
Routing Techniques for Data Aggregation in Sensor Networks

Jeong-Joon Kim*

Abstract

GR-tree and query aggregation techniques have been proposed for spatial query processing in conventional spatial query processing for wireless sensor networks. Although these spatial query processing techniques consider spatial query optimization, time query optimization is not taken into consideration. The index reorganization cost and communication cost for the parent sensor nodes increase the energy consumption that is required to ensure the most efficient operation in the wireless sensor node. This paper proposes itinerary-based R-tree (IR-tree) for more efficient spatial-temporal query processing in wireless sensor networks. This paper analyzes the performance of previous studies and IR-tree, which are the conventional spatial query processing techniques, with regard to the accuracy, energy consumption, and query processing time of the query results using the wireless sensor data with Uniform, Gauss, and Skew distributions. This paper proves the superiority of the proposed IR-tree-based space-time indexing.

Keywords

Itinerary, R-tree, Routing, Sensor Networks, Spatio-temporal Data

1. Introduction

The GR-tree constructs a grid-based tree, and the spatial index information is distributed and stored in wireless sensor nodes [1,2]. The GR-tree generation consists of two steps: advertisement and parent Selection. In the advertisement process, the list of parent and child candidate nodes is determined. In the Parent Selection process, the sensor node selects the optimal parent node by considering the spatial adjacency and wireless routing with the candidate parent sensor list. Through this process, GR-tree performs efficient tree-based routing using distributed spatial indexing techniques in wireless sensor networks. However, since the parent sensor node close to the base station (BS) consumes more energy and optimizes the GR-tree when nodes are added or deleted, the reconfiguration cost and communication cost both increase, resulting in increased energy consumption.

Traditional and simple query propagation models use sequential routing to transmit sequentially to required regions, but they have the disadvantage that they cannot handle concurrent queries [3]. The existing query aggregation algorithm removes the duplication of space and attribute information for the

※ This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Manuscript received April 12, 2017; first revision July 31, 2017; second revision September 7, 2017; accepted September 9, 2017.

Corresponding Author: Jeong-Joon Kim (jjkim@kpu.ac.kr)

* Dept. of Computer Science & Engineering, Korea Polytechnic University, Siheung, Korea (jjkim@kpu.ac.kr)

original query and processes the overlapped query to maximize the queries used simultaneously in the system for a spatial query. However, this algorithm has the disadvantage of high message overhead because it transmits to all nodes in the course of simultaneously delivering the combined query to the appropriate zone. Centralized and distributed query optimization (CDQO) algorithms have been studied as alternatives [4]. CDQO is a quad tree-based structure that can efficiently combine and transmit queries using spatial topology and load balanced cluster routing, thus saving energy and extending the validity period for the sensor network. However, the CDQO technique cannot process spatio-temporal queries and the increase in the index reorganization cost and communication cost of the parent sensor node still involves the problems of energy consumption that are required at the sensor node.

Therefore, the itinerary algorithm has been developed to solve problems in the existing infrastructure (R-tree, quad-tree, GR-tree, CDQO, etc.) that are influenced by phase changes in the network and that consume energy via dynamic updates. The itinerary query algorithm is a dynamic approach that constructs the query process without pre-configuring the search path. It only optimizes queries in the network without external optimization [5]. In other words, the itinerary technique does not pre-configure routing for the whole wireless sensor network, but instead processes queries by temporarily configuring the routing for only the area of interest at the time of the user query. The itinerary uses the query processing method that collects the intermediate aggregation query processing results for the sensor nodes in the communication range according to the routing, sends them to the next sensor node, and sends the final query processing result to the server. The itinerary query algorithm can be used independently on any network base and collects the sensed values in a specific order [6].

This paper proposes an itinerary-based R-tree (IR-tree) to solve the problem of this spatial query processing technique and attain more efficient query processing in wireless sensor networks. In IR-tree generation, a center node is first selected according to the sensor node distribution in a specific sensor space region, and an R-tree is constructed based on a minimum bounding rectangle (MBR). The MBRs of the R-tree may overlap each other, and the MBR of the parent node is a hierarchical tree structure that included the MBRs of the child nodes. For structural reasons with the index, the upper level of the query level is used for an R-tree search and most of the descendants use the itinerary method. In addition, an optimal R tree index level is derived through self-performance evaluation while searching for data [7,8].

A minimum path is searched from a sink node (S-node) to a center node (C-node) using a Greedy Perimeter Stateless Routing (GPSR) algorithm. In an S-node, the search process uses a query merging algorithm and transmits queries from the central node to the R-tree after spatio-temporal, the number of queries, and property merging are performed. In the sink node, the existing spatial query merging is extended to space-time merging.

When the query enters the quad tree cell, which is the itinerary region in the R-tree, the first Q node of each cell is searched by applying the GPSR algorithm and the association between the last Q node and the R-tree at the time when the query result is collected using the parent node of the pre-configured R tree. In addition, the IR-tree requires periodic R-tree reorganization on a certain index level when adding and removing sensor nodes, but it is advantageous in that it does not have to be updated because it is processed by the itinerary method under a certain specific index level.

2. Related Research

2.1 GR-Tree Distributed Indexing

In the GR-tree method, the distributed spatial index information such as ID, MBR, Grid ID, and Depth are distributed and stored in the nodes and GR-tree generation is performed through a two-step algorithm of advertisement and parent selection.

The advertisement process in Fig. 1 is the process of determining the list of parent and child candidate nodes. Starting from BS, BS broadcasts its advertisement message to all sensor nodes within the radius with which it can communicate to find candidate nodes. The sensor node that receives the message from the BS repeatedly broadcasts its advertisement message to the lower sensor node and the process ends when a message is received at all nodes and the message transmission broadcasts an AD message to the remaining nodes except for the node that sent the message to itself. The advertisement message $AD(i)$ of sensor node i is composed of $(i, Depth, L_i, L_b)$: $AD(i) = (i, Depth, L_i, L_b)$. In this case, L_i and L_b refer to the geometric position information of node i and the BS, and the depth refers to the height from the BS.

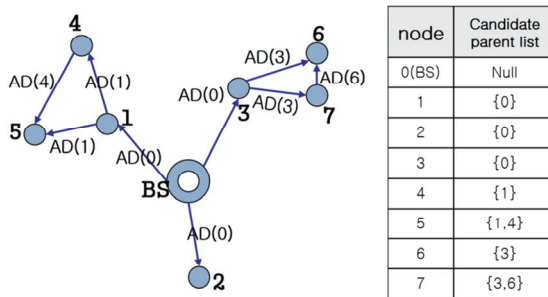


Fig. 1. Advertisement algorithm.

In the parent selection (PS) process, the node selects its parent by considering spatial adjacency and wireless routing to the parent node list. Fig. 2 shows the PS process for adjacent grids with a maximum of eight units. As shown in Fig. 2, to minimize the overlap between the MBRs constituting the GR-tree index, PS is performed in the grid order specified in advance.

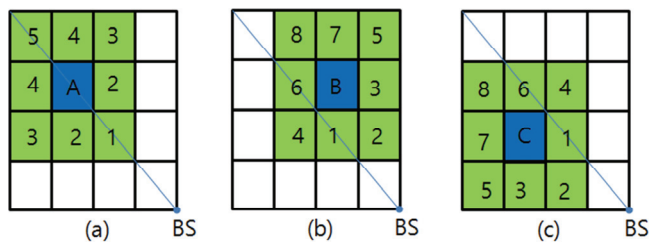


Fig. 2. Search order of parent nodes between grids.

When a parent node is selected, if there exists a plurality of candidate parent nodes in the same grid, a node adjacent to the BS is selected. That is, if there exists a plurality of parent node candidates in each of

the A, B, and C grids, the node closest to BS in each grid is selected as a parent node. If there is no candidate parent node in the same grid, PS is performed according to a predetermined grid sequence. In case (a), A grid searches eight adjacent grids in ascending order from 1 to 5 and the grid with the same order performs the bottom first. In the case of (b) and (c), eight grids adjacent to B and C are searched in ascending order from 1 to 8.

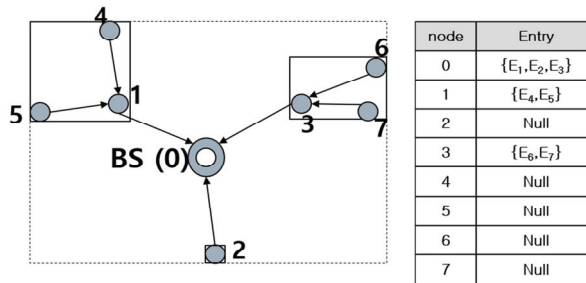


Fig. 3. Parent selection algorithm execution result.

Fig. 3 shows the result of the PS, and E denotes the Entry information (ID, MBR) of the sensor node i [6,8]. In the Insert process, a new node, n , broadcasts its message to neighbor nodes in its communication range. Then, AD and PS algorithms are executed in the same way as the GR-tree generation algorithm. The Deletion process uses a method of periodically checking the relationship between a parent and a child at a specific time in order to find out which node should be deleted due to a failure. When a failure node is deleted from the GR-tree, it does not operate and is therefore automatically deleted during the routing process. The Insertion and Deletion algorithm is a task for updating the information of the parent node, which causes a lot of wireless communication costs. That is, the disadvantage of the GR-tree is that the cost of index reconstruction is high and the cost of communication is high for the parent node close to the BS.

2.2 CDQO Algorithm

The CDQO algorithm is a spatial query technique that improves the disadvantages of existing query aggregation techniques. It also handles concurrent spatial queries irrespective of H/W limit of the sensor node. We apply the CDQO algorithm to BS considering the correlation between multiple attributes (temperature, humidity, illumination, etc.).

An efficient routing structure is an important factor for energy optimization. We use the cluster tree based routing (CTBR) algorithm for query propagation and result collection to reduce message flow and energy consumption of sensor nodes in large capacity networks. CTBR separates the space into rectangular subspaces and then bisects the space using the orthogonal recursive bisections (ORB) algorithm. In case of bisecting, load-balancing partitioning is performed considering the distribution of nodes.

In other words, it forms a quad-tree shape with four children and separates continuously, and each partial space means a cluster. For each cluster, the cluster head is selected considering energy conservation, the number of neighboring nodes, and the distance between neighboring nodes. As shown in Fig. 4, the head of the cluster repeats up to the BS to select a parent node at a higher level than

the current parent node.

The wireless sensor node selects the closest cluster head and transmits its position information to the cluster head. The cluster head collects location information of all the sensor nodes in the cluster. When a query arrives at the cluster head, the cluster head delivers it to all sensor nodes in its cluster. At this time, a sensor node executes only when its position is within the query space range. The cluster head collects the query results and sends them to the BS along with the parent node. Load balancing the routing structure consumes the same amount of energy for all of the clusters, which can prevent the entire network from going down when one node is overloaded and goes down.

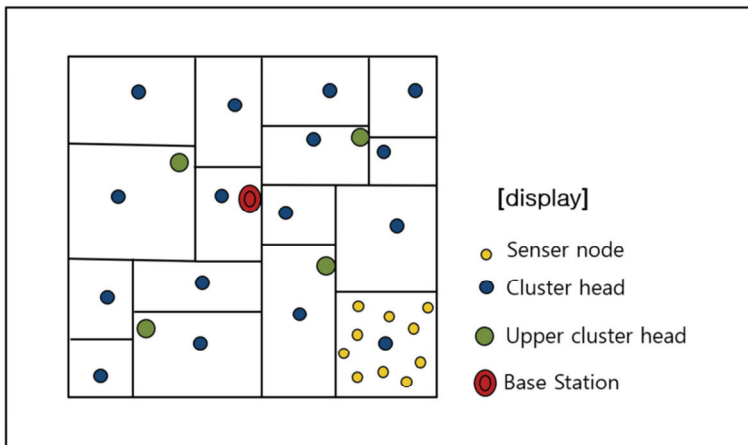


Fig. 4. Load-balancing partitioning.

The query merging of the CDQO algorithm applies spatial and attribute correlation. The query configuration used in the CDQO algorithm consists of the following elements. For example, assuming the following two queries, there is a spatial (attribute) redundancy in each.

$$Q = \langle S, A \rangle$$

S: Spatial information that points to the geographic area that the user needs,

A: Query attributes for the data to be collected,

Q1: Reporting the temperature is in the range $[(23,45)-(100,123)]$,

Q2: Reporting the temperature and the humidity are in the range $[(30,50)-(145,133)]$.

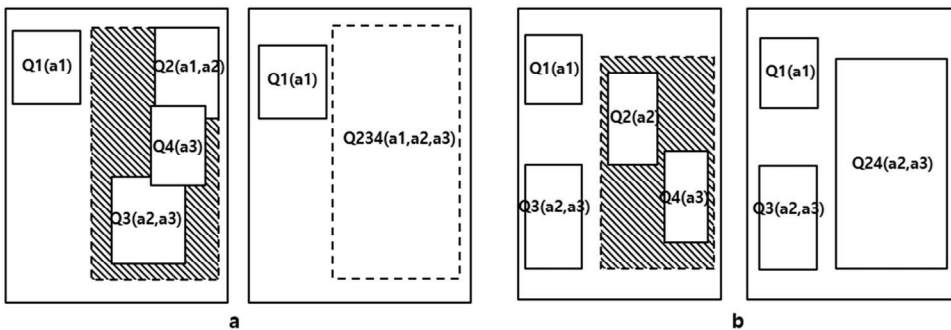


Fig. 5. Merging space and attributes between queries.

As shown in Fig. 5(a), queries Q2, Q3, and Q4 are merged into Q1 and Q234, and the attributes of Q234 are merged into a1, a2, and a3. In Fig. 5(b), the queries Q2 and Q4 are subjected to spatial merging, and finally reduced to three queries Q1, Q3, and Q24, and the attributes of Q24 are merged into a2 and a3. However, in the case of a, the accuracy of merging can be lowered.

Fig. 6 shows eight spatial phase relationships—Disjoint, Contains, Inside, Meet, Coveredby, Cover, and Overlap—among spatial queries. The CDQO algorithm performs query merging in the BS before it is propagated to the network in order to merge the overlapping parts in the high-rate spatial query. Query merging applies the criteria of spatial and attribute correlation. The CDQO algorithm does not consider spatio-temporal coalescence. In addition, there is still a disadvantage that the index reorganization cost and the communication cost of the parent sensor node increase when the node deletion and addition are performed, despite the advantage of spreading the query efficiently using the phase relation and the load-balanced cluster routing.

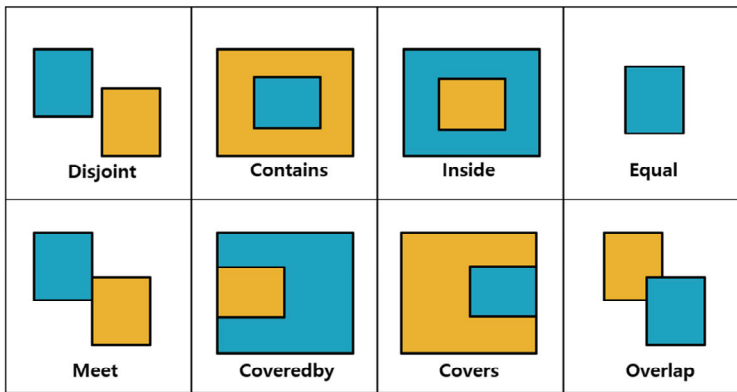


Fig. 6. Spatial phase relationships between queries.

3. Design

3.1 Create Index

For the spatio-temporal query processing, the IR-tree first collects information about the sensor nodes constituting the R-tree for all the wireless sensor nodes in the sensor area. For the R tree structure, the sensor node closest to the center point of the sensor network area (SR) is searched and selected as the root node of the routing, and this node becomes the C-node. Since the sensor network area is known in advance, the center point can be calculated. Fig. 7 shows the process of selecting the C-node, which is the root sensor node of R-tree routing.

Fig. 7 shows the C-node selection process starting from the S-node (base station), which is the query start sensor node. Each sensor node on the path calculates the distance between the center point of the sensor network area and its sensor node and calculates the distance between the center point of the sensor network area and the adjacent sensor node in its communication range. If the sensor node is farther from the C-node than its neighbor sensor node, it sends a query to the neighboring sensor node and otherwise returns to the C-node.

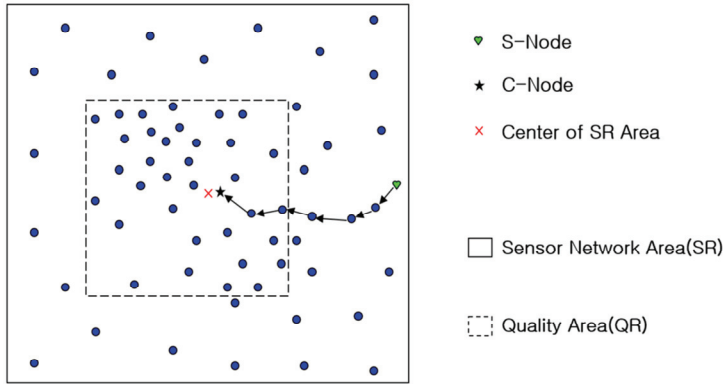


Fig. 7. C-node selection process.

The spatio-temporal query data structure consists of a 'query id' and 'area' indicating the query area coordinates, 'qt' indicating the query time, 'qa' indicating the query attribute, 'agg' indicating the aggregation operation type, 'frequency' indicating the query processing result transmission period, and 'snode(id, pos)' indicating the sensor node information. The R-tree and Itinerary boundary level information propagate information to all the sensor nodes with one-time after the initial R-tree configuration, and store sensor node information in all sensor nodes or reflect it in the R-tree routing information data structure. The IR-tree constructs R-tree routing for all wireless sensor nodes for the entire sensor network as the root of the selected C-node, and selects the optimal level boundary between the R-tree and Itinerary. Fig. 8 shows the R-tree routing structure.

As shown in Fig. 8, root and non-end sensor nodes have several child sensor nodes, and child sensor nodes have only one parent sensor node (the sensor node closest to the root node). If there are several sensor nodes closest to the root sensor node, the sensor node having the largest energy residual is selected as the parent sensor node.

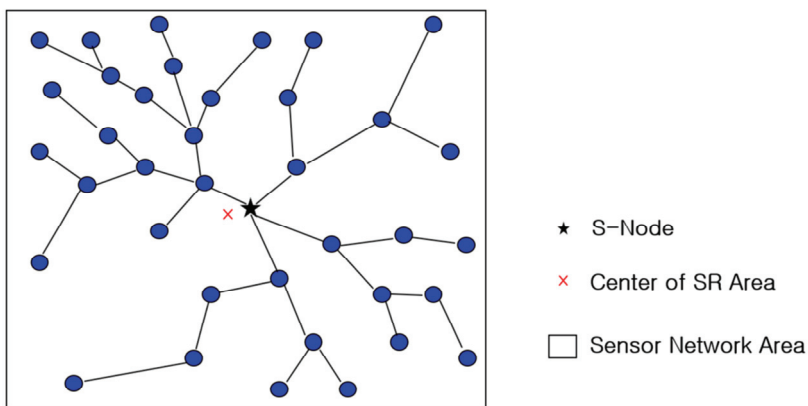


Fig. 8. R-tree routing structure.

To select a parent node, a build message (BM) broadcasts to all nodes within the communication radius from the root node, as shown in Fig. 9. The node receiving the BM broadcasts the BM again to all nodes within its communication radius except for the node that transmitted the BM, and continues

execution to the terminal node. The BM message includes the transmitting node id, the transmitting node position, the BS position, and the depth information of the transmitting node and the BS.

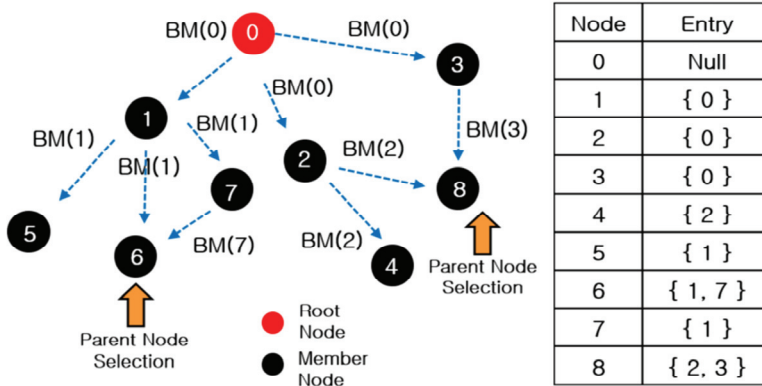


Fig. 9. BM message transmission.

3.2 Spatio-temporal Query Processing

In the spatio-temporal query processing, R-tree routing searches up to a certain level, which is a merge index step to be derived from the self-performance evaluation, and itinerary method is used for query processing from a specific index level. Information for a particular level is propagated to all sensor nodes during the initial R-tree configuration or tree update.

In Fig. 10, the segment to which the GPSR algorithm is applied is segment A (from the S-node to the C-node) and segment B (from the R-tree to the first Q-node of the Itinerary zone). In section A, since the sensor area is known in advance, the center point of the sensor area can be calculated. After proceeding from the S-node to the center point using the GPSR algorithm, the node closest to the center point is selected as the C-node (center node, root node). In section B, the GPSR algorithm is applied from the R-tree node to the first node of the Itinerary zone, and the first node is the first Q node as the most located node in the Itinerary zone. In section C, it is transmitted to the parent node on the R-tree that is constructed first in the last Q node. Then, R-tree routing is performed up to the root node.

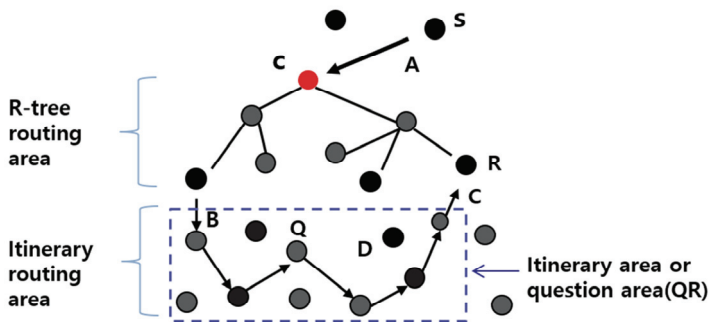


Fig. 10. GPSR before quad-tree application.

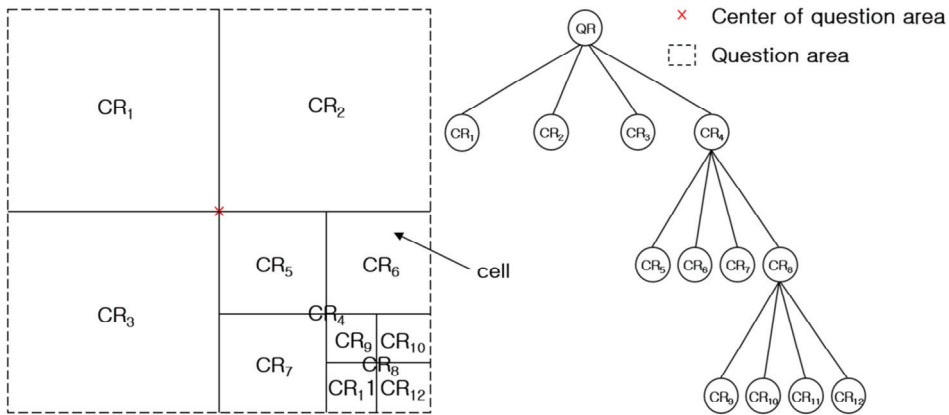


Fig. 11. Quad tree configuration example.

A quad-tree is formed by dividing the query region (QR) into a plurality of cells by using the information of the sensor nodes collected in the R-tree configuration. The reason for designing a quad tree is to reduce query processing time through distributed query processing. After the R-tree structure is established, the query region is determined by performing a merge process when a query arrives at the BS, and the query region information is pre-propagated to each node. Fig. 11 shows an example of quad tree configuration.

As shown in Fig. 11, the QR is divided into CR1, CR2, CR3, and CR4; CR4 is divided into CR5, CR6, CR7, and CR8; and CR8 is divided into CR9, CR10, CR11, and CR12 again. That is, the query area is repeatedly divided until the number of indoor sensor nodes becomes smaller than the indoor maximum sensor node number CMaxC.

To perform query processing in IR-tree in parallel for each cell region of the quad-tree, we select C-node which is representative sensor node in quad-tree cell. Fig. 12 shows an example of C-node selection as a representative sensor node in a quad-tree cell.

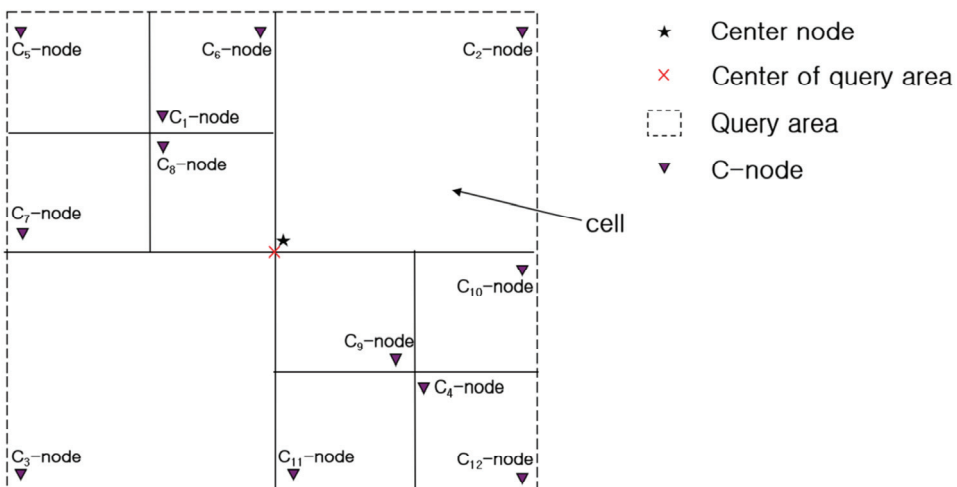


Fig. 12. C-node selection example.

As shown in Fig. 12, the closest node to the C-node of the query area is selected as the root node of the quad-tree as the representative sensor node C-node (the center node of the quad-tree). As the non-end node, the sensor node closest to the center of the cell is selected as the representative sensor node of each cell (C4, C5, C8, C9), and the sensor node closest to the corner in each cell is selected as the representative sensor node. In order to pass the query processing result to the root node of the quad tree, the query processing result of each cell region is transmitted to each parent node. Fig. 13 shows an example of the process of transmitting query processing results among representative sensor nodes in each cell.

As shown in Fig. 13, the query processing results of C10-node, C11-node, and C12-node are transferred to C8-node, and the query processing result of C8-node is transferred to C9-node. The query processing result of C6-node, C7-node and C9-node is transmitted to C4-node, and the result of query processing of C4-node is transmitted to C5-node. Finally, the query processing results of C1-node, C2-node, C3-node, and C5-node are transferred to the central node, and the final query result of the central node is transmitted to the parent node of its R-tree.

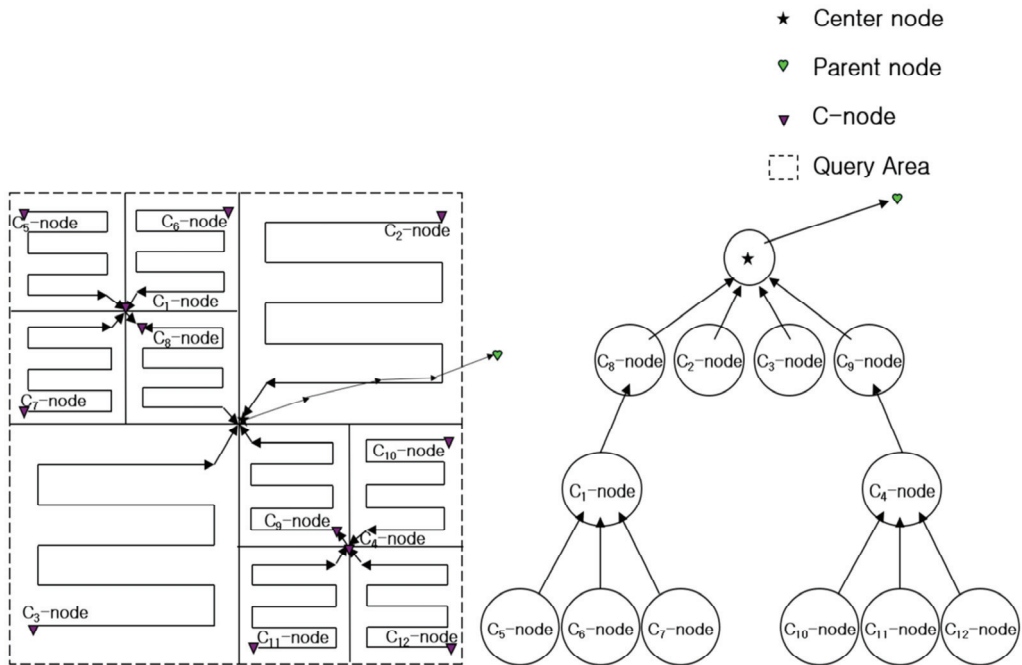


Fig. 13. Example of query processing result transmission process.

The quad-tree node structure consists of 'Type(root, non-terminal, terminal)', representing the node type; 'CInfo', representing the representative sensor node information of the cell; 'CR', indicating the cell region; '* PPtr', indicating the pointer pointing to the parent node; '* CNEPtr', '* CSWPptr', and '* CSEPtr', indicating the pointer pointing to the child node; 'SInfo []', indicating representative sensor node information to receive the query processing result; and 'DInfo', indicating representative sensor node information to transmit the aggregate operation query processing result.

The IR-tree processes the query by constructing the routing temporarily only for the region of interest at the given time, without pre-configuring the routing for the whole query area by the itinerary method after the quad-tree structure. The IR-tree uses itinerary routing to process spatio-temporal queries in quad tree cells. Fig. 14 shows an example of itinerary routing process.

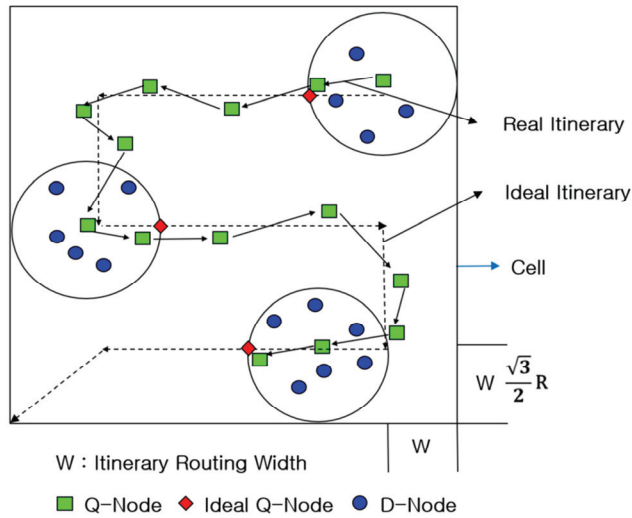


Fig. 14. Routing process in itinerary cell.

As shown in Fig. 14, the Q-node in each cell collects data of D-nodes within the communication range according to Ideal Itinerary routing, processes the space-time query, and transmits the result of spatio-temporal query processing to the next Q-node. In this case, the actual routing path of the Q-node is real itinerary routing, and each itinerary routing interval, W , is set to $\frac{\sqrt{3}}{2}R$ using the communication range, R , of the sensor nodes. Fig. 15 shows an example of itinerary routing.

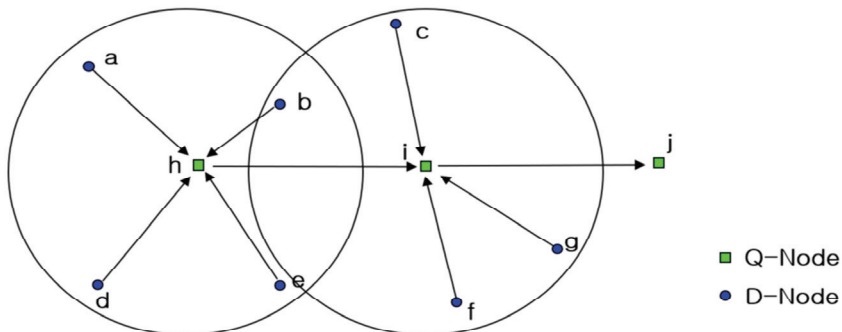


Fig. 15. Itinerary routing example.

As shown in Fig. 15, the Q-node transmits a query to the remaining D-nodes, except for the D-node included in the communication range of the previous Q-node, among the D-nodes within the

communication range, and the D-node transmits the sensing data to the Q-node. The Q-node processes the intermediate aggregation query and sends the result to the next Q-node according to Itinerary routing.

The D-node of the IR-tree performs dual data transmission without retransmitting data between sensor nodes in order to reduce errors resulting from aggregate query processing due to network transmission errors. That is, in order to prevent transmission delay, double transmission processing is performed without retransmission. As a result of this double data transmission, the accuracy of the query processing result and the processing time are improved, but energy is consumed. However, since it is necessary to maintain minimum accuracy, it performs double data transmission in spatio-temporal query processing. Fig. 16 shows an example of the D-node dual data transmission process of IR-tree.

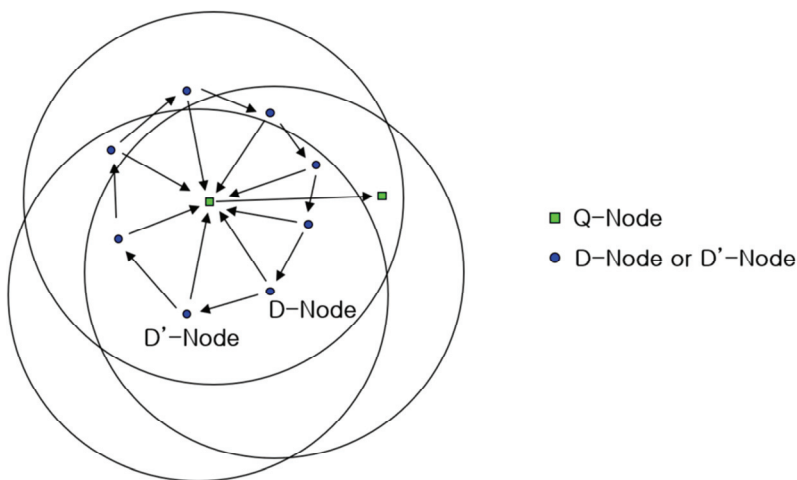


Fig. 16. D-node dual data transfer process example.

As shown in Fig. 16, the D-node transmits the sensed data to the Q-node and the neighboring D'-node. That is, the D-node includes the D'-node (i.e., the neighbor sensor node of the Q-node and the D-node) in the communication range, and double transmits the data to the D'-node, which is the D-node satisfying the Right-Hand Rule. Then, the D'-node transmits the data of the D-node and the data it has sensed to the Q-node. Due to the dual transmission of the D-node, the data of the D-node is duplicated in the data received from the D-node and the D'-node in the Q-node. Therefore, in the IR-tree, the D-node adds its own ID when transmitting data, and the Q-node that transmitted the data removes the data of the duplicated ID, processes the operation query, and transmits the data to the next Q-node. Fig. 17 shows an example of D-node dual data transmission of IR-tree.

As shown in Fig. 17, D-node *a* not only transmits sensor node ID and sensed data to Q-node *d*, but also transmits them to D'-node *b* within the communication range of D-node *a* and Q-node *d*. D'-node *b* transmits sensor *d* node ID, sensed data, and data received from D-node *a* to Q-node *d*. The other D-nodes transmit the data to the Q-node and other D'-nodes in the same way. After receiving these data, the Q-node removes the data collected from the duplicated IDs and transmits the result of the aggregation query processing to the sensor node ID and the sensed data to the next Q-node.

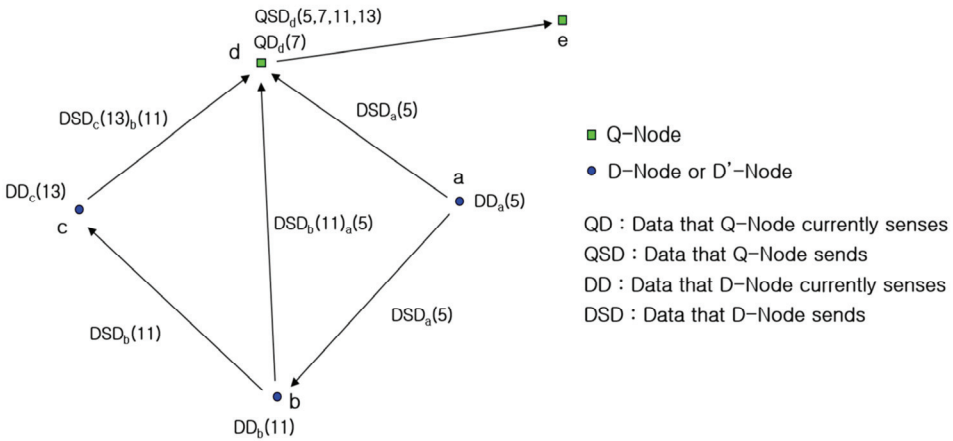


Fig. 17. D-node double data transfer example.

The D-node of the IR-tree transmits the data to the D'-node of the attention, without retransmission in case of error, in order to minimize the transmission delay when transmitting data to the Q-node. However, Q-node performs both retransmission and double data transmission in case of data transmission error between Q-nodes in order to reduce the error of processing result of spatio-temporal query due to network transmission error. This is to increase data reliability. However, in this research, the Q-node dual data transmission process is applied, as shown in Fig. 18, to minimize transmission delay.

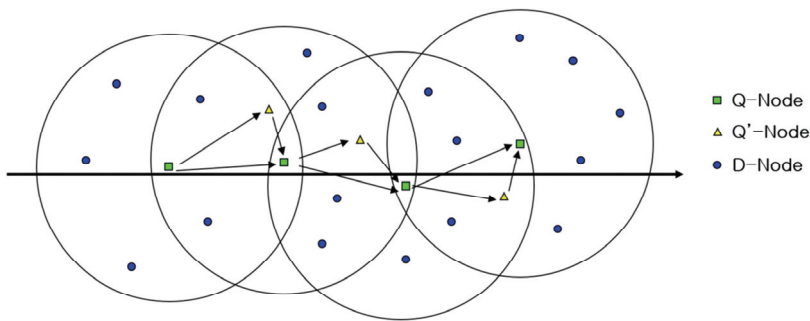


Fig. 18. Q-node data double transfer process example.

As shown in Fig. 18, the Q-node transmits the data collected from the D-node to the next Q-node and the neighboring Q'-node. That is, the Q node includes the next Q-node within the communication range of the next Q-node and the neighboring sensor node of the Q-node, and double-transmits the data to the Q'-node closest to the next Q-node. Then, the Q'-node transmits the data of the Q-node and the data it has sensed to the next Q-node. With the double transmission of the Q-node, Q-node data is duplicated among the data received from the Q-node and Q'-node in the next Q-node. Therefore, in the IR-tree, the Q-node transmits its data by adding its own ID when it transmits data. After receiving the data, the Q-node removes the data of the duplicated ID and transmits it to the next Q-node.

Fig. 19 shows an example of Q-node dual data transmission in the IR-tree.

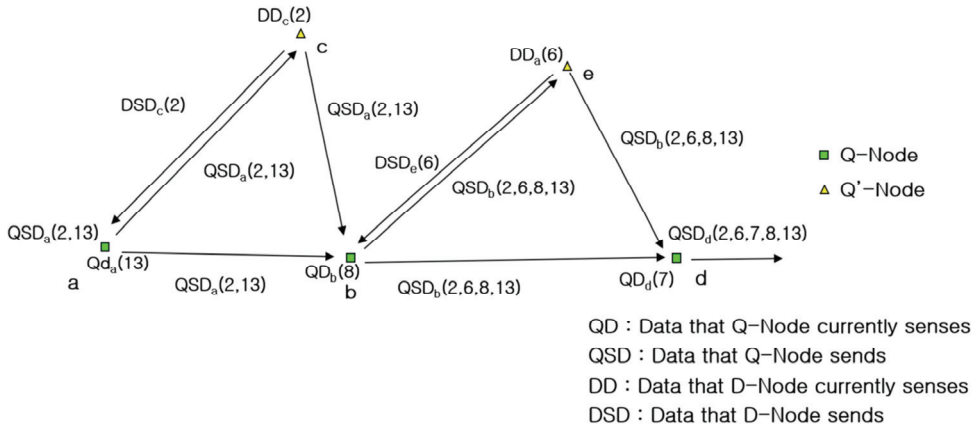


Fig. 19. Q-node double data transfer example.

As shown in Fig. 19, Q-node *a* not only transmits the sensor node ID and the sensed data to the next Q-node *b*, but also to Q-node *a* and the next Q'-node *c* within the communication range of Q-node *b*. Q'-node *c* transmits the data received from Q-node *a* to the next Q-node *b*. The other Q-nodes transmit data to the next Q-node and Q'-node in the same way. After receiving this data, the Q-node removes the data collected from the duplicated ID and then transmits the intermediate result of querying the sensor node ID and the sensed data to the next Q-node.

In Fig. 20, the region to which the GPSR algorithm is applied after the quad-tree is B is the region A from the base station to the C-node, and from the R-tree to the first Q nodes (4, Q1 to Q4) of the itinerary zone.

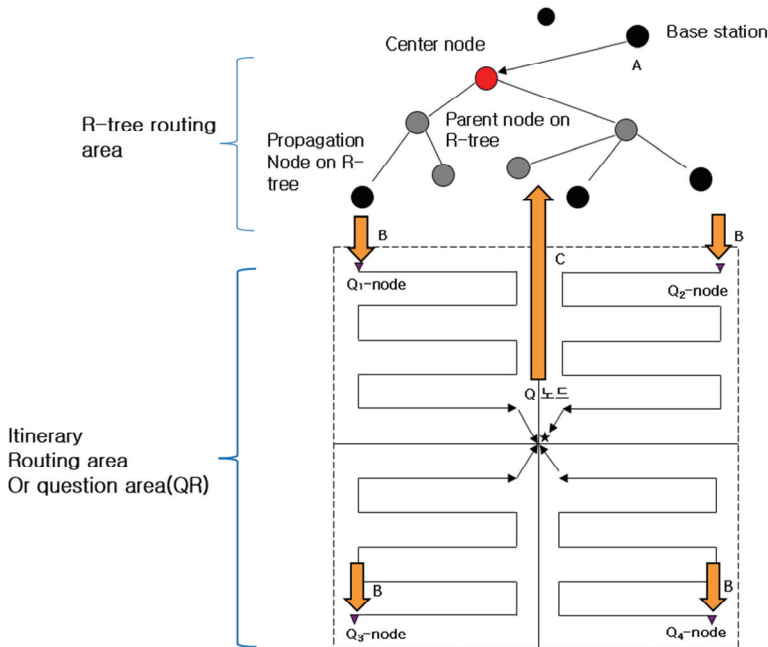


Fig. 20. GPSR after quad-tree application.

That is, since the sensor node region (SR) is known in advance in the period A, the center point of the SR can be calculated. From the BS to the center point, proceed to the GPSR algorithm and select the node closest to the center point. In section B, a query is transmitted simultaneously to the GPSR algorithm as a pre-selected representative node in each cell of the quad tree at the transfer node on the R-tree. At this time, the representative node of each cell becomes the first Q-node. In other words, as the non-end node, the sensor node closest to the cell center is selected as the representative sensor node of each cell, and as the terminal node, the sensor node closest to the edge in each cell is selected as the representative sensor node. After each itinerary query is processed for each cell, the query result is transmitted along the quad-tree query processing path. In section C, the last node, the central node, collects the query and sends it to the parent node on its R-tree.

3.3 Merging Spatio-temporal Qualities

The following is an example of merging space(region), attribute, and time for a spatio-temporal query, and query Q consists of <region, attribute, time, etc.>.

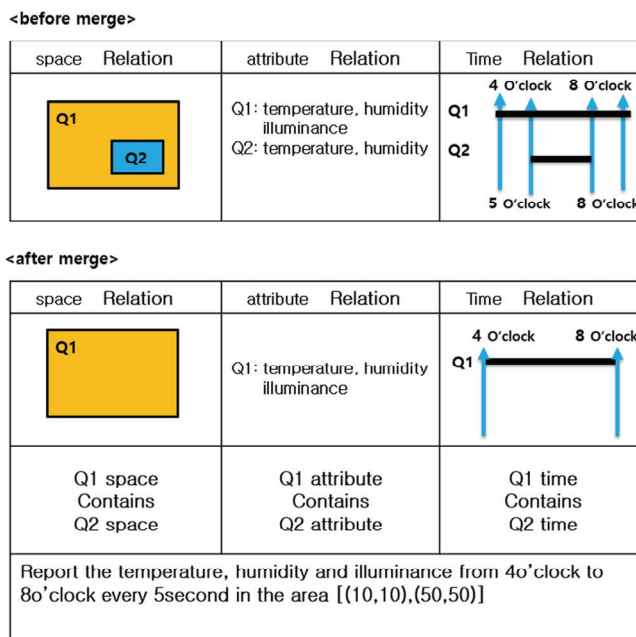


Fig. 21. Contains merge example.

Fig. 21 illustrates examples before and after the merger of Contains. That is, an example of merging of space, attribute, and time for two queries is shown. The Q1 query transmits the Q1 query in a relationship that completely contains the Q2 query.

Q1: Report temperature, humidity, and illumiance every 5 seconds from 4 to 8 in the area [(10,10), (50,50)].

Q2: Report the temperature and humidity every 10 seconds from 5:00 to 7:00 in the area [(20,20), (40,30)].

Fig. 21 illustrates the example before and after merging into overlap-space. That is, an example of merging of spaces and attributes for three queries is shown. A new query, Q4, is transmitted by combining the attributes including the space including the Q1, Q2, and Q3 spaces.

Q1: Report the temperature and humidity in the area [(10,10), (30,40)].

Q2: Report the temperature and illumination in the area [(20,30), (40,50)].

Q3: Report illumination and humidity in the area [(35,40), (50,60)].

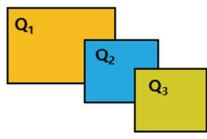

<before merge>			
space	Relation	attribute	Relation
		<p>Q₁: temperature, humidity</p> <p>Q₂: temperature, illuminance</p> <p>Q₃: illuminance, humidity</p>	
<after merge>			
space	Relation	attribute	Relation
		<p>Q₄: temperature, humidity, illuminance</p>	
<p>Q₄ space with Q₁, Q₂, Q₃ space together</p>		<p>Q₄ attribute with Q₁, Q₂, Q₃ attribute together</p>	
<p>Q₄: Report the temperature, humidity and illuminance in the area[(10,10), (50,60)]</p>			

Fig. 22. Overlap spatial merge example.

Fig. 22 illustrates the example before and after merging for overlap-time. That is, an example of merging time and attribute for three queries is shown. Q1, Q2, and Q3, and a new query Q4 that combines attributes and time.

Q1: Report temperature and humidity from 4 to 6 o'clock.

Q2: Report temperature and illuminance from 5 to 8 o'clock.

Q3: Report illuminance and humidity from 8:00 to 9:00.

Fig. 23 describes the structure of IR-tree index, query propagation, query result reception, and index update. The area is distinguished by a fixed sensor network area (sensor network range) and a query area (BS) that is determined when a query is merged. The query area is an itinerary area and a quad tree constituent area. The index configuration is a process of constructing a quad-tree by selecting the C-node, determining the parent node, constructing the R-tree, determining dynamic routing boundary value, and propagating the query area. The propagation of spatio-temporal queries and the reception of query results consist of spatio-temporal merging in BS, query propagation in R-tree, and dynamic routing. Index updating requires updating at a certain index level or more, but many nodes below a certain index level are not subject to dynamic updating, and index updating is unnecessary.

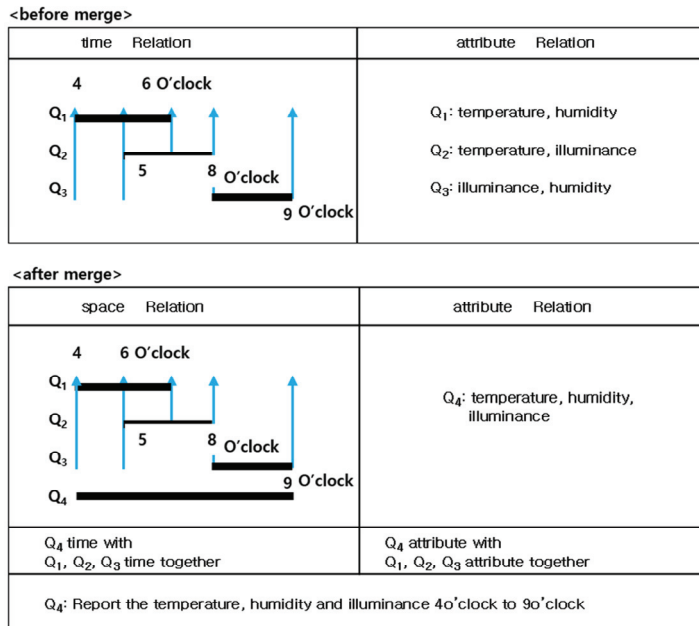


Fig. 23. Overlap time merge example.

4. Performance Evaluation

4.1 Performance Evaluation according to the Number of Sensor Nodes

In this section, performance evaluation according to the number of sensor nodes is described. We evaluate the accuracy of query processing results, energy consumption, and query processing time for existing research and IR-tree for query processing according to the number of sensor nodes for Uniform distribution, Gauss distribution, and Skew distribution.

In addition, the IR-tree Itinerary level of 80%–85% was the best performance in this evaluation. However, itinerary level was set to 70% in order to maintain accuracy over 95%.

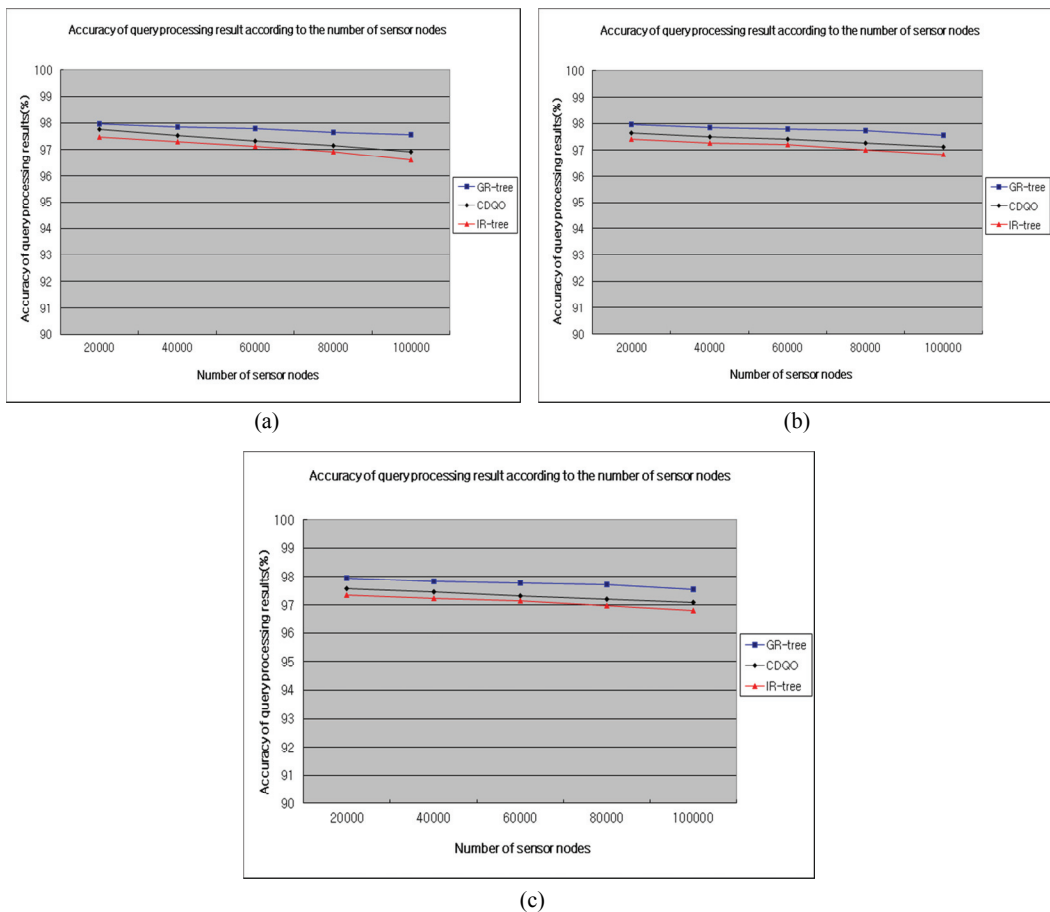
In the performance evaluation according to the number of sensor nodes, the number of continuous queries is 600, the size of query area is 990×990 m², and the number of sensor nodes is increased from 20,000 to 100,000 to perform performance evaluation.

For the performance evaluation, the size of the sensor network, the communication range of the sensor node, the total number of transmissions of the sensor node, the total energy amount of the sensor node, the energy consumption in a single data transmission, a single data transmission size, the number of sensor nodes, the number of continuous queries, and the size of the query area are set as parameters in Table 1.

In the performance evaluation according to the number of sensor nodes, the accuracy of query processing results of GR-tree, CDQO, and IR-tree are compared in terms of energy consumption and query processing time. Fig. 24 shows the performance evaluation of the query processing accuracy according to the number of nodes.

Table 1. Parameters for performance evaluation

Parameters	Value
Sensor network size (m ²)	1,650×1,650
Communication range of sensor node (m)	Within 30
Total transfer count of sensor node	10,000,000
Total amount of energy of sensor node (J)	10,000
Energy consumption during data transmission (mJ)	1
Size during data transmission (byte)	30
Time during data transmission (ms)	0.12
Error rate during data transmission	10% (once every 10 times)
Sensing range of sensor node (°C)	70 (-10 to 60)
Deviation range of sensing data (°C)	0.5 (-0.5 to 0.5)
Number of sensor nodes (million)	2, 4, 6, 8, 10
Number of continuous queries	200, 400, 600, 800, 1,000
Size of query region (m ²)	330×330, 660×660, 990×990, 1,320×1,320, 1,650×1,650

**Fig. 24.** Accuracy evaluation of query processing result according to number of sensor nodes with (a) Uniform, (b) Gauss, and (c) Skew distributions.

As shown in Fig. 24, the performance of the CDQO is lower than that of the GR-tree, and the IR-tree is less accurate than the CDQO. In particular, this performance difference is more significant when the nodes are Gaussian and Skew distributions.

Fig. 25 shows the energy consumption performance evaluation according to the number of sensor nodes.

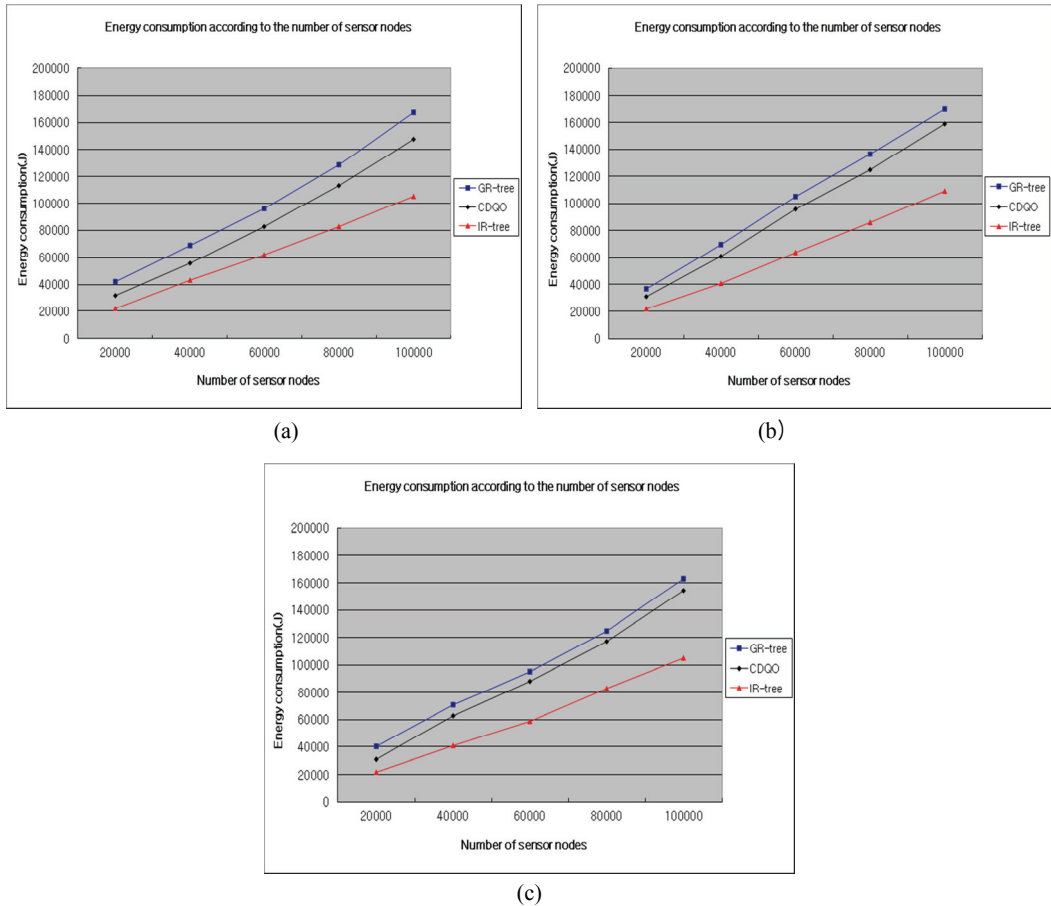


Fig. 25. Evaluation of energy consumption performance according to the number of sensor nodes with (a) Uniform, (b) Gauss, and (c) Skew distributions.

As shown in Fig. 25(a), IR-tree and CDQO improve performance by 58% and 35% on average, respectively, compared to GR-tree. As shown in Fig. 25(b), IR-tree improves performance by 57% on average and 44% on CDQO compared to GR-tree. As shown in Fig. 25(c), the IR-tree improves the performance by 60% on average and 45% on average compared to the GR-tree in the case of Skew.

Fig. 26 shows the query processing time evaluation result according to the number of sensor nodes.

As shown in Fig. 26(a), the IR-tree and CDQO improve performance by 55% and 25% on average, respectively, compared to GR-tree. As shown in Fig. 26(b), the IR-tree and CDQO improve performance by 64% and 26% on average, respectively, compared to the GR-tree. As shown in Fig. 26(c), the IR-tree and CD-tree improves performance by 66% and 27% on average, respectively, compared to the GR-tree.

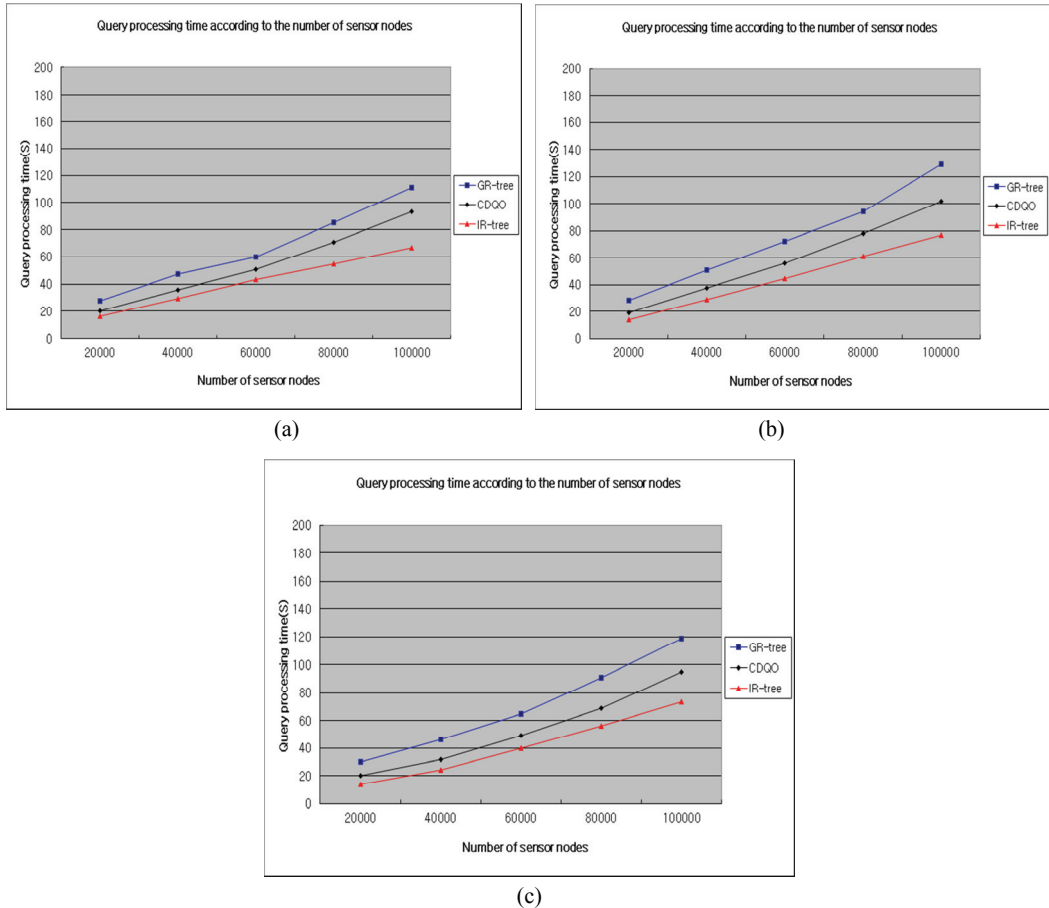


Fig. 26. Performance evaluation of query processing time according to the number of sensor nodes with (a) Uniform, (b) Gauss, and (c) Skew distributions.

The accuracy of query processing is low because the IR-tree merges spatio-temporal queries. However, since IR-tree uses itinerary based on dynamic routing scheme from a certain level, the energy consumption of all sensor nodes is lower than that of GR-tree and CDQO, and query processing time shows fast performance results. In particular, the greater the number of wireless sensor nodes, the greater the difference in performance.

5. Conclusion

This paper proposes an IR-tree for more efficient query processing in wireless sensor networks. In IR-tree generation, a C-node is first selected according to the distribution of wireless sensor nodes in a specific sensor space region, and an R-tree based on MBR is constructed. The query area is then configured as a quad-tree to improve accuracy and search speed. R-tree, and the MBR of an upper node, is a hierarchical tree structure including MBRs of lower nodes. When query processing in the index structure, a specific upper level performs R-tree search, and the Itinerary technique is used from the lowest level.

A minimum route is searched from S-node to C-node using a GPSR algorithm. In the search process, a sink or node (query merging) algorithm is used to perform spatial or temporal space, the number of queries, and property merging, and then the query is transmitted from the C-node to the R-tree. We have extended the existing spatial query merging in the sink node by space-time merging, and have proved that space-time query is superior to conventional space query processing, with less energy consumption and faster query processing time.

When the query enters the quad tree cell in the itinerary region of the R-tree, the first Q-node of each cell is searched using the GPSR algorithm, and the association between the final Q-node and the R-tree at the time of collecting the query result uses the parent node of the pre-configured R-tree. In addition, IR-tree needs periodic R-tree reorganization at a certain upper level in IR-tree when adding and removing sensor nodes. However, since most of the lower level is composed of itinerary technique, it is advantageous that the update is unnecessary.

This paper proposes a new method for analyzing query results (accuracy, energy consumption, query processing time) of GR-tree, CDQO algorithm, and IR-tree, which are existing spatial query processing techniques, through experiments using sensor data; with Uniform and Gauss distributions the performance of the proposed IR-tree based algorithm is proved to be superior.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2017R1A2B4011243).

References

- [1] K. T. Kim and H. Y. Youn, "Tree-based clustering protocol for energy-efficient wireless sensor networks," *The KIPS Transactions: Part C*, vol. 17, no. 1, pp. 69-80, 2010.
- [2] M. Lee and V. W. S. Wong, "An energy-aware spanning tree algorithm for data aggregation in wireless sensor networks," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, Canada, 2005, pp. 300-303.
- [3] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Network*, vol. 3 no. 3, pp.325-349, 2005.
- [4] V. S. Felix Enigo and V. Ramachandran, "Effective management of high rate spatio-temporal queries in Wireless Sensor Networks," *Wireless Personal Communications*, vol. 79, no. 2, pp. 1111-1128, 2014.
- [5] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+-Tree: a dynamic index for multi-dimensional objects," in *Proceedings of the 13th International Conference on Very Large Data Bases*, Brighton, England, 1987, pp. 507-518.
- [6] A. Guttman, "R-Trees: a dynamic index structure for spatial searching," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Boston, MA, 1984, pp. 47-57.
- [7] M. S. Kim and I. S. Jang, "The GR-tree: an energy-efficient distributed spatial indexing scheme in wireless sensor networks," *Journal of Korean Spatial Information Society*, vol. 19, no. 5, pp. 63-74, 2011.
- [8] J. J. Kim, I. S. Shin, K. Y. Lee, and K. J. Han, "Efficient Processing of aggregate queries in wireless sensor networks," *Journal of Korean Spatial Information Society*, vol. 19, no. 3, pp. 95-106, 2011.



Jeong-Joon Kim <https://orcid.org/0000-0002-0125-1907>

He received his B.S. and M.S. in Computer Science at Konkuk University in 2003 and 2005, respectively. In 2010, he received his Ph.D. in at Konkuk University. He is currently a professor at the department of Computer Science at Korea Polytechnic University. His research interests include database systems, big data, semantic web, geographic information systems (GIS) and ubiquitous sensor network (USN), etc.