Query processing Algorithm in Wireless Sensor Networks

SungSuk Kim , and Sun Ok Yang
Computer Science & Engineering Seokyeong University
Electrical & Electronic Engineering Yonsei University
sskim03@skuniv.ac.kr, soyang@korea.ac.kr

Abstract

Sensors have a function to gather environmental data in sensor networks. They operate by limited, small-size battery depending on applications. Thus there are several kinds of research efforts to make better use of local energy. In this paper, we propose multi-attribute query processing algorithm (Section Bit based scheme (SB)). In the initial setup, routing path is determined based on parent-child relationship by two phases and some partial information (section bits) about the sensing data of their descendant nodes are delivered to the parent in second phase (parent selection phase); that is, an attribute is divided into the same, small size, N sections and one bit is used to each section. By using a small volume of local information, it is possible to protect nodes from propagating messages who does not become the help during query processing. Finally, we measure energy efficiency by counting the number of messages. Experimental results show the improved energy efficiency of the proposed algorithm.

1. Introduction

Sensor networks continue to attract significant interest in a wide range of research communities and industry engineers. Generally, each sensor consists of a small node with sensing, computing, and communication capabilities. It is usually battery-operated and has a very limited computing capability. The gathered data is transmitted through a multihop communication route to a centralized sensor node (i.e., sink node), which needs much more energy consumption compared with a local computation work. Thus, there are several works to lessen the number of message transmissions [1].

One of the efforts is in-Network evaluation or aggregation [2]. For example, aggregate queries (such as COUNT, MAX, and SUM) over the sensor data are formulated using a simple SQL-like language, then distributed across the network. Finally, aggregated results are sent back to the sink node. It can reduce the cost of data collection and network traffic but the application scope may be restricted.

In [3], authors try to make use of Skyline operator, which is known as the maximum vector problem. The Skyline is defined as those points which are not dominated by any other point. A Point dominates another point if it is as good or better in all dimensions and better in at least one dimension. Thus, due to the operator, database system can process queries by just searching only some portions of whole databases. The best example is that a man is looking for a hotel cheap and close to the beach. Top-k query processing algorithm (*SPEERTO*) in P2P systems utilizes a threshold based super-peer selection mechanism based on the skyline points of each super-peer [4]. Relying on a threshold scheme, SPEERTO returns the exact results while the number of joined super-peers and transferred data is minimized. That work, however, assumes that the whole set of points fits into memory and that this set is not the result of a query (e.g., a join).

That is, the works using skyline operator are not directly applicable to query processing in

sensor networks but it gives us a motivation during developing an algorithm. If a sensor node maintains some partial information about nearby neighbor local sensing data, the number of unnecessary query deliveries may decrease; if the size of the information is too large or local data has to be exchanged too frequently, the maintaining cost sets off the benefits. Therefore, we have to make a balance between them.

In our previous work [5], we have proposed a spatial query processing algorithm in sensor networks. Basically, routing path is set up among all sensor nodes during initial phase. At that time, parent nodes get the approximated locations range of their children nodes. By using the partial information, each sensor can determine whether it has to join the query processing or not.

In this paper, we will develop the previous works to treat multi-attribute query processing. During initial setup phase, child node delivers the range of sensing data of both itself and its descendent nodes. To make the volume of the information small, we propose *Section-Bit based scheme(SB)*. Each attribute has maximum/minimum value and the interval between them is divided into N equal sections. Thus, for descendent nodes attribute, only N bits have to be delivered. Although local sensing data has changed compared with the previous one, no message is issued if the difference is less than section size. In this way, sensor nodes can maintain partial sensing data in their descendent nodes.

Our contributions can be summarized as follows:

- By using partial information about descendent nodes, lots of unnecessary messages can be reduced. Through experiments, we show that our algorithm is very energy efficient.
- During query processing, several attributes are considered. It is very advisable to deal with multi-attributes query since it can generalize applications in sensor networks.
- Finally, we present an extensive experimental evaluation of our proposed algorithm.

The remainder of this paper proceeds as follows. Related works is covered in Section 2. The algorithm about Maintaining Section-Bits for descendent nodes and query processing are discussed in Section 3. We validate our algorithm experimentally in Section 4 and conclude in Section 5.

2. Related works

People have developed a lot of query processing algorithms for sensor networks. Sensor nodes operate by small size battery depending on applications. Energy consumption is one of the most important factors to be considered when designing any query processing algorithms. Generally, message transmission job(P_{send}) consumes much more energy than message reception job($P_{receive}$) and energy usage in wireless communication systems is inversely proportional in distance(r). Therefore, reducing the number of message propagation is one of good solution methods in order to use energy efficiently in sensor and there have been several kinds of research efforts to lessen the number of unnecessary messages.

$$P_{send} \propto \frac{P_{receive}}{r^{\sigma}} \tag{1}$$

By now researchers have carried out various algorithms to make the sensor networks do query processing in an energy efficient way. These algorithms can be classified into five types, which are aggregation algorithm[6], JOIN algorithm[7], approximate algorithm[8], data collection algorithm[9], and event-based query processing algorithm[10].

One more thing to note here is that there are few works to deal with multi-attribute query processing. As related hardware technologies are advanced, it is possible for one sensor can gather one more attributes. Naturally, there are strong needs to support the kind of query processing.

3. Section-Bit based algorithm

3.1. Setting up routing path

In this paper, we try to develop a multi-attribute query processing algorithm in energy-efficient manner. To do so, each node has to maintain some useful information locally about its neighbor nodes. The information is delivered during routing path setup in initial setting. Therefore, the setup will be first explained in this section and we make assumptions as follows: - Queries are brought forward from users via the sink node to sensor network. That is, we assume that the sink node plays the part in the following function; it receives query from the user, sends it to sensor nodes, and then receives the final result from sensor nodes, and sends it to user. - Every sensor node within sensor network knows its location, and has unique its identification. - Each sensor node has the same computation(low-computation capability, small-size memory, battery operated, etc), communication range, the amount of energy.

Routing path setup is done by two phase – notification phase and parent selection phase. Notification phase starts to send Notification Message (NM) from the sink node to each neighbor sensor node. A NM includes the Hop value of the sink node(initial value = 0), the identifier and the location of the notifier. In the notification phase, each neighbor sensor waits to receive a NM before it notifies itself to the sensors in its radio range. If each neighbor sensor node receives NM_i, does an Algorithm 1. And then, each sensor node notifies the modified NM_{i+1} to neighbor sensor nodes again. If a node receives one more messages, Algorithm 1 plays an important role in the reduction of unnecessary messages transmission using Hops. That is, Algorithm 1 determines a parent node as the nearest node using the hops of the sink node.

Algorithm 1. When a node gets NM_i

- IF the node already got another NM_j (i ≠ j)
  /* Compare hops of NM_i and NM_j */
  - If (Hops(NM_i) > Hops(NM_j)) drop NM_i and exit
  - ELSE IF (Hops(NM_i) == Hops(NM_j))
    - determine its parent node by using location info.
  - End IF
- *New Hops* ← *Hops*(*NM*) + 1
- Disseminate new NM_{i+1} including *New Hops*, its (identification, location).

When sink node sends a NM including Hops(=0), identifier and location of sink node, it can provide the criteria to select a parent node; that is, the node with lower hops and closer to itself can be a parent node. Each node can also compute an estimate of waiting time (*wt*) until parent selection phase starts. That is, if the maximum value of Hops (*maxhop*) within the sensor networks defined as constant, each node determines its maximum waiting time using equation (1) as follows:

$$wt \leftarrow (maxhop - Hopi) \times K \times 2 \quad (2)$$

In equation (1), $Hops_i$ is Hops of the NM_i from each sensor node, constant K is time for process the NM. Thus, the case of sink node has waited for the parent selection message from its children for $maxhop \times K \times 2$. If a sensor node has no children, it chooses its parent. If a sensor has candidate children, it waits until they select their parent and then it starts the parent selection phase. That is, if a sensor has waited for *wt*, it has not received for any parent selection phase, and then it determines itself as leaf node and it starts the parent selection phase. Each node notifies its identifier, location, and partial information of its sensing data to the parent node(parent selection message (*PS*)). In the next subsection, we will explain the issue related with partial information.

3.2. Section-Bit

At the second phase in the previous subsection, sensor node has to deliver information about its local sensing data. Let us assume, for example, that a sensor node can gather n attributes. Each attribute($Attr_i$, $n \geq i \geq 1$) has maximum(Max_i) and minimum(Min_i) value. The interval between Min_i and Max_i of $Attr_i$ is divided by K equal sections(see Figure 1) and one bit is used to represent each area. Therefore, the volume for $Attr_i$ is just K bits. Of course, each sensor node has to maintain correct value locally for $Attr_i$ and SB_i(Section-Bit) is only used to deliver to its parent node.

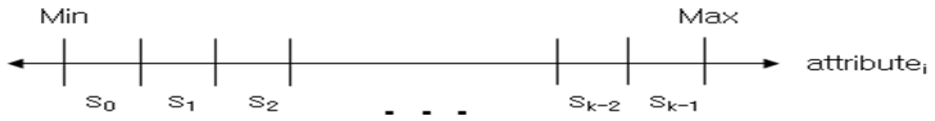

Figure 1. Section-bit for attribute $Attr_i$

At leaf nodes, the second phase(parent selection) start. Initial value for SB_i is 0 and the node sets up section bit 1 based on local sensing data. In case of Figure 2, humidity is shown as example. In the figure, Max(= 100%), Min(= 0%) for humidity is determined and the interval between 0% to 100% is divided into 5 sections to represent humidity. Sensor, *s*3 gathers 18% from its covering area, sets the first bit of *SB* 1, and delivers it to its parent node, *s*2. Some times later, *s*2 comes to know that its children are *s*3 and *s*4 in parent selection messages. It also gathers its local humidity(= 24%) and does OR operation for its local data and *SB*s from *s*3 and *s*4. The result is also delivered to the parent node(*s*1) of *s*2 again. This process will be repeated until sink node.

It should be mentioned that *s*2 has to maintain the *SB*s set for each child node separately although it makes one *SB* for its parent node. Making one *SB* is to decrease the size of *PS* message. If there are *N* attributes in sensor node, *N SB*s have to be delivered.

At the example of Figure 2, attribute humidity is divided into only 5 sections and thus the size of each section is 20%. Depending on the property of attribute, it may be too large or too small. Of course, as an attribute is divided into small size, lots of sections, the correctness is more enhanced but the size of *SB* for the attribute is larger. As a result, the number of sections for each attribute has to be considered carefully.

3.3. Query processing

When a user submits multi-attribute queries (see also Figure 3), they are propagated into networks via sink node. In the example query, 2 attributes are considered. When a sensor

node gets the query, it first checks whether its local value can satisfy the conditions and then whether its descendent nodes can do. If any node does not satisfy the query, the node just discards the query; if not, it propagates the query and wait. If no result message is delivered to itself after wt time later(see subsection 3.1), it also does nothing.

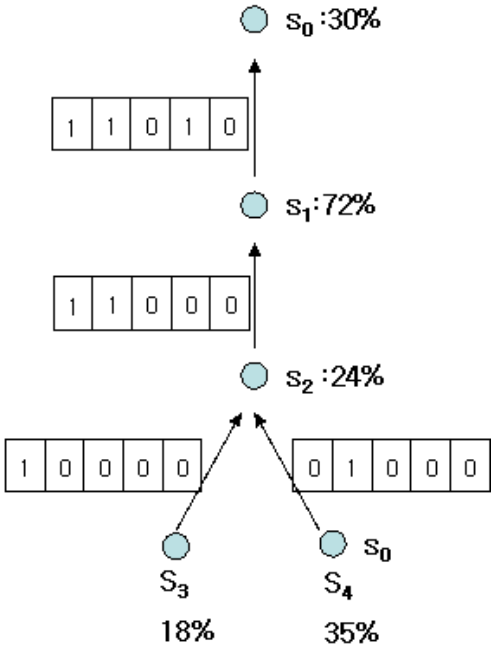

Figure 2. Example of section-bit delivery

Table 1. Example for multi-attribute query

```
Select A.a, A.b
From Attribute A
Where A.a > t1 and A.a < t2 and
        A.b > t3 and A.b < t4 ;
```

4. Performance Analysis

Author In this section, we present an evaluation of our algorithm using a simulator. Section 4.1 describes the various strategies employed for sensor data and Section 4.2 presents the experimental results for different scenarios.

4.1. Simulation Model

In this paper, our main concerns are related with multi-attribute query processing. The proposed algorithm mainly depends on local, partial information about descendent nodes. Naturally, if the local information is not accurate, unnecessary query messages may be propagated is during query processing. Of course, it is reasonable to make the amount of local information less. To show the effects of the facts mentioned above, a various of experiments

are done; that is, we make a simulator by using Java on Linux machine(Pentium Dual Core 3.0 GHz, 2Gbyte memory, Red-Hat Linux 9.0)(see Figure 4). Since there are few works dealing with multi-attribute query processing, we just dig out the properties of our algorithm in detail.

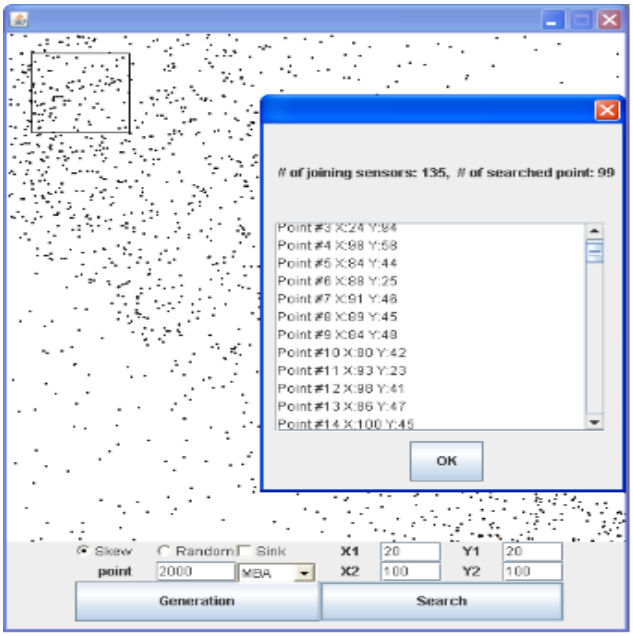

Figure 4. Simulator for Experiment

In the experiments, total area is set to 1024 × 1024 and *NUMs* sensors compose of network. And we assume that the communication range (*Range*) of a sensor is 5. Sink node is assumed to be located in the center of network. In setup phase, the locations of a number of sensors are generated randomly or in skewed manner. In case of skewed distribution, 1/3 of all area is set to dense area and the number of sensors in dense area is more three times than that in sparse area. User forwards queries to the network via sink node. Attribute values in sensor nodes are generated depending on both the values of a neighbor node and the distance between itself and a neighbor node. Finally, a sense node periodically gathers its attribute values, which are updated according to Poisson distribution.

Table 2. Parameter setting

parameter	value	meaning
NUM_s	1000	Number of sensors
Range	10	Communication range
$Size_{msg}$	10KB	Maximum size of a msg
NUM_{att}	5	Maximum number of attributes
$Size_s$	100	Bit number in section-bit

4.2. Results

When a sensor node gets a query, it just propagates or discards the message. If it can satisfy the condition of a query, it should send a reply to its parent node. At the experiments,

only the number of propagated messages is considered, not reply messages since the number of reply messages is the same regardless of experiment settings. It is assumed that the extent of a attribute in a query is 10% between maximum and minimum value on the average.

4.2.1 Effects on the section size

The proposed algorithm tries to decrease the number of messages by utilizing partial information. As the information is more correct, less number of messages is needed during query processing. Energy efficiency is improved as the number of messages keeps small. The first experiment is related with the accuracy of local information. 4 kinds of experiments are executed according to both random/skewed distribution and the number of attributes in a query. For example, '*random - 4*' means that sensors are distributed in random and 4 attributes are contained in a query.

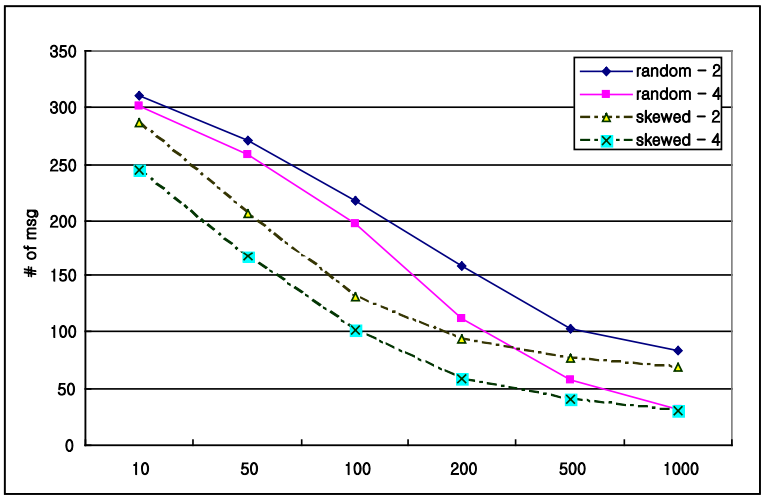

Figure 5. Effects on section size

From the results, we find out two facts. First, if there is more attributes in a query, a smaller number of message propagations is needed. This is because the query conditions explained in Subsection 3.3 are AND-based operation, so less number of sensors can satisfy the query.

Second, the experiments in skewed distribution show better energy efficiency than those in random. In skewed distribution, it is very probable that a node has lots of children sensors in case of dense area. Thus the children nodes of the same parent may have similar attribute values and only one query message from the parent can notify lots of nodes.

As a result, in the figure, skewed distribution with 4 attributes has the best energy efficiency. As mentioned before, if too many bits are allocated for an attribute, partial information(*SB*), itself becomes too large. If a sensor can detect 4 attributes and an attribute is divided into 1000 sections, parent selection message size is bigger than about 4KB, which is not advisable in sensor networks. Thus, we have to choose a proper value between correctness and size.

4.2.2 Effects on the number of attributes contained in a query

The second experiment examines the effects on the number of attributes in a query. In

Figure 6, 'random - 100', for example, means that sensors are distributed in random and 100 bits are allocated per each attribute. The results can be inferred from the first experiments. That is, if there are more attributes in a query, small portion of all sensors can satisfy the condition. From the experiments, 237 message propagations are needed to execute a query with 5 attributes while 421 with 1 attribute in case of 'random - 50'.

From the figure, a query with one attribute produces many message propagations. However, if the number of attribute is more than 2, the difference is not great. This is because it is not probable that two sensors have similar values for two or more attributes.

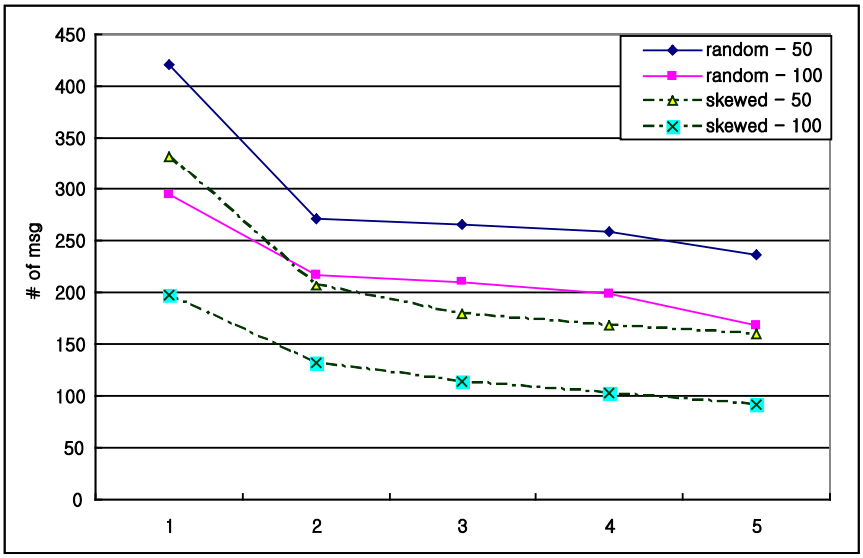

Figure 6. Effects on the number of attributes

4.2.3 Effects on the local data updates

In the above two experiments, it is assumed that there is no update in sensing data. However, a sensor node has to gather new sensing data periodically and compares it with older value. If new value corresponds to another section bit, the node has to inform to its parent. When the parent node gets the update from a child node, it first renews partial information(*SB*) for the child and then computes OR operation on *SB*s for itself and all its children. If the result is the same as one before, there is nothing to do; If not, it also has to inform the update to its parent, and the process will be repeated along routing path.

In the Figure 7, x-axis means update rate. 0.1, for example, means that there are 0.1% updates among all sensors during one query is processed. This experiment only deals with random distribution. If an update occurs, it is assumed that the new value changed between old value+1.5% and old value-1.5%. If more bits are allocated for an attribute, the effects on update are much heavier. In case of 'random - 50' in the figure, from 511 to 10787 update messages are delivered during executing 1000 queries. However, when *SB* size is 200 bits, each section means only 0.5% of all possible value. Thus, changes within ±1.5% may occur lots of update messages. For example, in case of 'random - 200', 334500 messages have to occur to notify the fact to their parent node.

5. Conclusion

In this paper, we deal with the multi-attribute query processing. As is known well, there are several limitations in sensor networks, especially energy. Thus we try to decrease the number of unnecessary messages during processing queries and update notifications. Our main approach is to maintain local, partial information about descendent nodes. The problem is to balance between the size and the correctness of local information. To deal with it, we develop *Section-based* algorithm. From the experiments, we can validate the properties of the proposed algorithm. In further works, we will have to analyze our algorithm and thus show the best balance value for the section size.

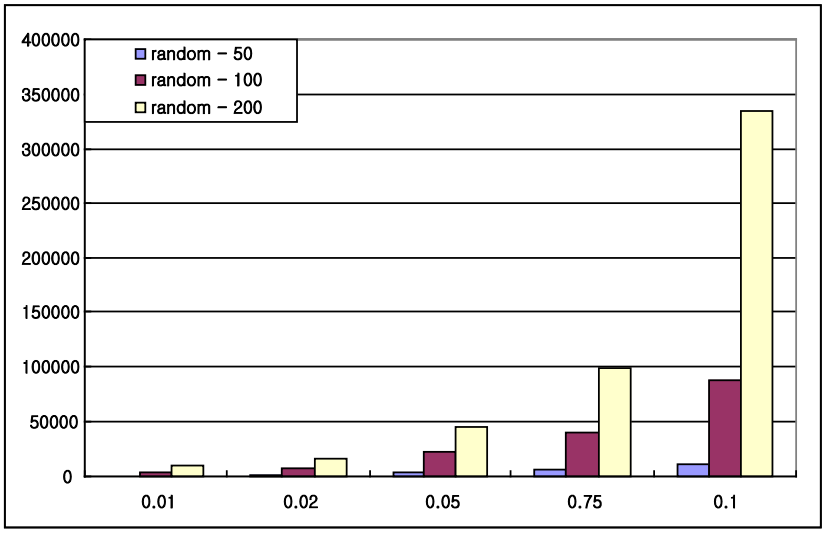

Figure 7. Effects on update rate

The results will help to broaden the application of sensor networks.

And we also try to develop a fault tolerance scheme. Depending on application, the requirements of a sensor node functionality is very different. The sensor model assumed in this paper has properties such as low computing power, short lifetime, cheap, vulnerable to fail and so on. Therefore there are frequent failures in networks and we have to deal with the failure.

References

[1] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson: Wireless sensor networks for habitat monitoring, ACM Workshop onWireless Sensor Networks and Applications, pp. 88-97, (2002).
[2] X. Yang, H. Lim, M.T. Ozsu, and K.L. Tan: In-Network Execution of Monitoring Queries in Sensor Networks, Proceedings of ACM SIGMOD, pp. 521-532, (2007).
[3] S. Borzsonyi, D. Kossmann and K. Stocker: The Skyline Operator, Proceeding of ICDE, pp. 421-430, (2001).
[4] A. Vlachou, C. Doulkeridis, K. Nłrvag, M. Vazirgiannis: On Efficient Top-k Query Processings in Highly Distributed Environments, Proceeding of ACM SIGMOD, pp. 753-764, (2008).
[5] S. Kim, M. Chang and S.O. Yang: Query processing based on local information in Sensor Networks, Proceedings of Circuits/Systems, Computers and Communications, P7(31), (2007).
[6] S.R. Madden, R. Szewczyk, MJ. Franklin, and D.Culler: Supporting aggregate queries over ad-hoc wireless sensor networks, Proceedings of the Workshop on Mobile Computing and Systems Applications, pp. 49-58, (2002).
[7] D.J. Abadi, S. Madden, and W. Lindner: REED: Robust, Efficient Filtering and Event Detection in Sensor Networks, Proceedings of VLDB, (2005).
[8] G. Li, and Jianghong Li: An energy efficient query processing algorithm based on relevant node selection for wireless sensor networks, Proceedings of IEEE SNPD, pp. 149- 154, (2007).

[9] A. Deshpande, C. Guestrin, W. Hong and S. Madden: Exploiting Correlated Attributes in Acquisitional Query Processing, Proceedings of ICDE, pp. 143-154, (2005).
[10] R. Sylvia, K. Brad, S. Scott, E. Deborah, G. Ramesh, Y. Li, and Y. Fang: Data-Centric storage in sensornets with GHT, a geographic hash table, Mobile Networks and Applications, vol. 8, No. 4, pp. 427-442, (2003).

Acknowledgement

This work was supported by National Information Society Agency (grant No. NIA Ⅱ-RER-08013/2008.12).

Authors

Kim, SungSuk received his BS, MS, and Ph.D degree in Computer Science and Engineering from Korea University, Seoul, South Korea, in 1997, 1999 and 2003, respectively. He is currently a assistant professor in department of Computer Science, Seokyeong university, Soeul from 2003. His current interests include mobile information systems, sensor database and ubiquitous computing systems.

Sun Ok Yang received her MS and Ph.D degree in Computer Science and Engineering from Korea University, Seoul, South Korea, in 2002, 2006, respectively. She is currently working as Research professor in Yonsei University from 2008. She has published lots of papers and books in mobile computing systems, ubiquitous computing systems, and sensor network systems.