

An efficient spatio-temporal index for spatio-temporal query in wireless sensor networks

Donhee Lee, and Kyoungro Yoon

Department of Computer Science and Engineering, Konkuk University
Seoul, Korea

[e-mail: donhlee@dreamwiz.com; yoonk@konkuk.ac.kr]

*Corresponding author: Kyoungro Yoon

*Received December 12, 2016; revised March 14, 2017; accepted April 17, 2017;
published October 31, 2017*

Abstract

Recent research into wireless sensor network (WSN)-related technology that senses various data has recognized the need for spatio-temporal queries for searching necessary data from wireless sensor nodes. Answers to the queries are transmitted from sensor nodes, and for the efficient transmission of the sensed data to the application server, research on index processing methods that increase accuracy while reducing the energy consumption in the node and minimizing query delays has been conducted extensively.

Previous research has emphasized the importance of accuracy and energy efficiency of the sensor node's routing process. In this study, we propose an itinerary-based R-tree (IR-tree) to solve the existing problems of spatial query processing methods such as efficient processing and expansion of the query to the spatio-temporal domain.

Keywords: Itinerary, Spatio-temporal Query, MBR, R-tree, IR-tree

1. Introduction

Owing to the significant advancement in technology of various sensors such as temperature, humidity, and light sensors along with wireless communication systems, many studies have been actively conducted to adopt wireless sensor networks (WSNs) in various application fields including military, medicine, meteorology, environment, transport, and education. In particular, numerous studies have been actively performed with a focus on the development of efficient processing of spatio-temporal queries in an environment suited to the characteristics of the sensor nodes, such as low energy capacity with relatively high energy consumption for wireless data transmission [1, 2].

In these sensor networks, the simplest approach to processing spatio-temporal queries is a centralized method. The first step is to collect all the information regarding the location of sensor nodes with sensed data at the server and then processing spatio-temporal queries in the centralized server [3, 4, 5]. Although this method has the advantage of simplicity, it has a severe disadvantage, as it results in poor energy efficiency of the sensor node because of the consumption of a large amount of energy [6] when accessing all the sensor nodes through wireless communication. To overcome the disadvantage of the centralized method, an in-network-based distributed spatial index (DSI) was proposed, which attempts to decrease the number of wireless transmissions required between sensor nodes by employing a distributed spatial filtering of sensor nodes. Since the DSI methods simply apply the conventional spatial index methods to a wireless network without proper modification, this method is also disadvantageous in that it cannot fully optimize the effect of spatial filtering and wireless routing among sensor nodes in an in-network configuration. To overcome this disadvantage, the GR-tree method was proposed [3, 4, 5].

In the GR-tree method, a grid-based tree is created and the spatial index data is stored in a distributed manner at wireless sensor nodes. These procedures allow the GR-tree to achieve an efficient tree-based routing using a DSI mechanism in WSNs. However, GR-tree also has a problem such that parent sensor nodes would consume more energy as they get closer to the base station (BS). Another problem is that energy consumption increases when nodes are added or deleted because GR-tree needs to be restructured and optimized after node addition and deletion.

On the other hand, query aggregation is an essential part of processing multiple queries, which might arrive simultaneously at a high rate [7]. While most existing work has focused on data aggregation, the fact that an efficient query distribution contributes to the improvement in energy consumption has been overlooked. The conventional simple query propagation model has adopted the sequential routing to transmit queries sequentially over the required regions. However, this simple method suffers from the shortcoming that it cannot process concurrent queries. A query aggregation algorithm is a technique that eliminates the duplication of spatial and attribute information for the original query and processes overlapping queries to maximize the use of queries that are simultaneously used in the system for spatial query. This algorithm also has disadvantage in that it generates high message overhead because queries must be sent simultaneously to all nodes in the process of delivering the combined queries to the proper region. As an alternative, a centralized and distributed query optimization (CDQO) algorithm was suggested [8]. Using spatial topological relationships and load balanced cluster routing, CDQO, which is based on a quad-tree infrastructure, performs efficient aggregation and transmission processing so that it can save energy and expand the lifetime of a sensor network.

However, CDQO still has shortcomings: it cannot process spatio-temporal queries and the energy consumption required by the sensor node increases in owing to the communication cost incurred by the parent sensor node and the increased index restructuring cost.

To overcome the problems of the traditional infrastructure (R-tree, quad-tree, GR-tree, CDQO, etc.), which consume large amounts of energy because of dynamic update and are greatly affected by topological changes to the network, an itinerary algorithm was proposed. The itinerary query algorithm is basically a dynamic approach. Instead of building the search path in advance, this algorithm determines the search path at the time of query processing. Without performing an external optimization, it optimizes only the queries within the network. To process the query, the itinerary scheme performs the routing for the interested region without performing the routing for the entire region of WSNs when the user's query is submitted. Following the path routed by the itinerary algorithm, aggregated intermediate query results from the sensor nodes within the communication coverage are collected. The results are then sent to the next sensor node and the final aggregated results of query processing are returned to the server. The itinerary query algorithm can be used on any network environment. It collects the sensed data in a specific order.

To overcome the problems of a spatial query processing scheme and achieve more efficient query processing in WSNs, we propose the itinerary-based R-tree (IR-tree) in this paper. IR-tree can be created by first selecting a center node in accordance with the distribution of sensor nodes in a specific spatial region and second by building a minimum bounding rectangle (MBR)-based R-tree. MBRs in R-tree can overlap owing to its hierarchical tree structure, wherein the MBR of upper layer nodes incorporates the MBRs of lower layer nodes. Owing to the nature of the index structure, while the nodes in the upper layers above a certain level perform R-tree retrieval, the nodes in most of the lower layers use the itinerary algorithm when processing the query. A certain index level for an optimal R-tree level is derived from the performance evaluation for IR-tree.

In a sink node, whether spatial or spatio-temporal, the number of queries and attribute aggregation are carried out using a query aggregation algorithm. Then query is transmitted from the center node to R-tree. Traditionally, spatial query aggregation is performed in the sink node, which is extended to spatio-temporal aggregation in this algorithm. The performance evaluation for the proposed IR-tree is used to prove the superiority of this spatio-temporal aggregation over the traditional spatial query aggregation [9].

2. Related work

2.1 Wireless sensor networks

As shown in Fig. 1, WSNs consist of three parts [10]: (a) a number of distributed sensor nodes which are interconnected via a wireless network to measure physical environment conditions such as temperature, humidity, and illumination; (b) a gateway (or sink node, BS) which collects the sensed data sent by wireless sensor nodes and transmits the data to a central server; and (c) user software, which is a user interface allowing users to store, retrieve, investigate, and utilize the collected data [1]. In general, WSNs support self-organized functions [2,11,12].

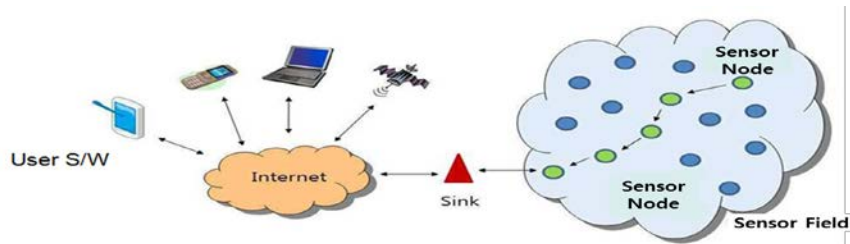


Fig. 1. Structure of WSNs

A wireless sensor node is a microcomputer system in which a microcontroller is embedded. To run the sensing application and to communicate among nodes, an operating system is essential for the microcontroller [3,13]. The operating system running on the microcontroller should be lightweight, consume low power, support low-power communication among sensor nodes, and manage processing and memory in an efficient manner because it needs to be able to run on resource-constrained hardware.

A wireless sensor node has numerous constraints in terms of both software and hardware: the processor's capacity for processing the sensed data, data storage capacity for storing data, transmission distance over which data can be transmitted, and power capacity of the sensor node. Most of all, the power capacity of the sensor node is a major limitation since the data transmission requires considerably larger amounts of energy than data processing. The way to organize WSNs can be divided into three categories: (a) the cluster routing methods; (b) the hierarchical routing method; and (c) the itinerary routing method [14,15].

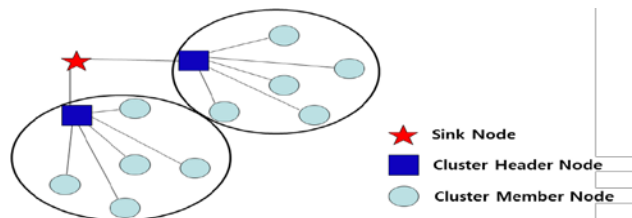


Fig. 2. Cluster routing structure

In the cluster routing method, as shown in Fig. 2, a number of sensor nodes are clustered based on a certain criterion. Once they are clustered, the cluster header sensor node will be selected in each cluster according to the predefined rule. Cluster routing is performed prior to data transmission. Data transmission in the cluster routing algorithm is performed by the following steps. First, the sensor nodes which are members of a cluster send data to the cluster header sensor node in their cluster. Second, the cluster header sensor nodes collect the data sent by their member nodes and send the collected data to the sink node. The cluster routing method enables the amount of data transmitted over the network to be reduced by collecting data for each cluster. However, the large energy consumption occurring at the header sensor node can be a major problem.

In the hierarchical routing method, as shown in Fig. 3, the tree structure is organized for routing, as a sink node becomes a root node. Routing is performed prior to data transmission. Data transmission is performed in a hierarchical way. The Hop3 node sends data to its parent

node, which is the Hop2 node. The Hop2 node sends data to its parent node, which is the Hop1 node. Finally, the Hop1 node sends data to the sink node which is the parent sensor node. The hierarchical routing method enables the amount of data transmitted over the network to be reduced by collecting data through the hierarchical routing path. However, this method has a major problem in that the parent sensor node, which is in charge of routing, might have a large energy requirement.

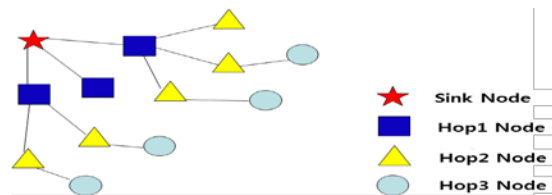


Fig. 3. Hierarchical routing structure

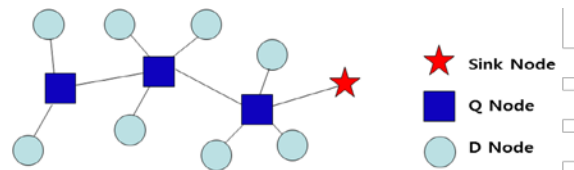


Fig. 4. Itinerary routing structure

Different from the two routing schemes described above, routing is not performed prior to data transmission in the itinerary routing scheme, as shown in Fig. 4 (see also [16]). This means that the routing path is set up once the query is entered into the network. Data transmission is done by first selecting the Q-node in accordance with the predefined rule. The sensor nodes residing within the communication coverage of Q-nodes are selected as D-nodes for setting up the routing path. When a Q-node sends a query to a D-node, the D-node sends data back to the Q-node and the Q-node passes the data to subsequent Q-nodes. The final Q-node collects data and then sends the data to the sink node. The itinerary routing scheme reduces the amount of data transmitted over the network by collecting data in accordance with itinerary routing. However, it also has a weakness in that the total query processing time might be considerably longer due to the sequential processing.

2.2 Grid-based R-tree

As described in Table 1, a centralized processing scheme, which processes data after collecting all the data from the sensors, might cause an increase in communication cost at the sensor node, which results in a reduction in energy efficiency. It utilizes an existing spatial index method (R-tree, R*-tree etc.) [17,18,19,20,21] when it carries out spatial query. With an in-network scheme, on the other hand, spatial queries are processed by nodes, not by the server, which leads to a decrease in the number of wireless communications among sensor nodes owing to spatial filtering. Owing to this, energy consumption at the sensor node can be

reduced [22,23]. However, the traditional in-network method simply applies an algorithm, which is designed for a server-based spatial query processor, to the sensor network without any modification [24]. Although spatial filtering is possible in an in-network scheme, routing among sensor nodes cannot be optimized simultaneously.

Table 1. Types of spatial query processing scheme

Type	Characteristics
Centralized Processing	<ul style="list-style-type: none"> All data sensed by the sensor nodes are sent to server in accordance with routing path set up by specific routing algorithm and spatial query is carried out at the server. Energy consumption at sensor nodes is high.
In-network Processing	<ul style="list-style-type: none"> The sensor nodes collect intermediate query results sent by the previous sensor nodes in accordance with the routing path set up by specific routing algorithm and its own data resulting from processing spatial query by itself. Then they send the combined data to subsequent nodes. Last sensor node sends the final query results to the server.

Therefore, GR-tree was suggested as an alternative since it can minimize energy consumption by nodes and routing among nodes by optimizing spatial filtering in a sensor network. Similar to R-tree, GR-tree constructs a grid-based tree and employs a grid-based clustering scheme to ensure stable wireless routing among nodes and achieving spatial adjacency and minimized overlapping at the same time. GR-tree can be created in two phases: advertisement and parent selection [3,6].

Advertisement is a phase during which parent and children candidate lists are determined. This phase begins at the BS. To identify candidate children nodes, the BS broadcasts advertisement message to all sensor nodes residing within the communication coverage. Once the sensor node receives the advertisement message sent by the BS, it repeats broadcasting the advertisement message to the other sensor nodes. This phase will finish when all the nodes receive the message. When the node sends the message, it broadcasts the AD message to all other nodes except for the node, which was a sender of the message.

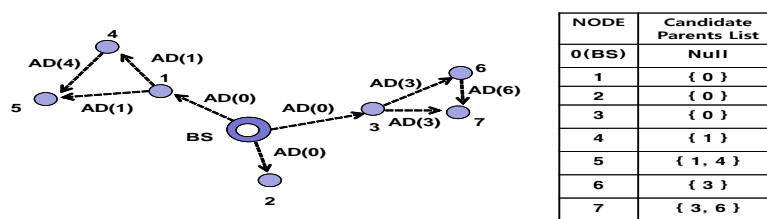


Fig. 5. Advertisement algorithm

Parent selection is a phase during which the node selects its own parent among the nodes in the parent node list by taking into account spatial adjacency and wireless routing. Fig. 6 shows the result of the completion of a parent selection phase. In Fig. 6, E_i denotes the entry information (ID, MBR) for a sensor node i . In the process of insertion, a new node n broadcasts its own message to adjacent nodes residing within communication coverage. Following this, advertisement and parent selection algorithms are performed in the same way as the GR-tree creation algorithm.

Both insertion and deletion algorithms incur high wireless communication cost for updating information on the parent node. The GR-tree algorithm has shortcomings because of the high

cost necessary for restructuring the index and high communication cost for parent nodes that are closer to the BS.

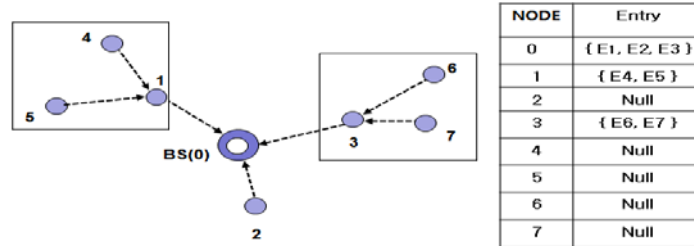


Fig. 6. Result of the completion of a parent selection algorithm

2.3 CDQO

Owing to resource constraints in the wireless sensor nodes, query optimization is essential for handling multiple queries concurrently. Most existing works have focused on data aggregation, while the fact that query dissemination might cause an increase in energy consumption has not been considered. The conventional simple query propagation model has adopted the sequential routing to disseminate queries sequentially over the required regions. This simple method cannot perform concurrent query processing. Moreover, it consumes a large amount of energy because it operates by transmitting queries throughout the entire network and replying. To overcome this, the CDQO algorithm performs query aggregation to allow sensor nodes to efficiently carry out concurrent query processing. Since query aggregation is performed at the BS in advance, the energy consumed by the wireless sensor nodes in the network can be reduced [8,25,26].

The query aggregation scheme minimizes the number of queries to thereby minimize the energy consumed by the sensor node. This scheme is an approach for aggregating multiple spatial queries run on geographical routing. To maximize the number of concurrently used queries in a system for spatial query, this approach eliminates duplication of spatial and attribute information and creates overlapping queries for the original query. A multi-layered structure comprising the query manager, access point, and node was proposed for handling this [10,27,28,29,30].

Although the query aggregation scheme is capable of handling multiple spatial overlapping queries and eliminating duplicate queries, it just transmits all the queries to a specific spatial coordinate without spatial filtering in the process of delivering a combined query to the proper region. As a result, it might generate significantly high message processing overhead and incur large energy consumption at the sensor node when queries are distributed at an access point node in the case of query processing and dissemination over the network.

The CDQO algorithm is based on a spatial query scheme which is proposed to improve the weakness of the existing query aggregation method. Regardless of constraints on hardware at the sensor node, concurrent spatial queries can be processed. This approach applies the CDQO algorithm to the BS considering the relationship among different attributes (temperature, humidity, illumination, etc.).

Fig. 7(a) illustrates that queries Q1, Q2, Q3, and Q4 are reduced to two queries Q1 and Q234 by means of performing spatial query aggregation for Q2, Q3, and Q4. Attributes a1, a2, and a3 for Q234 are also aggregated. Fig. 7(b) illustrates that Q2 and Q4 are aggregated to

Q24 and attributes a2 and a3 for each distinct query are aggregated into the attribute of Q24. In the case of (a), the accuracy might be lower when aggregation is carried out.

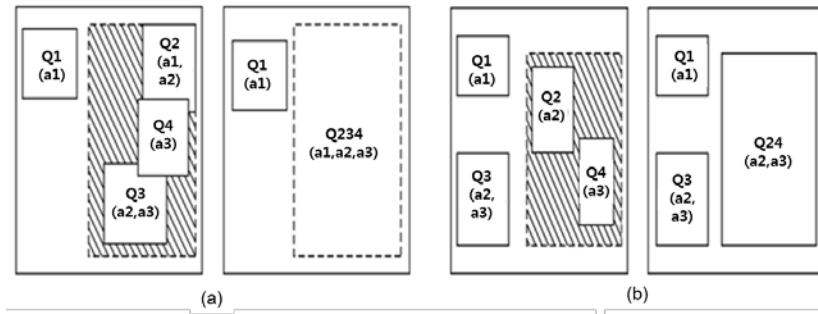


Fig. 7. Space between queries and attribute aggregation

For aggregating the overlapped part in high-rate spatial query, the CDQO algorithm carries out query aggregation at the BS before the query is propagated over the network. Query aggregation is fulfilled by applying the criteria of the spatial and attribute relationship. In the CDQO algorithm, spatio-temporal aggregation is not considered. Although the CDQO algorithm has strength in its efficient query dissemination by making use of topological relation and load-balanced cluster routing, it still has weaknesses in its high communication cost that might occur at the parent sensor node and expensive index restructuring cost when nodes are added and deleted.

2.4 Considerations when querying in WSNs

Studies on spatio-temporal query processing aim at improving three main considerations for efficient processing of queries [20,31].

First, sensor nodes deployed on WSNs can only use strictly limited energy. The largest portion of energy consumption by sensor nodes comes from the communication cost. The communication cost for single data transmission is ten or one hundred times the cost of a single query operation. To lengthen the lifetime of WSNs, energy consumption by sensor nodes should be reduced as much as possible.

Second, sensor nodes deployed for WSNs generally operate using wireless communication. For this reason, the data loss rate is likely to be high due to data transmission errors. Thus, an important consideration lies in an improvement of accuracy for spatial query processing results by efficiently reducing transmission error while sensor nodes transmit data.

Third, query processing would take a long time if too many sensor nodes are deployed in a certain region in a WSN or if the query region is too large. Another important consideration is to reduce spatial operation query processing by considering the distribution of sensor nodes and the size of the query region.

2.5 Problems with the existing work and improvements

In GR-tree, a parent sensor node, which is closer to the BS, consumes more energy. When a query is transmitted, it would be propagated throughout the network starting from the BS

following the path of the tree structure. Aggregated data, in turn, are transmitted up to the BS following the second higher layer parent nodes in reverse order of the query propagation. Consequently, it causes a problem that high layer parent node consumes more energy due to an increase in the number of queries transmitted and the amount of data to transmit. In particular, a skewed structure suffers from extremely high energy consumption at the parent node. If a sensor node is added to or deleted from a tree structure, which is already set up, GR-tree should be optimized and restructured. In the process of restructuring a tree adopting AD and PS algorithm, the cost and energy consumption are likely to be higher.

The CDQO algorithm is based on the quad-tree infrastructure. Making use of spatial topological relation and LBCR, the algorithm performs an efficient optimization for query and transmission processing in a balanced way so that it can save energy at sensor nodes and reduce processing time in the sensor network. In spite of these strengths, however, it still suffers from the weakness of high energy consumption for index restructuring when adding and deleting nodes, coming from the tree structure. Parent nodes also consume large amounts of energy due to an increase in the number of queries transmitted and the amount of transmitted data. In addition, only spatial and attribute aggregation is carried out when query aggregation is performed at the BS. That is, spatio-temporal aggregation cannot be handled. In addition, the accuracy for spatial aggregation is likely to be reduced [32].

As discussed above, the traditional tree structure has the problem that there is considerable energy consumption occurring at the parent node. Traditional infrastructures (GR-tree, quad-tree, R-tree, CDQO, etc.) [6,8,21,31] also have the high-energy-consumption problem caused by dynamic updates and being severely affected by changes in network topology. To resolve those problems, we propose IR-tree in this paper.

In IR-tree, an R-tree is created up to a certain level, while the itinerary query algorithm is applied to the rest of the lower levels [16,32,33]. The selection of an optimized level for dividing the R-tree and itinerary algorithm will be determined by the performance evaluation for IR-tree. The itinerary algorithm is categorized to dynamic routing. That is, the retrieval path is not set up in advance, it is determined at query propagation time. Since this is a dynamic routing scheme, routing optimization is carried out only within the network without necessitating external optimization. The itinerary method performs routing for an interested region to process a query at the time when the query is given rather than setting up a routing path in advance for the entire network region. Moreover, IR-tree covers spatio-temporal query rather than just spatial query when queries are aggregated at the BS so that it contributes to improving efficiency and utilization [24, 34].

3. IR-tree design

3.1 Design introduction

IR-tree is created by first selecting the center node based on the distribution of sensor nodes at the specific sensor spatial region, and then building the MBR-based R-tree. Different MBRs in R-tree can overlap. It presents a hierarchical tree structure so the MBR of upper layer nodes encloses the MBRs of lower layer nodes, and quad-tree is repeatedly constructed for query regions. By nature of the index structure, R-tree retrieval is applied to the upper index above a certain level at query processing, while the itinerary algorithm is applied to most of the lower-level index nodes. The selection of an optimized level for dividing R-tree and the itinerary algorithm will be determined by the performance evaluation for IR-tree. The itinerary method performs routing for an interested region to process a query at the time when the query

is given rather than setting up the routing path in advance for the entire network region. For a path from the sink node to the center node, a minimum path will be searched using the Greedy Perimeter Stateless Routing (GPSR) algorithm [35]. Retrieval is processed in the following steps. The sink node performs a number of spatial or spatio-temporal queries and attribute aggregation using a query aggregation algorithm. Then the query is sent from the center node to R-tree. When the query goes into the itinerary region coming out of R-tree, the GPSR algorithm is used to associate the first Q node for each quad-tree. To collect query results, the final Q node will transmit the query results to the parent node of R-tree, which was built first. For addition and deletion of sensor nodes, updating of the lower index below the specific level is not required because this lower level is based on the itinerary method.

There are two ways to access queries in IR-tree: accessing from itinerary to R-tree in the forward direction and from R-tree to itinerary in the reverse direction. The method to access from itinerary to R-tree makes multiple number of regional R-tree and requires to carry out itinerary routing for the root node. In this case, R-tree should be updated whenever itinerary routing is performed, which results in an increased maintenance cost of R-tree and a reduced efficiency. Therefore, our approach uses both R-tree and a dynamic itinerary algorithm: R-tree is applied for accessing static higher layer, while a dynamic itinerary algorithm is applied for accessing the internal region of R-tree.

3.2 Creation of the index

For processing a spatio-temporal query, IR-tree collects information from all the wireless sensor nodes residing in a sensor region when R-tree is built. To build R-tree, a sensor node closest to the center of SR (sensor network range) is selected as the root node for routing. This node becomes the center node (C-node). Since SR is known beforehand, the center node can be obtained by calculation. C-node selection is proceeded starting from S-node (the BS) which is a starting sensor node for query. Each sensor node on the path calculates a distance from itself to the center node of SR and then estimates distances from the center node to its neighbor sensor nodes residing within its communication coverage. If a sensor node is farther away from the center node than a neighbor node, the node sends the query to the neighbor node. If the C-node is closer to the sensor node than neighbor nodes, the C-node is selected and returned. The C-node selection process is shown in Fig. 8.

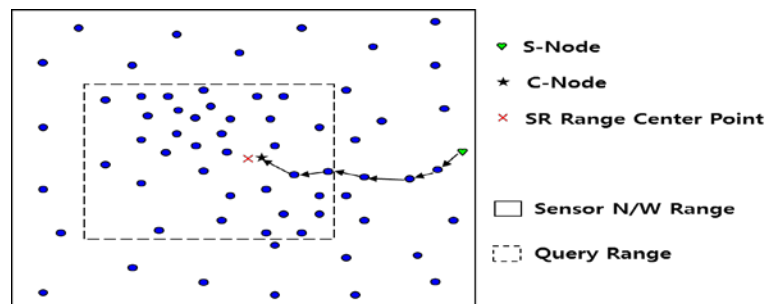


Fig. 8. C-node selection process

The data structure of a spatio-temporal query consists of query *id*, an area which denotes the coordinates of a query region, *qt*, which denotes the query time, *qa*, which denotes the query attributes, *agg*, which denotes the operation type of aggregation, *frequency*, which denotes

transmission frequency of query processing results, and $snode(id, pos)$, which denotes sensor node information. Once R-tree is constructed, R-tree and itinerary boundary-level information will be disseminated to all sensor nodes. When all the sensor nodes receive the information, the information will be stored at the sensor node or reflected to the routing data structure for R-tree. IR-tree performs R-tree routing for all wireless sensor nodes over the entire sensor network by designating C-node as the root node, and selects an optimized boundary level between R-tree and itinerary.

3.3 Parent node selection

To select a parent node, a root node broadcasts build messages (BMs) to all nodes within the communication coverage. The node that has received a BM, in turn, broadcasts BMs to all nodes under its own communication coverage except for the sending node. This cascaded broadcasting will be repeated up to a terminal node. The BM includes the following data: sending node id , sending node location, BS location, and depth information between sending node and BS. The BM transmission process is shown in Fig. 9.

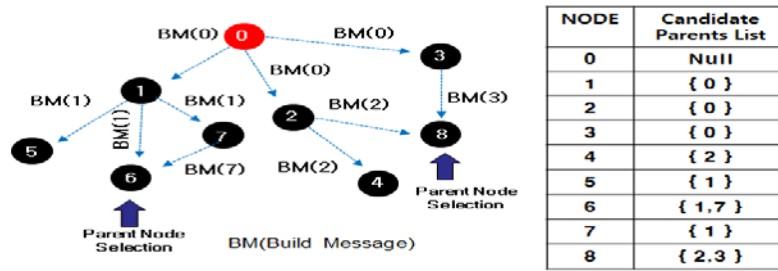


Fig. 9. Build message (BM) transmission

3.4 Construction of the MBR

R-tree constructs an MBR of sensor nodes to collect information on sensor nodes. Fig. 10 illustrates how MBRs are built by sensor nodes. Starting from the C-node which is the root sensor node for R-tree routing, an MBR is organized by including itself and other sensor nodes with children nodes. Once an MBR is organized, the information from sensor nodes within the MBR are collected and passed to the parent sensor node.

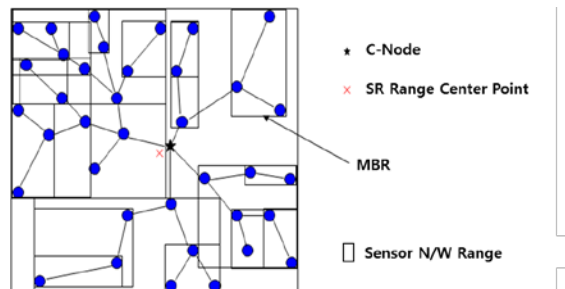


Fig. 10. Construction of an MBR by sensor nodes

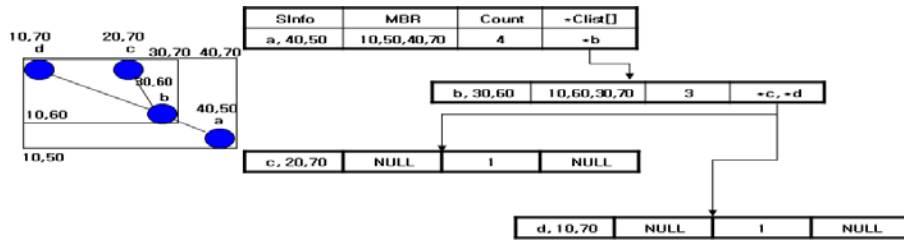


Fig. 11. Data structure of sensor node information

As shown in Fig. 11, the data structure of sensor node information consists of four data sets: (a) SInfo, which denotes its own sensor node information (ID, POS); (b) MBR, which specifies the MBR including itself and its children sensor nodes; (c) Count, which denotes the number of sensor nodes within an MBR; and (d) *CList[], which is an array of pointers which points to data for children sensor nodes. MBR is specified with the bottom left corner point and top right corner point. The bottom left corner point is defined as coordinates x and y which are set to the value of the lowest location of itself and children sensor nodes, while the top right corner point is defined as coordinates x and y which are set to the value of the highest location data. Once an MBR is organized, R-tree and the itinerary routing level boundary value will be disseminated to all nodes in the MBR.

3.5 Spatio-temporal query processing

When spatio-temporal query is processed, R-tree routing is applied for retrieval up to a certain index level which would be derived by the performance evaluation for IR-tree. For the rest of the levels, the itinerary scheme is applied when a query is processed. Information about the boundary level would be disseminated to all sensor nodes at the time when R-tree is first constructed or updated.

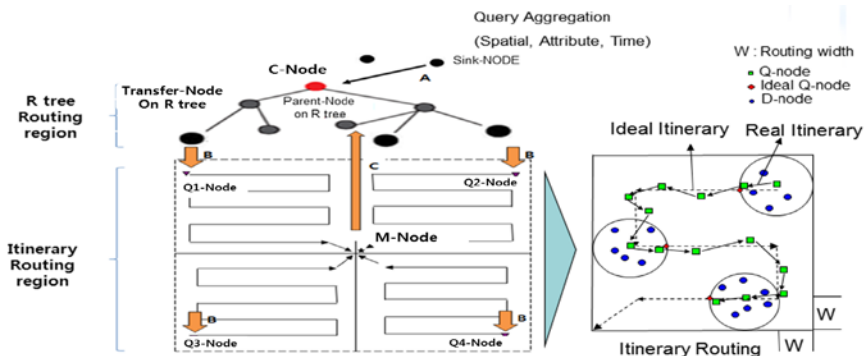


Fig.12. Spatio-temporal query processing

In Fig. 12, the GPSR algorithm is applied to region A which covers from S node to C-node, and region B which covers from R-tree to the first Q node at the boundary of the itinerary region. Since the sensor range is already known in region A, a center node for this sensor range can be calculated. Starting from the S node to the center node, the GPSR algorithm is adopted. Then select the node which is closest to the center position to be the C-node (center node or root node). In region B, the GPSR algorithm is used along from R-tree node to the first node in











the itinerary range. In this case, the first node in the itinerary range is the first Q node, which is positioned at the outermost edge of the itinerary range. In region C, transmission goes to the parent node on R tree which is first built at the last Q node. Following this, the R-tree routing keeps going up to the root node.

Based on itinerary, IR-tree performs routing for an interested region to process a query at the time when a query is given rather than setting up a routing path in advance for entire network regions. IR-tree utilizes itinerary routing for processing spatio-temporal queries in a quad-tree cell.

3.6 Spatio-temporal query aggregation





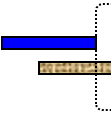

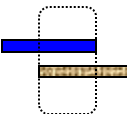

The spatio-temporal query processing module provides a temporal relation operator to support temporal operation processing, as shown in Table 2. The operation returns true or false when it completes.

Table 2. Types of interval temporal relation operators (examples)

Interval temporal relation Operator	A time  B time 
tEquals(stGeometry A, Temporal B) : Boolean	 time range A is equal to  time range B ?
tContains(stGeometry A, Temporal B) : Boolean	 time range A contains  time range B ?
tDisjoint(stGeometry A, Temporal B) : Boolean	 time range A disjoints  time range B ?
tTouches(stGeometry A, Temporal B) : Boolean	 time range A touches  time range B ?

The interval time analysis operators, shown in Table 3, receive two temporal values A and B as input parameters and returns temporal type of value once the operation completes.

Table 3. Types of interval time relation operators (examples)

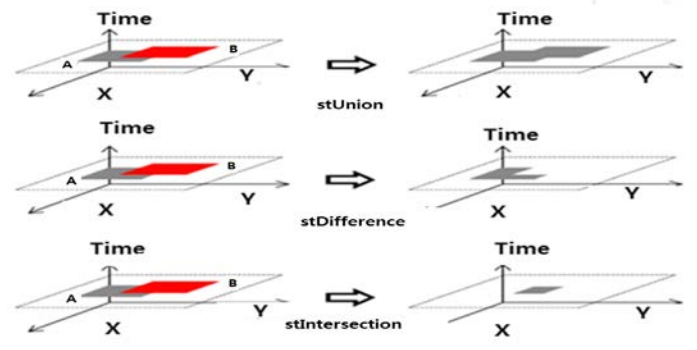
Interval time analysis Operator	A time  B time 
tUnion(Temporal A, Temporal B) : Temporal	 →  Return Union(A, B)
tDifference(Temporal A, Temporal B) : Temporal	 →  Return Difference(A, B)
tIntersection(Temporal A, Temporal B) : Temporal	 →  Return Intersection(A, B)

The spatio-temporal relation operators, as shown in Table 4, receive two spatio-temporal values A and B as input parameters and returns true or false.

Table 4. Spatio-temporal relation operators (examples)

Interval spatio-temporal relation operator	Description
stEquals(stGeometry A, stGeometry B) : Boolean	It returns whether spatio-temporal object in a specific time or specific time interval A and spatio-temporal object in a specific time or specific time interval B are identical.
stCrosses(stGeometry A, stGeometry B) : Boolean	It returns whether spatio-temporal object in a specific time or specific time interval A and spatio-temporal object in a specific time or specific time interval B are crossed over.
stAppears(stGeometry A, stGeometry B) : Boolean	It returns whether spatio-temporal object of a specific time or specific time interval B appears in a specific time or specific time interval A
stDisappears(stGeometry A, stGeometry B) : Boolean	It returns whether spatio-temporal object of a specific time or specific time interval B disappears from a specific time or specific time interval A

To allow a sensor node to perform spatio-temporal operator processing, a spatio-temporal analysis operator processing module (Fig. 13) provides a spatio-temporal analysis operator by extending standard structured query language (SQL) specification released by the Open Geospatial Consortium (OGC). These operators take two spatio-temporal objects A and B, perform an operation, and then return a value of spatio-temporal data type.

**Fig. 13.** Spatio-temporal analysis operator processing module

3.7 Summary of IR-tree functions

The contribution of IR-tree suggested in this paper is that it achieves a reduction of overall query processing time and energy consumption compared with the traditional approaches by decreasing the number of queries by adopting spatio-temporal query aggregation. In addition, 44 operators supporting spatio-temporal query aggregation were also devised. Parallel query processing also contributes to a reduction of query processing time, which is accomplished by organizing the query range as in quad-tree. Aside from that, a dynamic routing scheme is applied to most levels so that the index update cost can be reduced.

Functions of IR-tree are categorized into three parts: index creation, query dissemination/reception, and index update.

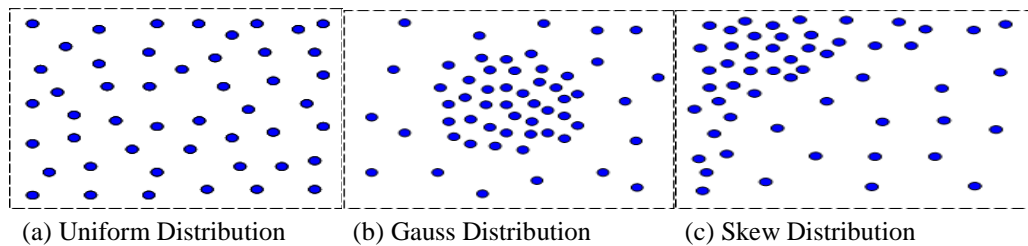
Table 5. Functions of IR-tree

Function	Description
Index creation	<ul style="list-style-type: none"> ▪ Select a center node ▪ Select parent candidate nodes and determine parent nodes ▪ Build MBR-based R-tree ▪ Disseminate information on Itinerary level and query range ▪ Organize Quad tree - Decompose Cell and select representative node for each cell - Set up the query result transmission path in Quad tree
query dissemination/reception	<ul style="list-style-type: none"> ▪ Aggregation of spatio-temporal query at BS ▪ Query propagation from R tree ▪ Itinerary routing and query processing results reception - D node and Q node propagation - D node and Q node duplicated propagation
Index update	<ul style="list-style-type: none"> ▪ Update R tree above specific index level ▪ Quad tree is updated when QR is changed

4. IR-tree performance evaluation

4.1 Configuration for the performance evaluation

Hardware with Intel Core i7 3.4 GHz CPU, 4 GB RAM, and 500GB HDD was used to carry out the performance evaluation. On the computer, Windows 7 was used as an operating system. For performance evaluation and comparison, GR-tree, CDQO, and the proposed IR-tree were implemented in JAVA programming language.

**Fig. 14.** Type of distribution for sensor nodes

To simulate diversified distributions, a uniform distribution, Gauss distribution, and skew distribution were applied when generating sensor nodes randomly (**Fig. 14**). In addition, parameters for performance evaluation are listed in **Table 6**.

In the performance evaluation, depending on whether spatio-temporal query aggregation is applied or not, the case with spatio-temporal query aggregation and the case without it are compared while queries were generated with different types of distribution: uniform, Gauss, and skew distributions.

4.2 Performance evaluation for the proposed scheme

As seen in Fig. 15, the accuracy in the case that spatio-temporal aggregation is applied is slightly lower than in the case that spatio-temporal aggregation is not applied. Despite this, overall accuracy is higher than 95%. In terms of sensor node energy consumption, the case that spatio-temporal aggregation is applied produces better performance by average 7% with the uniform distribution, 24% with the Gauss distribution, and 14% with the skew distribution compared with the case that spatio-temporal aggregation is not applied. In terms of query processing time, the case applying spatio-temporal aggregation produces an improved performance by average of 4% with a uniform distribution, 22% with a Gauss distribution, and 10% with a skew distribution.

Table 6. Parameters for performance evaluation

Parameter	Value
Sensor N/W Size	1,650 m * 1,650 m
Communication Range	30 m limit
Transmission Number	10,000,000
Energy Consumption	1 mJ, per one time
Transmission data size	50 Byte, per one time
Transmission Time	0.1 ms, per one time
Error Period	1% ex) One per 100 times
Sensing Range	70 °C) 70 °C-10 °C +60 °C)
Deviation Range	0.5 °C) 0.5 °C-0.5 °C +0.5 °C)
Node Number	20000, 40000, 60000, 80000, 100000
Query Number	200, 400, 600, 800, 1000
Query Range	330 m * 330 m, 660 m * 660 m, 990 m * 990 m, 1320 m * 1320 m, 1650 m * 1650 m

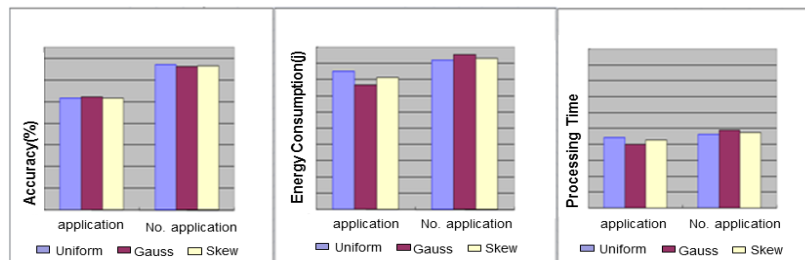


Fig. 15. Performance evaluation depending on whether spatio-temporal aggregation is applied or not (application/no application)

These results can be attributed to the reduced number of data transmissions over the network owing to the application of spatio-temporal query aggregation, which eliminates the necessity to collect data from the sensor nodes for every single query. For this reason, the

accuracy for the case with applying spatio-temporal query aggregation can be slightly lower than that for case without applying spatio-temporal query aggregation. However, from the perspective of more important factors for efficient utilization of sensor nodes such as the overall energy consumption and the query processing time, the IR-tree method gives better performance. These performance differences are notable with the Gauss and skew distributions.

For the performance evaluation at varying itinerary levels, performance was compared by changing the itinerary organizing level from 0%, 30%, 40%, 50%, 60%, 70%, 75%, 80%, 85%, 90%, 95%, to 100% against the whole index height. Above 70%, the interval was set to 5% because many changes appear in that range.

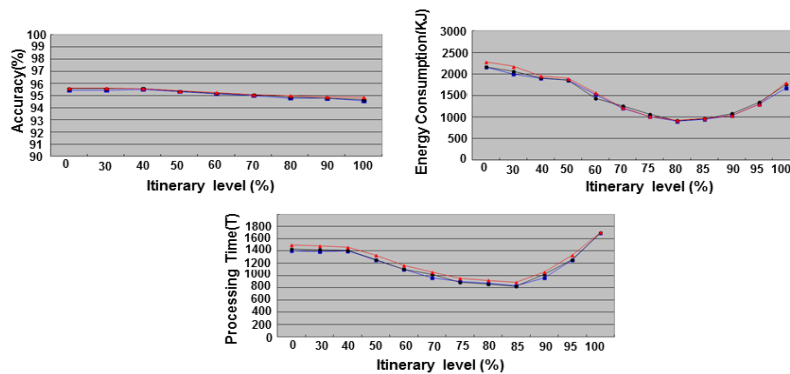


Fig. 16. Performance evaluation at varying itinerary levels

As shown in Fig. 16, the accuracy for query processing results is 95% or higher up to 70% of itinerary level, while performance degrades slightly in the range from 70% to 100%. The accuracy at 0% is higher than at 100%. As the itinerary level increases, the energy consumption and query processing time are reduced. Starting from 80%, the energy consumption increases. Starting from 85%, the query processing time also increases. Performance evaluation results indicate that itinerary consumes less energy than R-tree, while the query processing time in itinerary is longer than R-tree. As the itinerary level increases, query processing accuracy is reduced but energy consumption is also reduced due to itinerary's strengths in index restructuring cost and lower energy consumption at a certain parent node. In addition, parallel processing within the itinerary range also contributes to shortening query processing time. Above a certain level, however, the number of nodes in itinerary is increased, which leads to increased energy consumption and processing time. Consequently, this results in an increase in energy consumption and processing time in IR-tree. From the perspective of accuracy for query processing, too much of a reduction in accuracy can severely affect reliability for the query processing results. Therefore, it is critical to set the itinerary level properly considering all those situations. The performance evaluation results in this study indicate that an itinerary level of approximately 70% against the entire index height maintains 95% or higher accuracy without sacrificing energy consumption and query processing time at a reasonable level.

4.3 Relative performance evaluation

Fig. 17 demonstrates that IR-tree outperforms GR-tree and CDQO by an average of 73% and 25%, respectively, with a uniform distribution, while IR-tree outperforms GR-tree and

CDQO by 80% and 27%, respectively, with a Gauss distribution. In the case of a skew distribution, IR-tree outperforms GR-tree and CDQO by 82% and 24%, respectively.

These results indicate that IR-tree shows better performance than GR-tree and CDQO in terms of energy consumption occurring at sensor nodes because IR-tree employs a dynamic itinerary routing scheme at a specific level. In particular, as the size of the query range increases, these performance differences become larger. Performance evaluation in terms of processing time at different query range size also demonstrates that GR-tree is superior. Although the accuracy in GR-tree is slightly lower than other algorithms, the accuracy level is proved to be conventionally acceptable.

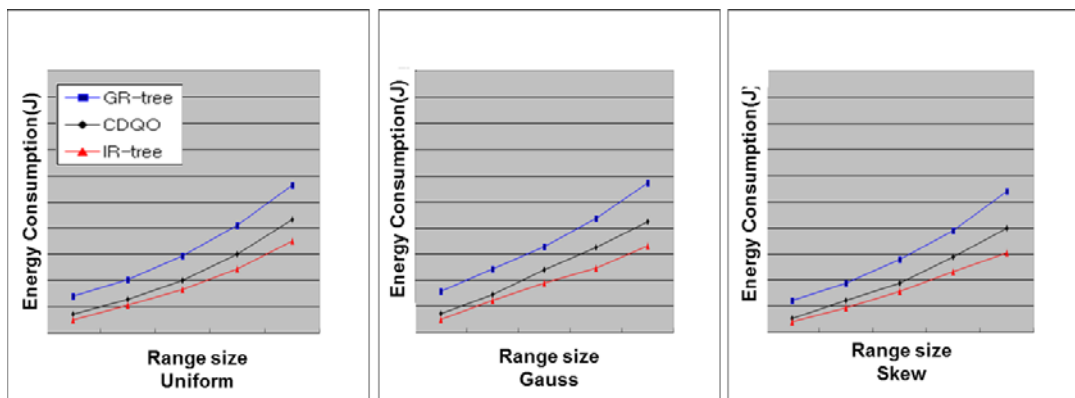


Fig. 17. Performance evaluation in terms of energy consumption while varying the size of the query range

Table 7. Comparison among different algorithms

Classification		IR-tree	GR- tree	CDQO
Aggregation	Object	Spatio-temporal	No function	Spatial
	Operator	① space, ② time / space, ③ time Total 44 species	No function	spatial only Total 8 species
Query accuracy		Maintain accuracy level 95 %	High	Lower than GR-tree
Energy consumption		Good efficiency <ul style="list-style-type: none"> ▪ To reduce the number of communication ▪ To reduce the number of query processing 	Low efficiency	IR-tree and GR-tree middle level (Increase burden of parent node)
Processing speed		Good efficiency (query aggregation & parallel processing)	Lower han IR-tree	Better than GR-tree (spatial aggregation)

4.4 Comparison among algorithms

This section describes conclusions obtained by comparing the performance evaluation results in different cases. IR-tree, GR-tree, and CDQO algorithms are compared in terms of aggregation target and operator, index routing, index update target and cost, query accuracy, energy consumption, processing speed, and range that parent nodes cover. **Table 7** compares the three algorithms. The results indicate that IR-tree reduces query processing time by means of spatio-temporal query processing using various operators and itinerary routing has the advantage in that it consumes less energy by minimizing index updates and the overhead of the parent node.

6. Conclusion

In this study, we have proposed an IR-tree-based algorithm and proved its superiority over the existing spatial query processing schemes (GR-tree and CDQO algorithm) through evaluating its performance and comparing the performance results of the IR-tree, GR-tree, and CDQO algorithms in terms of the accuracy of query processing results, energy consumption, and query processing time.

Spatio-temporal query aggregation performed at the BS contributes to a reduction in the processing time and energy consumption. Parallel processing for the query range in accordance with node distribution also contributes to a reduction in the processing time. Furthermore, most lower levels are routed by applying a dynamic routing scheme, which results in a reduction in the index update cost. Although the accuracy of the proposed IR-tree is slightly lower than that of traditional algorithms, it is persistent with 95% at the 90% criterion. For future study, we plan to extend IR-tree to support efficient spatio-temporal query processing even for moving wireless sensor nodes, which were not considered in this study.

References

- [1] ITU-T Technology, "Ubiquitous Sensor Networks (USN)," *ITU-T Technology Watch Briefing Report Series*, No. 4, 2008. [Article \(CrossRef Link\)](#).
- [2] Abdul Razaque, et al., "P-LEACH: Energy efficient routing protocol for Wireless Sensor Networks," in *Proc. of Long Island Systems, Applications and Technology Conference (LISAT), 2016 IEEE*, 2016. [Article \(CrossRef Link\)](#).
- [3] Jung-Jun Kim, et al., "Efficient Processing of Aggregate Queries in Wireless Sensor Networks," *Journal of Korea Spatial Information Society*, Vol. 19, No. 3, pp. 96–106, 2011. [Article \(CrossRef Link\)](#).
- [4] Meng, Min, et al., "Query aggregation in wireless sensor networks," *Int. J. Multimedia Ubiquit. Eng* 3, pp 19-26, 2008. [Article \(CrossRef Link\)](#).
- [5] Ozdemir, Suat, and Yang Xiao, "Secure data aggregation in wireless sensor networks: A comprehensive overview," *Computer Networks*, 2009. [Article \(CrossRef Link\)](#).
- [6] Min-Soo Kim and In-Sung Jang, "An Energy-Efficient Distributed Spatial Indexing Scheme in Wireless Sensor Networks," *Journal of Korea Spatial Information Society*, Vol. 19, No. 5, pp. 63–74, 2011. [Article \(CrossRef Link\)](#).
- [7] He, Wenbo, et al., "Pda: Privacy-preserving data aggregation in wireless sensor networks," in *Proc. of INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE. IEEE*, 2007. [Article \(CrossRef Link\)](#).
- [8] V.S. Felix Enigo, V.Ramachandran, "Effective Management of High Rate Spatio-Temporal Queries in WSN," *Wireless Pers Com.*, 2014. [Article \(CrossRef Link\)](#).

- [9] Y. Xu et al., "PSGR: Priority based stateless geo-routing in wireless sensor networks," in *Proc. of IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, Washington, DC, Nov. 2005. [Article \(CrossRef Link\)](#).
- [10] Lee, M. and Wong, V. W. S., "An Energy-aware Spanning Tree Algorithm for Data Aggregation in Wireless Sensor Networks," in *Proc. of the IEEE PacRim, Communications, Computers and Signal Processing*, pp.300–303, 2005. [Article \(CrossRef Link\)](#).
- [11] Hartl, G. and Li, B., "infer: A Bayesian Approach Towards Energy Efficient Data Collection in Dense Sensor Networks," in *Proc. of the 25th IEEE Int. Conf. on Distributed Computing Systems*, pp.371–380, 2005. [Article \(CrossRef Link\)](#).
- [12] Jain, A. and Chang, E. Y., "Adaptive Sampling for Sensor Networks," in *Proc. of the DMSN*, pp.10–16, 2004. [Article \(CrossRef Link\)](#).
- [13] Levis, P. et al., "TinyOS: An Operating System for Wireless Sensor Networks," *Springer Ambient Intelligence*, Chapter 7, pp.115-148, 2005. [Article \(CrossRef Link\)](#).
- [14] Kyung-Tae Kim, "Tree-Based Clustering Protocol for Energy Efficient Wireless Sensor Network," *Journal of Korea Information Processing Society*, Vol.17-C, No.1, 2010. [Article \(CrossRef Link\)](#).
- [15] Deligiannakis, A., Kotidis, Y., and Roussopoulos, N., "Compressing Historical Information in Sensor Networks," in *Proc. of the SIGMOD*, pp.527–538, 2004. [Article \(CrossRef Link\)](#).
- [16] Akkaya, K. and Younis, M., "A Survey of Routing Protocols in Wireless Sensor Networks," *Journal of the Elsevier Ad Hoc Network*, Vol.3 No.3, pp.325–349, 2005. [Article \(CrossRef Link\)](#).
- [17] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pp. 47-57, 1984. [Article \(CrossRef Link\)](#).
- [18] GONG Jun et al., "An Efficient Trajectory Data Index Integrating R-tree, Hash and B*-tree," *Cehui Xuebao*, Vol. 44 No. 5 pp.570 ~ 577, 2015. [Article \(CrossRef Link\)](#).
- [19] Jin-Su Kim, et al., "An Energy-Efficient Data Aggregation using Hierarchical Filtering in Sensor Network," *Journal of Korea Society of Computer and Information*, Vol.12, No.1, pp.73–82, 2007. [Article \(CrossRef Link\)](#).
- [20] Y Xu et al., "Processing window queries in wireless sensor networks," *Data Engineering, ICDE*, 2006. [Article \(CrossRef Link\)](#).
- [21] Lakshmi Balasubramanian, et al., "A State-of-Art in R-Tree Variants for Spatial Indexing," *International Journal of Computer Applications*, Vol.42 No.20, 2012. [Article \(CrossRef Link\)](#).
- [22] Li, Hongjuan, Kai Lin, and Keqiu Li., "Energy-efficient and high-accuracy secure data aggregation in wireless sensor networks," *Computer Communications*, 2011. [Article \(CrossRef Link\)](#).
- [23] Liu, Chen Xu, et al., "High energy efficient and privacy preserving secure data aggregation for wireless sensor networks," *International Journal of Communication Systems* 26.3, 2013. [Article \(CrossRef Link\)](#).
- [24] Deligiannakis, Antonios, Yannis Kotidis, and Nick Roussopoulos, "Processing approximate aggregate queries in wireless sensor networks," *Information Systems*, 2006. [Article \(CrossRef Link\)](#).
- [25] Krishnamachari, B., Estrin D., and Wicker, S., "The Impact of Data Aggregation in Wireless Sensor Networks," in *Proc. of the 22nd IEEE Int. Conf. on Distributed Computing Systems*, 2002. [Article \(CrossRef Link\)](#).
- [26] Immanuel Trummer, Christoph Koch, "Multi-objective parametric query optimization," in *Proc. of the VLDB Endowment*, pp.221-232, 2014. [Article \(CrossRef Link\)](#).
- [27] Considine, J. et al., "Approximate Aggregation Techniques for Sensor Databases," in *Proc. of the ICDE*, pp.449–460, 2004. [Article \(CrossRef Link\)](#).
- [28] Manjhi, A., Nath, S., and Gibbons, P. B., "Tributaries and Deltas: Efficient and Robust Aggregation in Sensor Network Streams," in *Proc. of the SIGMOD*, pp.287–298, 2005. [Article \(CrossRef Link\)](#).
- [29] Bash, B. A., Byers J. W., and Considine, J., "Approximately Uniform Random Sampling in Sensor Networks," in *Proc. of the First Workshop on Data Management in Sensor Network*, pp.32–39, 2004. [Article \(CrossRef Link\)](#).

- [30] Deligiannakis, A. and Kotidis, Y., "Data Reduction Techniques in Sensor Networks," *Journal of the IEEE Data Engineering*, Vol.28 No.1, pp.19–25, 2005. [Article \(CrossRef Link\)](#).
- [31] Gaikwad, Priyanka B., and Manisha R. Dhage, "Survey on secure data aggregation in wireless sensor networks," in *Proc. of Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on. IEEE*, 2015. [Article \(CrossRef Link\)](#).
- [32] Gehrke, J. and Madden, S., "Query Processing in Sensor Networks," *Journal of the IEEE Pervasive Computing*, Vol.3 No.1, pp.46–55, 2004. [Article \(CrossRef Link\)](#).
- [33] Xu, Y. et al., "Processing Window Queries in Wireless Sensor Networks," in *Proc. of the IEEE Int. Conf. on Data Engineering*, pp.270–280, 2006. [Article \(CrossRef Link\)](#).
- [34] Patt-Shamir, Boaz, "A note on efficient aggregate queries in sensor networks," *Theoretical Computer Science*, 2007. [Article \(CrossRef Link\)](#).
- [35] P.Samundiswary, et al., "Secured Greedy Perimeter Stateless Routing For Wireless Sensor Network," *IJASUC*, Vol.1 No.2, 2010. [Article \(CrossRef Link\)](#).



Donhee Lee received a B.S. degree in Computer Science from Kangweon University, Korea, in 1991, an M.S degree in Computer Engineering from Yonsei University, Korea, in 2005, and a Ph.D. in Computer and Information Communication Engineering from Konkuk University, Korea, in 2016. He has been working for SK holdings Ltd, Bundang, Korea, from 2002 to present. His research interests include databases, big data, spatio-temporal index in wireless communications, location-based services, and information system audit.



Kyoungro Yoon received a B.S. degree in Computer and Electronic Engineering from Yonsei University, Seoul, Korea in 1987, an M.S.E. degree in Electrical Engineering/Systems from the University of Michigan, Ann Arbor in 1989, and a Ph.D. in Computer and information Science from Syracuse University in 1999. He was a principal researcher and a group leader in the Mobile Multimedia Research Lab, LG Electronics Institute of Technology from 1999 to 2003. He joined the school of Computer Science and Engineering in 2003 as an assistant professor and became a full professor in 2012. He served as a co-chair of Ad Hoc Group on User Preferences and the chair of Ad Hoc Group on MPEG Query Format and Ad Hoc Group on MPEG-V of ISO/IEC JTC1 SC29 WG11 (a.k.a. MPEG). He also served as the chair of the Metadata Subgroup and JPSearch Ad Hoc Group of ISO/IEC JTC1 SC29 WG1 (a.k.a. JPEG). He is an editor of various international standards such as ISO IS 15938-12, 23005-2, 23005-5, 23005-6, 24800-3, 24800-5, and 24800-6. His main research interests include image processing, multimedia information processing, and metadata