodes in its dependent set *D*, it selects its

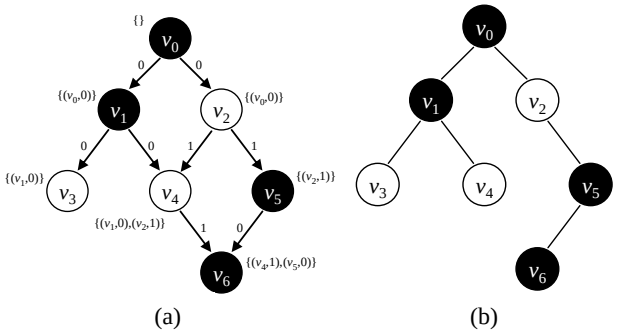


Figure 6. A QP step for query *q* and the MD-tree for the query

parent. It selects, as its parent node *p*, the node whose minimum distance is the smallest among the nodes on which it directly depends. If there are two or more nodes that have the smallest minimum distance, then it randomly chooses one as its parent node. In addition, it adds pair (*q*, *p*) to its parent set *P*. It next calculates its minimum distance. If it is a query node for query *q*, it sets its minimum distance to 0. If not, it sets its minimum distance to the minimum distance of its parent node plus one. Finally, it broadcasts the query message with its identifier, minimum distance, and the query information.

4. This process is repeated until all the nodes in the network determine their minimum distances and select their parent nodes.

Figure 6 shows how the QP step for query *q* proceeds after the SRG construction step in Figure 5. In the figure, black-colored nodes are query nodes for the query. In Figure 6-(a), arrows indicate query message transmissions, and the number over an arrow indicates the value of the *md* field in the query message. The message transmissions ignored in the receivers are not shown. The minimum distance set of each node is presented next to it. Figure 6-(b) shows the constructed MD-tree for query *q*.

In Figure 6-(a), the root node v_0 first prepares the query message for query *q*. It records, in the message, its identifier, minimum distance (which is zero), and the query information, and broadcasts the message. When node v_1 receives the query message from node v_0 on which it directly depends on, it adds a pair of the identifier and distance of node v_0, i.e., (v_0,0) to its minimum distance set. Now that it has received from all the nodes on which it directly depends, it selects node v_0 as its parent node and adds pair (q,v_0) to its parent set. Next, node v_1 calculates its minimum distance. Because it is a query node for query *q*, it sets its minimum distance to 0. Finally, node v_1 broadcasts the query message with its identifier, minimum distance, and the query information. When node v_4 first receives the query message from node v_1, it adds pair (v_1,0) to its minimum distance set. Next time node v_4 receives the query message from node v_2, it adds (v_2,1) to its minimum distance set. Now that it has received the query messages from all the nodes on which it directly depends, it selects its parent node. Node v_1 becomes the parent node of node v_4, because its minimum distance is the smallest. Node v_4 adds pair (q,v_1) to its parent set. Next, it calculates its minimum distance. Because it is not a query node for query *q*, its minimum distance is the minimum distance of the parent node v_2 plus one, i.e., 1. Finally, node v_4 broadcasts the query message with its identifier, minimum distance, and the query information. In this way, the MD-tree for query *q* is constructed.

When it is time for a node to send a message for a query, the node finds the parent node for the query in its parent set and transmits the message to the parent node. Note that a node may have different parent nodes for different queries.

4. PERFORMANCE EVALUATION

In this section, we evaluate the benefit of MD-trees over existing routing trees in various simulation environments. Because SRTs are used in query dissemination, and the GaNC method can be used only for aggregation queries with the group-by clause, we target query-independent routing trees (QIRTs) in performance evaluation. The metric employed in the evaluation is the total number of message transmissions required for one result collection of a query.

4.1 Settings and Assumptions

In various simulation experiments, sensor nodes are randomly deployed in a sensor network. A sensor network is of size width *w* and height *h*, and both are set to the same values in the simulations. Given communication range *r* and node density *d* (the number of nodes within the communication range), the number of nodes *N* to be deployed in a sensor network is $\left\lfloor \frac{d}{\pi \times r^2} \times w \times h \right\rfloor$. The selectivity of a query indicates the percentage of the query nodes for the query in a sensor network. We place the base station at the center of a sensor network. Table 1 summarizes the default values for the parameters used in the simulations. In all the experiments, we have executed the simulation 10 times and computed the average of those results.

Table 1. Default parameters

Parameter	Default value
Communication range (m)	30
Node density	15
Network size (m^2)	600×600
Query selectivity (%)	30

In the evaluation, we assume that wireless communication is lossless: A node successfully receives all the messages from other nodes. In addition, we assume that all queries have only static conditions: Thus, all query nodes for a query generate messages. Finally, we assume that, in the result collection phase, each node performs in-network processing and transmits one message.

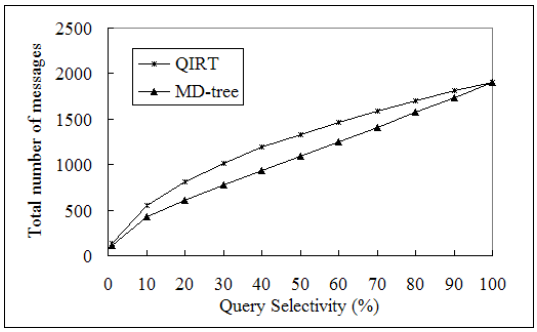


Figure 7. Query selectivities

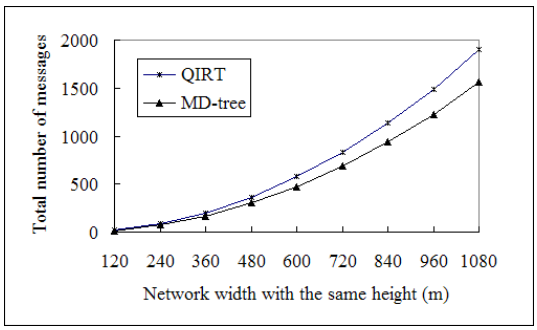


Figure 8. Network size

4.2 Performance of Various Query Selectivities

We vary the query selectivity from 1% to 100% to evaluate the effect of various query selectivities on the benefit of MD-trees over QIRTs. Figure 7 shows the simulation results. In the figure, when the query selectivity is low (below 10%), the benefit of MD-trees is small. This is because a small number of nodes are the query nodes for a query, hence few messages are generated in the network. As the query selectivity increases, the benefit of MD-trees also increases. As the query selectivity approaches 100%, however, the benefit again decreases. This is because all the nodes in the network generate messages: Thus, in-network processing occurs at almost every node in both routing trees.

Overall, MD-trees outperform QIRTs in various query selectivities, with at maximum 25% reduction of message transmissions.

4.3 Performance of Various Network Size

In this experiment, we change the network size from 120m×120m to 1080m×1080m to evaluate the effect of various network size on the benefit of MD-trees over QIRTs. Figure 8 shows the experimental results. In small size networks, the benefit of MD-trees is small because there are a small number of nodes in the network. However, as the network size increases, the benefit of MD-trees also increases.

Note that the benefit of MD-trees converges to about 18%, that is, no longer increases, starting from network size 600m×600m. At first, we expected that more frequent and earlier in-network processing in MD-trees gains more benefit in large sensor networks: Because in general the larger a sensor network, the higher the height of a routing tree. However, this is not the case as the experimental results show. The reason is that in large sensor networks, even in QIRTs, messages from sensor nodes are merged within a few hops, rather than transferred up to the base station without being merged.

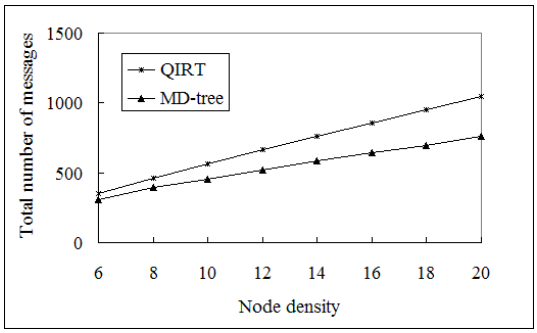


Figure 9. Node density

Overall, MD-trees show better performance over QIRTs in various network size, with at maximum 18% reduction of message transmissions.

4.4 Performance of Various Node Density

We investigate the effect of various node density on the benefit of MD-trees over QIRTs. We varied the node density from 6 to 20. Figure 9 shows the experimental results. As in the figure, the benefit of MD-trees increases as the node density increases.

The maximum benefit of MD-trees is 27% at node density 20. With additional experiments, we verified that the benefit continues to increase as the node density increases.

5. CONCLUSIONS

In this paper, we have proposed a query-based routing tree, called the minimum distance tree (MD-tree). We have designed the MD-tree in such a way that in-network processing occurs as many as and as early as possible in result collection. We also have defined the MD-tree for a given query using a measure, called the minimum distance, and presented how to construct MD-trees in sensor networks. Experimental evaluation shows that MD-trees outperform existing routing trees in various environments, in terms of the number of message transmissions for result collection.

6. ACKNOWLEDGMENTS

This work was supported by grant No. (R01-2006-000-10809-0) from the Basic Research Program of the Korea Science & Engineering Foundation.

7. REFERENCES

[1] D. Abadi, S. Madden, and W. Lindner. REED: robust, efficient filtering and event detection in sensor networks. In *Proc. of VLDB Conf.*, 2005.

[2] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: Application driver for wireless communications technology. In *ACM SIGCOMM Workshop*

on Data Communications in Latin America and the Caribbean, 2001.

[3] J. Gehrke and S. Madden. Query Processing in Sensor Networks. *Pervasive Computing*, Jan.-Mar., 2004.

[4] J. Hill and D. Culler. Mica: a wireless platform for deeply embedded networks, *IEEE Micro* 22(6):12-24, 2002.

[5] S. Madden, M. Franklin, and J. Hellerstein. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. on Database Systems*, Vol. 30, No. 1, 122-173, 2005.

[6] S. Madden, M. Frankln, J. Hellerstein, and W. Hong. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proc. of OSDI*, 2002.

[7] D. Pescovitz. Smart Buildings Admit Their Faults. Lab Notes, http://www.coe.berkeley.edu/labnotes/1101smartbuildings.html.

[8] M. Sharaf, J. Beaver, A. Labrinidis, and P. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *The VLDB Journal*, Vol. 13, No. 4, 384-403, 2004.

[9] Y. Yao and J. Gehrke. Query processing for sensor networks. In *Proc. of CIDR Conf*., 2003.

[10] F. Zhao and L. Guibas. *Wireless Sensor Networks*. Morgan Kaufmann, 2004.