Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2012, Article ID 625798, 15 pages
doi:10.1155/2012/625798

Research Article

Aggregate Queries in Wireless Sensor Networks

Jeong-Joon Kim,[1] In-Su Shin,[1] Yan-Sheng Zhang,[2] Dong-Oh Kim,[3] and Ki-Joon Han[1]

[1] *Division of Computer Science & Engineering, Konkuk University, Seoul 143-701, Republic of Korea*
[2] *Division of Software Engineering, Northeastern University, Shenyang 110819, China*
[3] *Cloud Computing Research Department, Electronics and Telecommunications Research Institute, Daejeon 305-700, Republic of Korea*

Correspondence should be addressed to Ki-Joon Han, kjhan@db.konkuk.ac.kr

Received 19 February 2012; Revised 25 May 2012; Accepted 25 June 2012

Academic Editor: Jianliang Xu

Recently as efficient processing of aggregate queries for fetching desired data from sensors has been recognized as a crucial part, in-network aggregate query processing techniques are studied intensively in wireless sensor networks. Existing representative in-network aggregate query processing techniques propose routing algorithms and data structures for processing aggregate queries. However, these aggregate query processing techniques have problems such as high energy consumption in sensor nodes, low accuracy of query processing results, and long query processing time. In order to solve these problems and to enhance the efficiency of aggregate query processing in wireless sensor networks, this paper proposes Bucket-based Parallel Aggregation (BPA). BPA divides a query region into several cells according to the distribution of sensor nodes and builds a quadtree, and then processes aggregate queries in parallel for each cell region according to routing. It sends data in duplicate by removing redundant data, which, in turn, enhances the accuracy of query processing results. Also, BPA uses a bucket-based data structure in aggregate query processing, and divides and conquers the bucket data structure adaptively according to the number of data in the bucket. In addition, BPA compresses data in order to reduce the size of data in the bucket and performs data transmission filtering when each sensor node sends data. Finally, in this paper, we prove its superiority through various experiments using sensor data.

1. Introduction

With the rapid advance of sensing technologies for capturing various types of data such as temperature, humidity, and pressure as well as the development of wireless communication technologies, research is being made actively for utilizing wireless sensor network technologies in diverse application areas including military, medicine, meteorology, environment, transportation, home, and business [1, 2].

Generally sensor nodes do not use unicasting (use ACK) whsen they regularly send the sensed data. Rather, they multicast or broadcast to the sensor nodes within the scope of the communication [3]. Also, sensor nodes basically know the content of the query that S-node (the starting node) sent, effective time of the query (from the reception of the query to the transmission of the first sensed data), and the cycle of the query (interval to transmit the sensed data).

In particular, the aggregate query process, which is to obtain aggregate results from data collected by sensors, is recognized as an important research area [4]. Aggregate queries execute functions such as MAX, MIN, SUM, AVG, COUNT, MEDIAN, and HISTOGRAM on the entire wireless sensor network or a specific region of the network.

Conventional centered aggregate query processing techniques have the problem of high energy consumption by the sensor nodes. Thus, in order to reduce the energy consumption of sensor nodes, aggregate query processing in network is being studied actively, which processes aggregate queries on sensed data in the sensor nodes and then sends the results to the server [5–7]. Representative techniques of aggregate query processing in network include TAG (Tiny AGgregation) and IWQE (Itinerary-based Window Query Execution) that focus on routing algorithm, and q-digest (quantile digest) and SMC (Secure Median Computation)

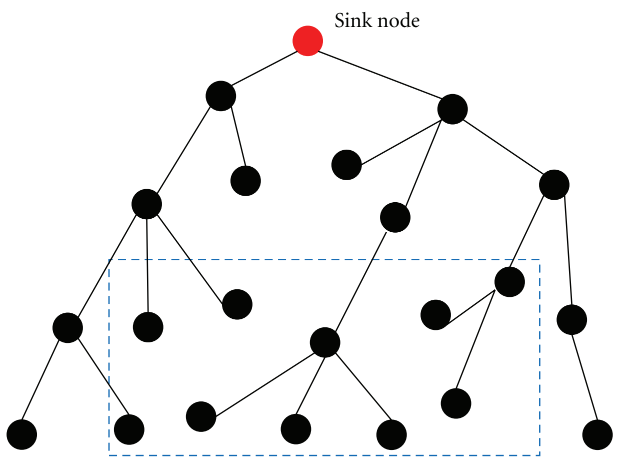


Figure 1: Hierarchical routing structure of TAG.

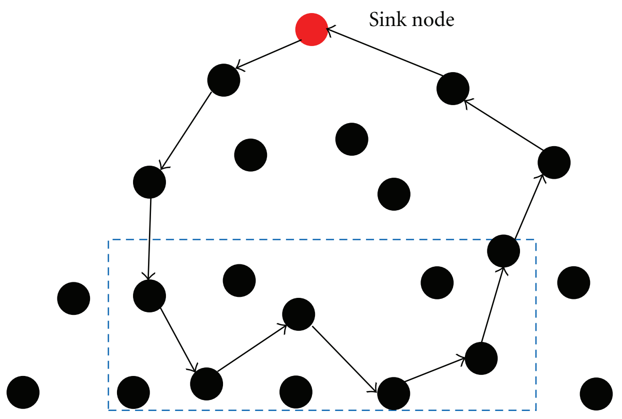


Figure 2: Itinerary routing structure of IWQE.

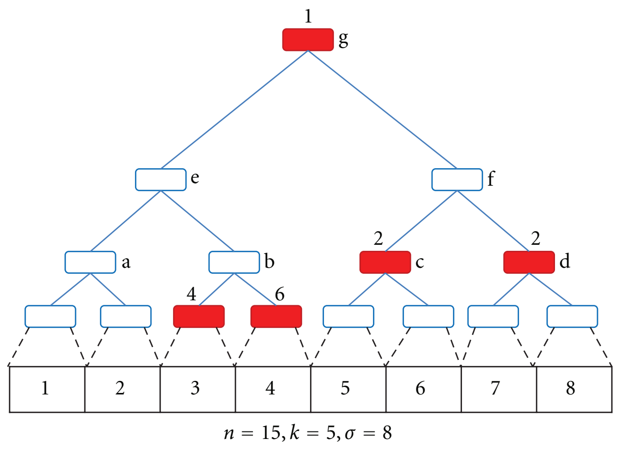


Figure 3: Tree data structure of q-digest.

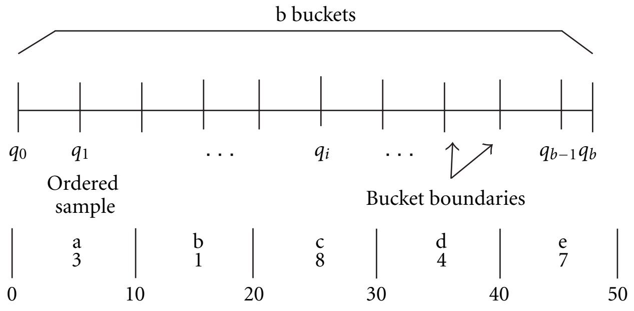


Figure 4: Bucket data structure of SMC.

that focus on data structure [8–11]. TAG is an aggregate query processing technique using hierarchical routing [8] and IWQE is an aggregate query processing technique using itinerary routing [11]. Although TAG, and IWQE propose routing algorithm for efficient aggregate query processing, they have problems such as high energy consumption by the sensor nodes, low accuracy of query processing results, and long query processing time. Moreover, TAG, and IWQE have the shortcoming that they cannot consider aggregate operations MEDIAN and HISTOGRAM.

Q-digest is an approximate aggregate query processing technique using tree data structure for aggregate operations MEDIAN and HISTOGRAM [10, 12], and SMC is an approximate aggregate query processing technique using bucket data structure for aggregate operations MEDIAN and HISTOGRAM [9]. In this way, q-digest, and SMC suggest data structures for efficient aggregate query processing but they still have problems such as high energy consumption by the sensor nodes, low accuracy of query processing results, and long query processing time. Moreover, q-digest, and SMC do not consider composite types of aggregate queries (executing two or more aggregate queries at the same time).

In order to solve these problems in existing aggregate query processing techniques and to enhance the efficiency of aggregate query processing in wireless sensor networks, this study proposed aggregate query processing technique BPA (bucket-based parallel aggregation). BPA collects information on sensor nodes within a query region, divides the query region into multiple cells according to the distribution of sensor nodes, builds a quad-tree using the cells, and processes an aggregate query in parallel according to itinerary routing for the cell coverage of the quad-tree nodes. Because BPA processes an aggregate query in parallel, the sensor nodes consume less energy and query processing time is short even if the query region is wide or the number of sensor nodes is large.

Moreover, BPA minimizes the number of missing nodes, which cannot participate in aggregate query processing, among the sensor nodes of the query region, and on the occurrence of a missing node, it sends data to the closest node to the sensor node that started the query. Moreover, the sensor nodes of BPA send data double, to reduce data loss resulting from transmission errors. As it minimizes missing nodes and sends data double after removing redundant data, BPA can enhance the accuracy of query processing results.

BPA uses bucket-based data structure for aggregate operations MEDIAN and HISTOGRAM. The bucket data structure stores the minimum and maximum values of collected data, the mean value of data in the bucket, and the

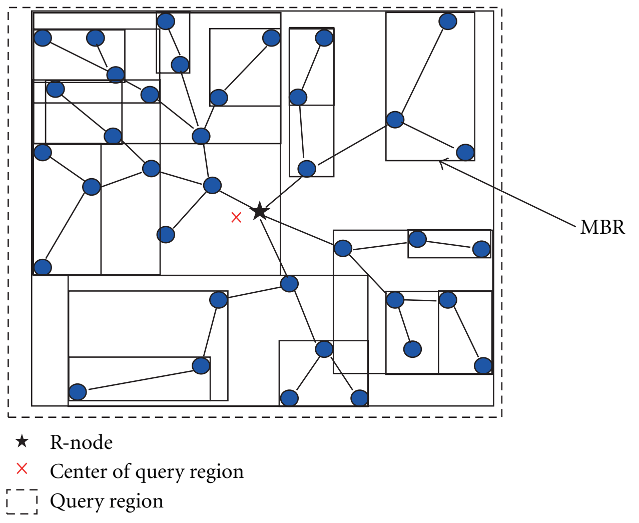


Figure 5: Hierarchical routing structure and example of MBR structure.

number of data in the bucket in consideration of composite types of aggregate queries. Moreover, it compresses data using the variable bit compression-coding technique [13] in order to reduce the size of data in the bucket. Using the bucket data structure and the variable bit compression coding technique, BPA can reduce the energy consumption of sensor nodes.

Because BPA divides and merges the bucket data structure adaptively according to the number of data in the bucket, it can enhance the accuracy of query-processing results even if data distribution is uneven. Moreover, BPA performs data transmission filtering, which sets a filtering range in each sensor node and sends data only when the data are outside the range, and this reduces the energy consumption of sensor nodes.

2. Related Works

2.1. TAG (Tiny AGgregation). TAG is a technique of aggregate query processing in network that uses hierarchical routing for aggregate query processing [8]. That is, TAG establishes hierarchical routing for the entire wireless sensor network in order to process aggregate queries in the network. Figure 1 shows the hierarchical routing structure of TAG.

As in Figure 1, TAG establishes hierarchical routing by defining parent-child relations among all the sensor nodes. A child sensor node in the query region sends sensed data to its parent sensor node, which sends intermediate aggregate query results to its parent sensor node. At last, the sink node returns the final results of aggregate query processing to the server.

TAG reduces overall energy consumption by sensor nodes through processing aggregate queries within the wireless sensor network instead of centralized processing [5, 11]. Thus, TAG is efficient when the query region is wide or the number of sensor nodes is large. However, there is high energy consumption by sensor nodes not included in the query region, and a large amount of energy is consumed by sensor nodes in order to maintain routing [3, 11]. In addition, TAG has the shortcoming that aggregate operations MEDIAN and HISTOGRAM are not considered [9, 10, 12].

2.2. IWQE (Itinerary-Based Window Query Execution). IWQE is a technique of aggregate query processing in network that uses itinerary routing for aggregate query processing [11]. IWQE processes aggregate queries by establishing temporary routing for the region of interest when a user query is given instead of establishing routing for the entire region of wireless sensor network. Figure 2 shows the itinerary routing structure of IWQE.

As in Figure 2, IWQE processes an aggregate query for data sensed by sensor nodes within the query region using itinerary routing, and the sink node returns the final result of the query to the server.

In IWQE, there is no unnecessary energy consumption by sensor nodes not included in the query region and no cost of routing maintenance. Furthermore, it is efficient when the query region is narrow or the number of sensor nodes is small [5]. However, the accuracy of query results is low due to the occurrence of missing nodes, and redundant data arising from broadcasting transmission for reducing data transmission errors impairs the accuracy of query processing results and increases energy consumption by the sensor nodes. Moreover, it takes a long time to process queries if the number of sensor nodes is large or the query region is wide [3]. Moreover, IWQE does not consider aggregate operations MEDIAN and HISTOGRAM [9, 12].

2.3. Q-digest (Quantile-Digest). Q-digest is an approximate aggregate query processing technique using tree data structure in order to process aggregate operations MEDIAM and HISTOGRAM [10, 12]. The tree data structure of q-digest has characteristics as follows. The root node has a range value of $[1, \sigma]$, and its child nodes have range values of $[1, \sigma/2]$ and $[\sigma/2+1,\ \sigma]$, respectively. In addition, each node stores the number of sensing data included in the range. Figure 3 is the tree data structure of q-digest and an example.

As in Figure 3, it is assumed that the whole range of sensing data is 1–8, and number of data is 15, and compression rate k is 5. Accordingly, root node g has a range value of (1, 8), and its child nodes e and f have range values of (1, 4) and (5, 8), respectively. In addition, child nodes a, b, c, and d have range values of (1, 2), (3, 4) and (5, 6) and (7, 8), respectively. Moreover, each node stores the number of sensed data that belong to its range. In particular, each node of q-digest compresses data in case the sum of the number of data in the node and the number of data in its parent node and its neighbor node is smaller than compression rate n/k.

As q-digest processes aggregate queries using tree data structure, it reduces the amount of data transmission and, consequently, shows high performance in aggregate operations MEDIAN and HISTOGRAM. However, the energy consumption of sensor nodes is high due to additional information for building the tree data structure, and data compression lowers the accuracy of aggregate query results. Moreover, because the range values of nodes are fixed, if sensed data are not distributed evenly, the results of aggregate

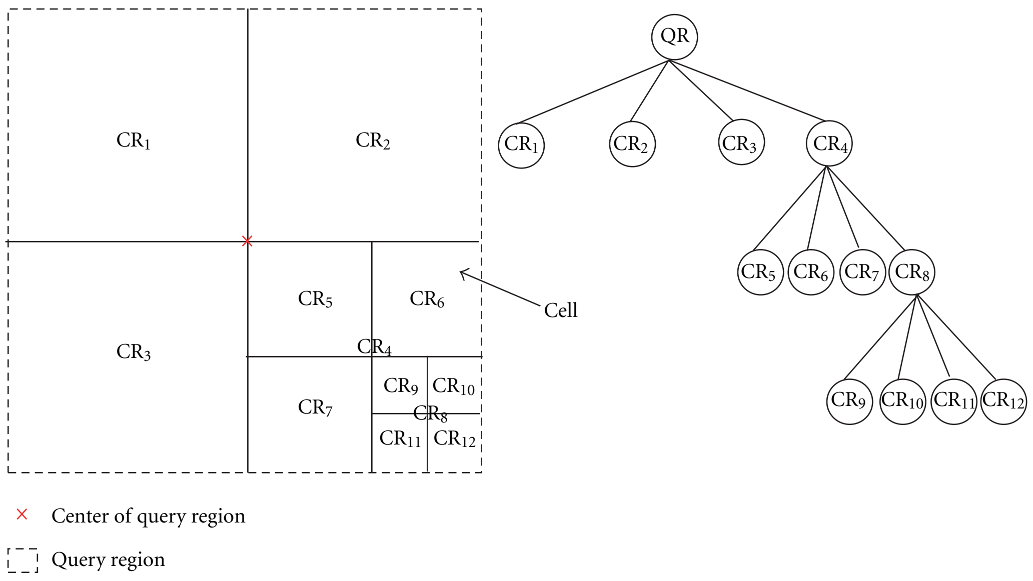


Figure 6: Example of quad-tree structure.

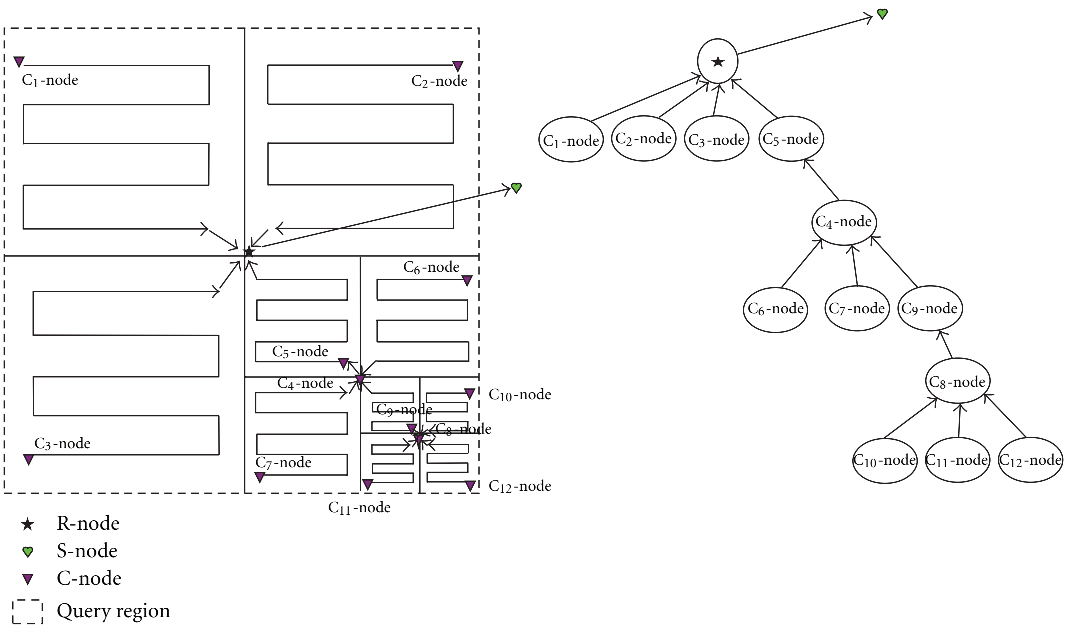


Figure 7: Transfer process of query processing results.

query processing become less accurate [9, 14]. What is more, q-digest does not consider composite types of aggregate queries.

2.4. SMC (Secure Median Computation). SMC is an approximate aggregate query processing technique that uses bucket data structure for aggregate operations MEDIAM and HISTOGRAM [9]. The bucket data structure of SMC has characteristics as follows. SMC forms b buckets to store sensed data, and each bucket B_i has range value q_i–q_{i+1} and stores the number of sensed data. In particular, SMC divides or merges buckets in which specific aggregate values are stored and, by doing so, it enhances the accuracy of aggregate query processing results in the next query. Figure 4 is the bucket data structure of SMC and an example.

As in Figure 4, the bucket data structure has the entire data range of 0–50, and consists of 5 buckets. Bucket a has range value 0–10, b 11–20, c 21–30, d 31–40, and e 41–50, and the buckets store the numbers of sensed data 3, 1, 8, 4 and 7, respectively.

SMC processes aggregate queries using bucket data structure and, by doing so, it reduces the amount of data transmission and shows high performance in aggregate operations MEDIAN and HISTOGRAM. However, the energy consumption of sensor nodes is high due to the transmission of fixed bucket data structure, and the accuracy of aggregate

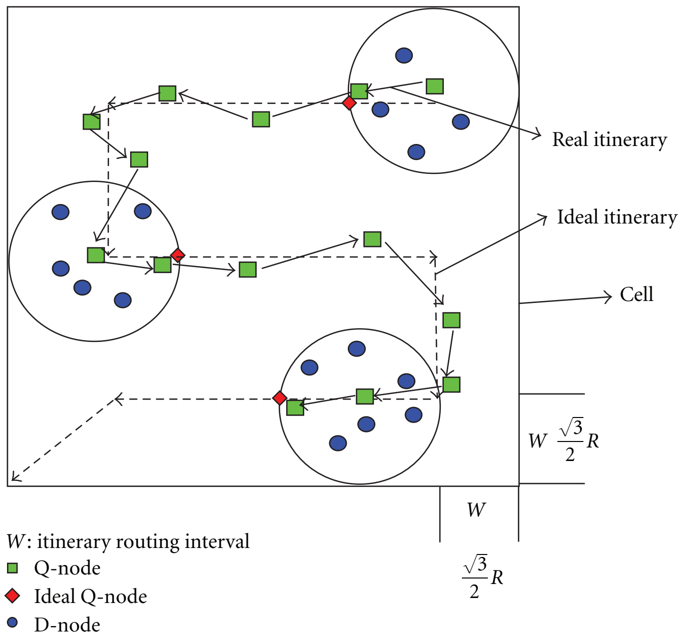


FIGURE 8: Example of routing process in quad-tree cells.

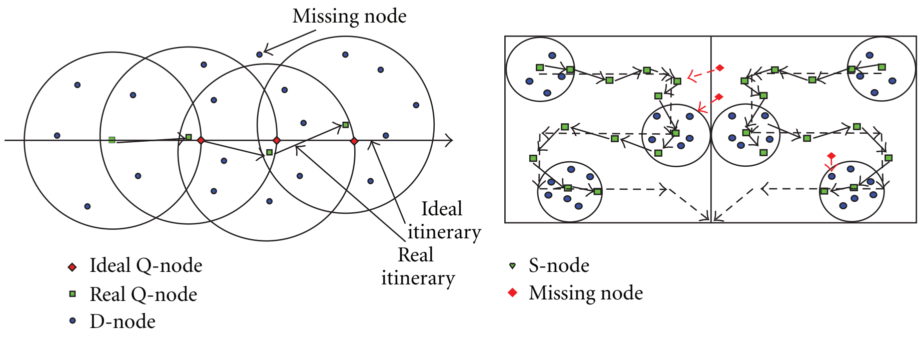


FIGURE 9: Example of missing node process.

query processing results is low due to the merger of bucket data [14]. Moreover, SMC does not consider composite types of aggregate queries.

3. BPA (Bucket-Based Parallel Aggregation)

3.1. Routing. BPA establishes hierarchical routing and collects sensor node information in order to reduce energy consumption by sensor nodes and query processing time. Then, using collected sensor node information, it divides the query region into a number of cells according to the distribution of sensor nodes, builds a quad-tree with the cells, and processes an aggregate query in parallel on the cell coverage of the quad-tree through the itinerary routing. Figure 5 shows the hierarchical routing structure and an example of MBR structure for collecting sensor node information.

As in Figure 5, the closest sensor node to the center of the query region is searched for, and the node is used as R-node (root node) of hierarchical routing to be established. Starting from R-node, a sensor node with child nodes defines MBR (minimum boundary rectangle) that includes itself and its child sensor nodes, collects information on the sensor nodes within the MBR, and sends the data to its parent sensor node.

Using the collected sensor node information, BPA divides the query region into a number of cells and builds a quad-tree with the cells. Figure 6 shows an example of quad-tree structure.

As in Figure 5, the closest sensor node to the center of the query region is searched for, and the node is used as R-node (root node) of hierarchical routing.

As in Figure 6, query region QR is divided into CR_1, CR_2, CR_3 and CR_4, and CR_4 is again subdivided into CR_5, CR_6, CR_7, and CR_8, and again CR_8 into CR_9, CR_{10}, CR_{11}, and CR_{12}. Each cell is subdivided until the number of sensor nodes in the cell becomes smaller than the threshold maximum number of sensor nodes in each cell.

BPA selects C-node, which is the representative sensor node, within each quad-tree cell and processes an aggregate query in parallel for the coverage of each cell in the quad-tree. The result of aggregate query processing for the coverage of each cell is transmitted recursively to the representative

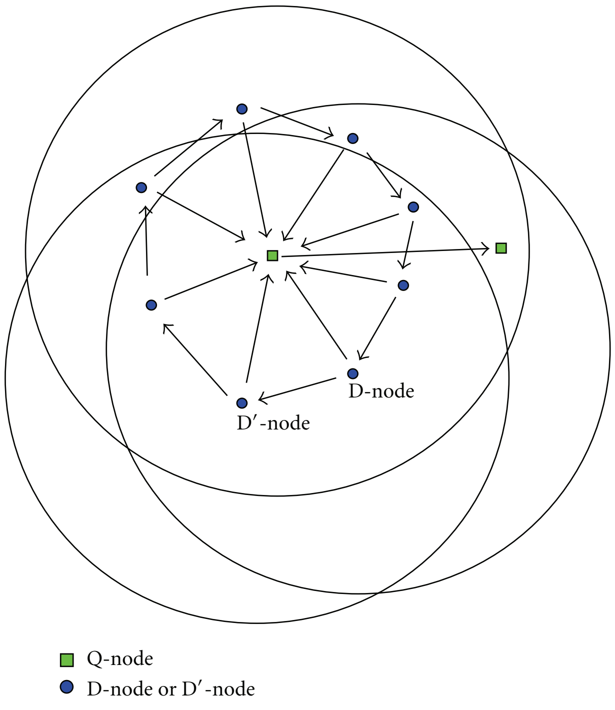


Figure 10: Example of double data transmission of D-node.

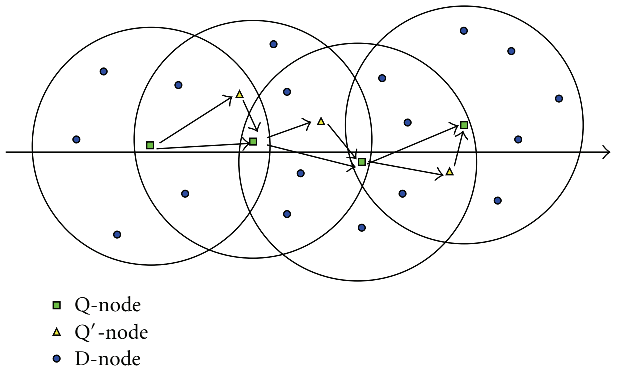


Figure 11: Example of double data transmission of Q-node.

sensor node of the parent node cell. Figure 7 shows the recursive process transmitting the result of aggregate query processing to the representative sensor node of the parent node cell.

As in Figure 7, the results of aggregate query processing in C_{10}-node, C_{11}-node, and C_{12}-node are transmitted to C_8-node, and the result of aggregate query processing in C_8-node is transmitted to C_9-node. In addition, the results of aggregate query processing in C_6-node, C_7-node, and C_9-node are sent to C_4-node, and the result of aggregate query processing in C_4-node is sent to C_5-node. Lastly, the results of aggregate query processing in C_1-node, C_2-node, C_3-node, and C_5-node are transmitted to R-node, and the result of aggregate query processing in R-node is returned to S-node, the sensor node that started the query.

BPA uses itinerary routing in order to process aggregate queries in quad-tree cells. Figure 8 shows an example of routing process in quad-tree cells.

As in Figure 8, Q-node, which is the query transmission sensor $(\sqrt{3}/2)R$ node within each cell, collects data from D-nodes, which are data transmission sensor nodes within the communication range, through the ideal itinerary routing, processes an aggregate query, and sends the result to the next Q-node. At that time, the actual routing path of Q-nodes is the real itinerary routing and each itinerary routing interval W is set as using sensor nodes' communication range R.

In order to minimize the number of missing nodes not participating in aggregate query processing in the itinerary routing process, BPA selects the closest sensor node to Ideal Q-node among D-nodes within the communication range of Q-node as the next Q-node. Ideal Q-node is a virtual sensor node that does not exist but is set for optimal routing, and means the intersecting point between the line of the ideal itinerary routing and the communication range of Q-node.

Sensor nodes basically know the content of the query that S-node (the starting node) sent, effective time of the query (from the reception of the query to the transmission of the first sensed data), and the cycle of the query (interval to transmit the sensed data). Therefore if it were not selected as Q-node or D-node within the effective time of the query, it would perceive it as a missing node.

If a missing node occurs, it is processed as follows. If a sensor node finds itself to be a missing node because it has not been selected as a Q-node or a D-node within a query valid time, it sends its data to the closest sensor node to S-node.

Figure 9 shows an example of Missing node process.

As shown in Figure 9, a missing node calculates the distance using the location information of S-node and the sensor nodes within the communication scope and selects the nearest sensor node from S-node.

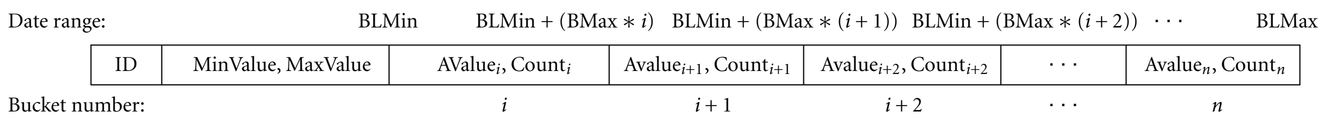
D-nodes and Q-nodes in BPA perform double data transmission among sensor nodes in order to reduce errors in the results of aggregate query processing caused by network transmission errors. Figure 10 shows an example of double data transmission process by a D-node in BPA.

As in Figure 10, a D-node sends sensed data to the Q-node and its neighbor D′-node, which is another D-node. That is, the D-node sends data to both the Q-node and D′-node, which is one of its neighbor sensor nodes, includes the Q-node within its communication range, and satisfies the right-hand rule. Then, D′-node sends data from the D-node and data sensed by itself to the Q-node.

Figure 11 shows an example of double data transmission process by a Q-node in BPA.

As in Figure 11, a Q-node sends data collected from D-node to the next Q-node and its neighbor Q′-node. That is, the Q-node sends data to both the next Q-node and Q′-node, which is one of its neighbor sensor nodes, includes the next Q-node within its communication range, and is the closest to the next Q-node. Then, Q′-node sends data from the Q-node and data sensed by itself to the next Q-node.

In BPA, when a sensor node sends data, it adds its ID and sends the data double, and the sensor node that has received



BLMin: minimum range of bucket list
BLMax : maximum range of bucket list
BMax : maximumn size of bucket = initial size of bucket

Figure 12: Bucket data structure of Q-node.

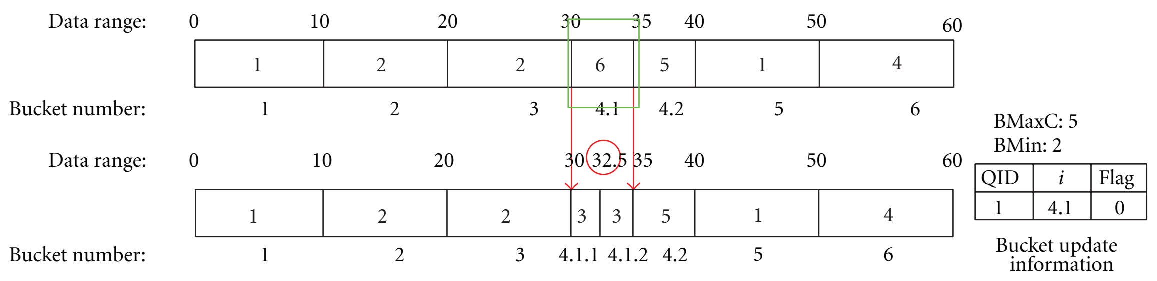

Data range: 0 10 20 30 35 40 50 60
1 2 2 6 5 1 4
Bucket number: 1 2 3 4.1 4.2 5 6
Data range: 0 10 20 30 32.5 35 40 50 60
1 2 2 3 3 5 1 4
Bucket number: 1 2 3 4.1.1 4.1.2 4.2 5 6
BMaxC: 5
BMin: 2
QID i Flag
1 4.1 0
Bucket update information

Figure 13: Example of bucket division.

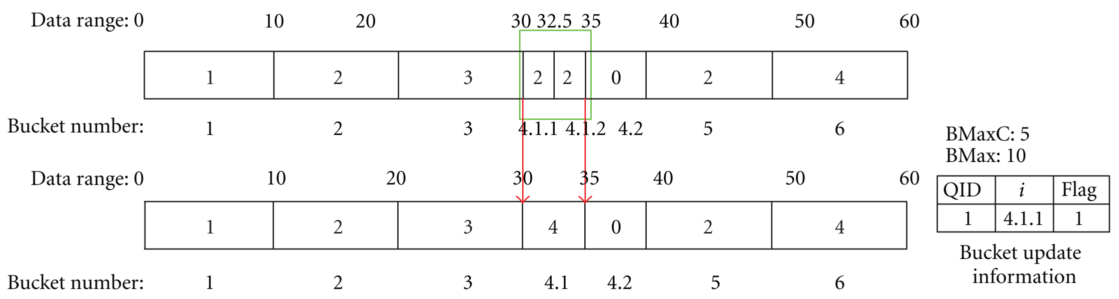

Data range: 0 10 20 30 32.5 35 40 50 60
1 2 3 2 2 0 2 4
Bucket number: 1 2 3 4.1.1 4.1.2 4.2 5 6
Data range: 0 10 20 30 35 40 50 60
1 2 3 4 0 2 4
Bucket number: 1 2 3 4.1 4.2 5 6
BMaxC: 5
BMax: 10
QID i Flag
1 4.1.1 1
Bucket update information

Figure 14: Example of bucket merge.

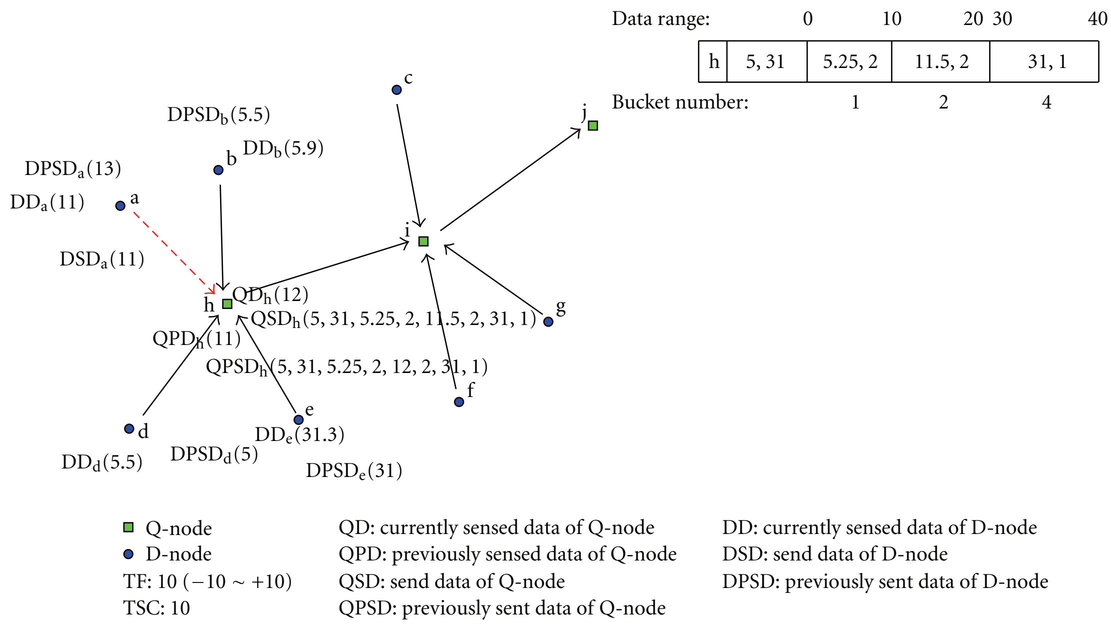

Data range: 0 10 20 30 40
h 5, 31 5.25, 2 11.5, 2 31, 1
Bucket number: 1 2 4
c
j
DPSD_b(5.5)
DD_b(5.9)
b
DPSD_a(13)
DD_a(11)
a
DSD_a(11)
i
QD_h(12)
h
QSD_h(5, 31, 5.25, 2, 11.5, 2, 31, 1)
g
QPD_h(11)
QPSD_h(5, 31, 5.25, 2, 12, 2, 31, 1)
f
e
d
DD_e(31.3)
DD_d(5.5)
DPSD_d(5)
DPSD_e(31)
🟩 Q-node
● D-node
TF: 10 (−10 ∼ +10)
TSC: 10
QD: currently sensed data of Q-node
QPD: previously sensed data of Q-node
QSD: send data of Q-node
QPSD: previously sent data of Q-node
DD: currently sensed data of D-node
DSD: send data of D-node
DPSD: previously sent data of D-node

Figure 15: Example (1) of data transmission filtering.

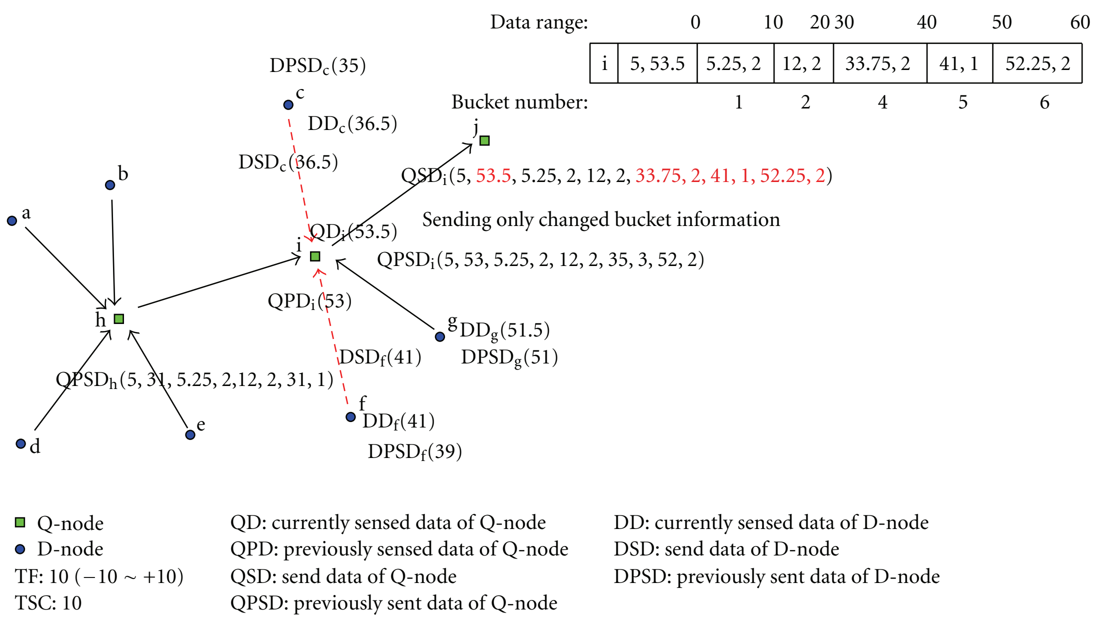

Data range: 0 10 20 30 40 50 60
i | 5, 53.5 | 5.25, 2 | 12, 2 | 33.75, 2 | 41, 1 | 52.25, 2
Bucket number: 1 2 4 5 6
DPSD_c(35)
c
DD_c(36.5)
DSD_c(36.5)
j
QSD_i(5, 53.5, 5.25, 2, 12, 2, 33.75, 2, 41, 1, 52.25, 2)
Sending only changed bucket information
QD_i(53.5)
i
QPSD_i(5, 53, 5.25, 2, 12, 2, 35, 3, 52, 2)
QPD_i(53)
a
b
h
g DD_g(51.5)
DPSD_g(51)
DSD_f(41)
QPSD_h(5, 31, 5.25, 2,12, 2, 31, 1)
f DD_f(41)
DPSD_f(39)
d
e
Q-node
D-node
TF: 10 (−10 ~ +10)
TSC: 10
QD: currently sensed data of Q-node
QPD: previously sensed data of Q-node
QSD: send data of Q-node
QPSD: previously sent data of Q-node
DD: currently sensed data of D-node
DSD: send data of D-node
DPSD: previously sent data of D-node

Figure 16: Example (2) of data transmission filtering.

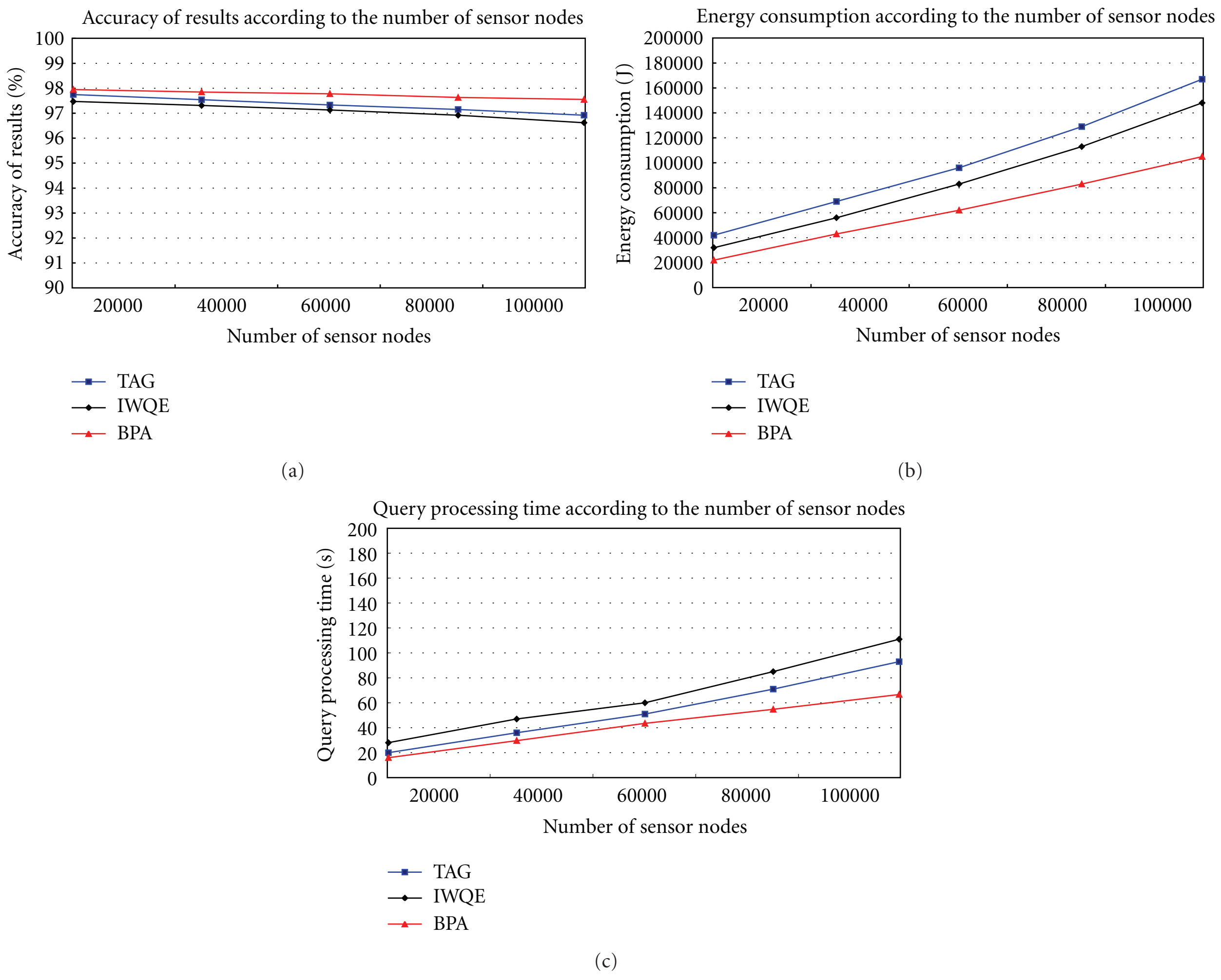


Figure 17: AVG aggregation according to the number of sensor nodes.

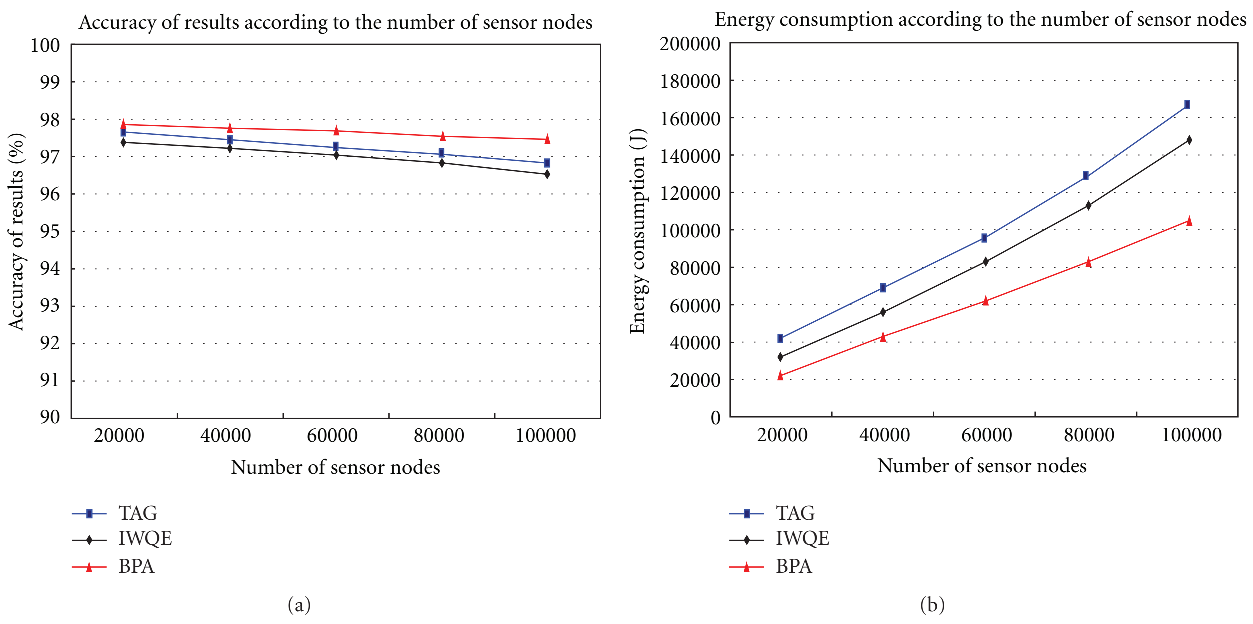


FIGURE 18: MEDIAN aggregation according to the number of sensor nodes.

the data solves the data redundancy problem by removing data with a redundant ID.

3.2. Data Structure. BPA uses bucket-based data structure and the variable bit compression coding technique in order to reduce energy consumption by sensor nodes in processing aggregate queries such as MEDIAN and HISTOGRAM.

The data store structure of D-node consists of ID, which is the ID of the sensor node, and Value, which is sensed data. In addition, the data store structure of Q-node contains ID indicating the ID of the sensor node, MinValue and MaxValue indicating the minimum and maximum values of collected data, A Value$_i$ indicating the average value of data in the bucket, and Count$_i$ indicating the number of data in the bucket. Figure 12 shows the bucket data structure of Q-node.

As in Figure 12, the whole bucket list has size BLSize (BLMax − BLMin), and the initial size of each bucket is BMax, which is the maximum size of a bucket. In addition, the bucket list stores information on individual buckets such as A Value$_i$, which is the mean value of data in bucket i, and Count$_i$, which is the number of data in bucket i. A Value$_i$ is obtained by

$$A\,\mathrm{Value}_i = \frac{\sum_{k=1}^{\mathrm{Count}_i} \mathrm{Value}_i^k}{\mathrm{Count}_i}. \quad (1)$$

In (1), A Value$_i$ is the mean of data included in bucket i, Count$_i$ the number of data in bucket i, and the Value$_i^k$ data in bucket i.

BPA uses the variable bit compression coding technique [13] for data compression. That is, BPA compresses Value in D-nodes and MinValue, MaxValue and A Value in Q-nodes, which occupy the largest part of data in D-nodes and Q-nodes, and stores the compressed data in order to reduce the size of data in sensor node data transmission.

Moreover, BPA divides and merges bucket data structure adaptively according to the number of data in the bucket in order to enhance the accuracy of query processing results.

Figures 13 and 14 show an example of bucket division and bucket merger when bucket list size BLSize is 60 (MinValue = 0, MaxValue = 60), maximum bucket size BMax is 10, minimum bucket size BMin is 2, and maximum number of data in the bucket BMaxC is 5.

As in Figure 13, when the number of data in bucket 4.1 exceeds the maximum number of data in the bucket (BMaxC = 5), it is divided into buckets 4.1.1 and 4.1.2, but because minimum bucket size BMin is 2, buckets 4.1.1 and 4.1.2 are not divided any longer.

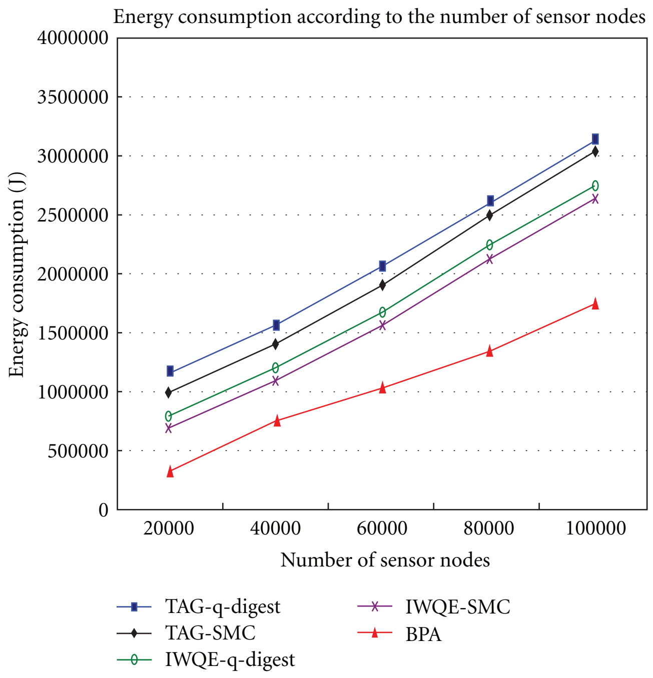
As in Figure 14, because the sum of the data counts of two buckets 4.1.1 and 4.1.2 is less than the maximum number of data in the bucket (BMaxC = 5) and maximum bucket size BMax is 10, the two buckets merge into bucket 4.1.

In order to reduce the energy consumption of sensor nodes, BPA sets a filtering range for each sensor node and sends data only when the data are outside the filtering range.

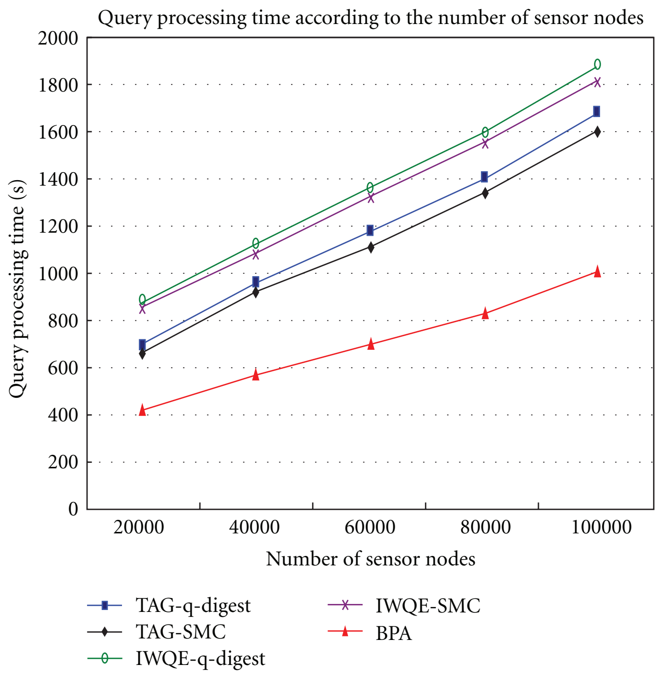
D-nodes are allocated filtering range DF. As in (2), DF of a D-node is calculated using TF (total filtering range) and TSC (the number of sensor nodes in the query region). The D-node sends data only when the data are outside DF:

$$\mathrm{DF} = \frac{\mathrm{TF}}{\mathrm{TSC}}. \quad (2)$$

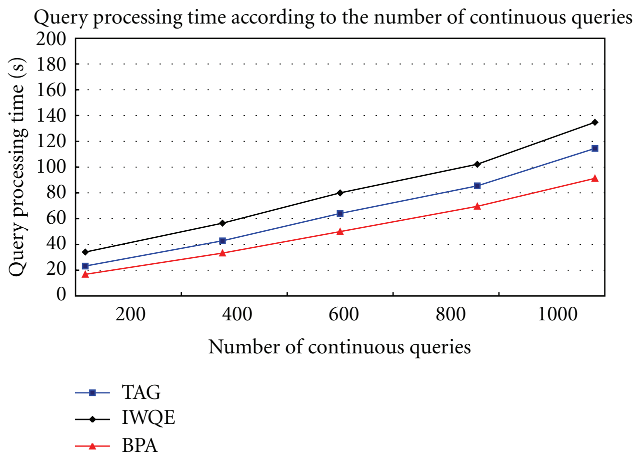
Q-nodes are allocated initial filtering range IQF and reset filtering range QF in aggregate query processing. In addition, Q-nodes send data only when the sum of bucket lists is outside filtering range QF.



(a)



(b)



(c)

Figure 19: AVG aggregation according to the number of continuous Queries.

As in (3), IQF of a Q-node is calculated using TF (total filtering range) and TSC (the number of sensor nodes in the query region)

$$\mathrm{IQF} = \frac{\mathrm{TF}}{\mathrm{TSC}}. \tag{3}$$

In aggregate query processing, a Q-node resets QF using IQF of the Q-node, DF of D-nodes, DSC (the number of D-nodes that have sent data to the Q-node within its communication range), and QF′ of the previous Q-node. If there is a previous Q-node and it has sent data, the Q-node resets QF with the sum of IQF of the Q-node, DFs of D-nodes that have sent data, and QF′ of the previous Q-node. This can be expressed as

$$\mathrm{QF} = \mathrm{IQF} + (\mathrm{DF} \times \mathrm{DSC}) + \mathrm{QF}'. \tag{4}$$

In case there is a previous Q-node but the previous Q-node has not sent data or in case there is no previous Q-node, QF of the Q-node is reset with the sum of IQF of the Q-node and DFs of D-nodes that have sent data. This can be expressed as in (5)

$$\mathrm{QF} = \mathrm{IQF} + (\mathrm{DF} \times \mathrm{DSC}). \tag{5}$$

Sensor nodes basically know the content of the query that S-node (the starting node) sent, effective time of the query (from the reception of the query to the transmission of the first sensed data), and the cycle of the query (interval to transmit the sensed data). Therefore, if data were not transmitted within the cycle of the query even when there existed Q-node or when there is no previous Q-node, it would deal with the query by setting QF according to (5).

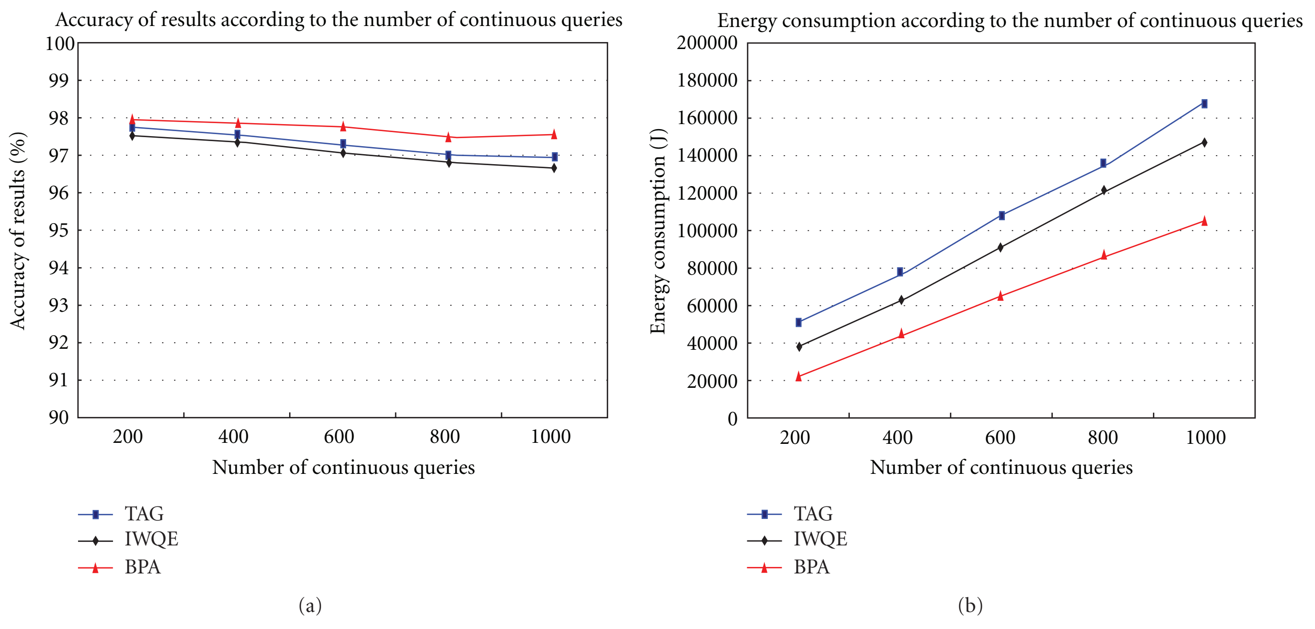


FIGURE 20: MEDIAN aggregation according to the number of continuous queries.

Figures 15 and 16 show an example of data transmission filtering by sensor nodes.

As in Figure 15, because DF of D-nodes is 1, D-node b, d, and e do not send data. However, D-node a sends $DSD_a(15)$, to Q-node h because its sensing data are outside filtering range DF_a. In addition, because Q-node h does not have a previous Q-node, QF_h is $(1 + (1 \times 1) = 2)$ by (5). In Q-node h, the sum of bucket list $QPSD_h$ is 65.5 and the sum of bucket list QSD_h is 68.5, which are outside filtering range QF_h, so Q-node h sends $QSD_h(5,21,5.25,2,13.5,2,31,1)$ to Q-node i.

As in Figure 16, because DF of D-nodes is 1, D-node g does not send data, but D-node c and f send DSD_c (36.5) and $DSD_f(37)$, respectively, to Q-node i because their sensing data are outside filtering ranges DF_c and DF_f, respectively. In addition, because if there is a previous Q-node the previous Q-node sends data, QF_i of Q-node i is $(1 + (1 \times 2) + 2 = 5)$ by (4). In Q-node i, the sum of bucket list $QPSD_i$ is 243.5 and the sum of bucket list QSD_i is 245.9, which are within filtering range QF_i, so Q-node i does not send data to Q-node j.

3.3. Algorithm. The BPA algorithm consists of processes for generating itinerary routing, sending an aggregate query through the generated itinerary routing, and returning the results of the aggregate query. Algorithm 1 shows the whole of the BPA algorithm.

As in Algorithm 1, the BPA algorithm first selects R-node, the closest sensor node to the center of the query region, builds hierarchical routing based on R-node, and collects sensor node information. Then, it forms a quad-tree using the collected sensor node information, and creates itinerary routing by selecting representative sensor node C-node from each cell of the quad-tree.

In addition, the BPA algorithm processes an aggregate query through the itinerary routing, starting from each C-node. That is, a Q-node sends an aggregate query through itinerary routing, and D-nodes return the results of the aggregate query. At that time, the Q-node and D-nodes determine whether data have been filtered or not, and if not filtered, they compress only changed bucket information and send it to the next node. In aggregate query processing, missing nodes, which are neither a Q-node nor a D-node, send query results to the closest node to S-node, the sensor node that started the query.

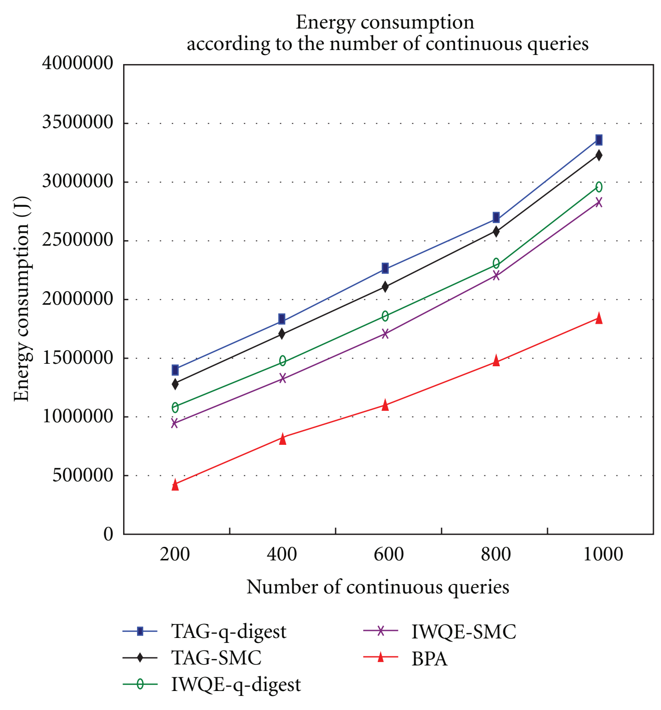
4. Performance Evaluation

4.1. Performance Evaluation Environment. The hardware specifications of the system used in performance evaluation were Intel Core 2.4 GHz CPU, 2 GB RAM, and 300 GB HDD, and its operating system was Windows XP. In addition, MFC (Microsoft Foundation Class Library) was used in simulation, and 13 parameters as in Table 1 were set in performance evaluation.

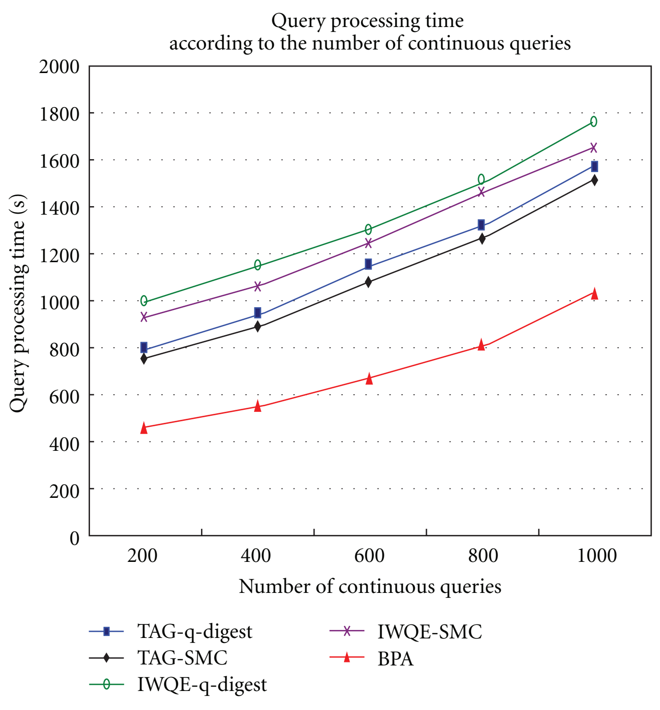
Particularly for BPA, TAG-q-digest, TAG-SMC, IWQE-q-digest, and IWQE-SMC, we optimized the data structure of BPA, q-digest, and SMC in evaluating the performance of MEDIAN query processing in order to maintain the accuracy of query processing results over 95%.

4.2. Results of Performance Evaluation. Figure 17 presents the results of performance evaluation in AVG query processing for BPA, TAG, and IWQE, showing the accuracy of results, energy consumption, and query processing time according to the number of sensor nodes.

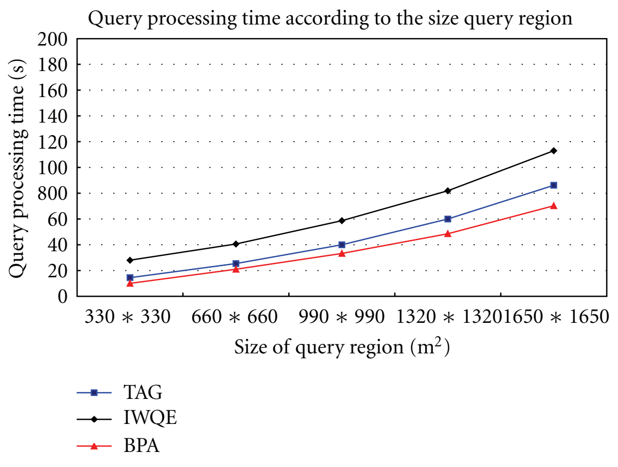
As in Figure 17, BPA showed 29% higher performance than IWQE and 18% higher than TAG in terms of the accuracy of query processing results. In terms of energy consumption, BPA showed 59% higher performance than TAG, and 37% higher than IWQE. Also in terms of query processing time, BPA showed 57% higher performance than IWQE and 28% higher than TAG.



(a)



(b)



(c)

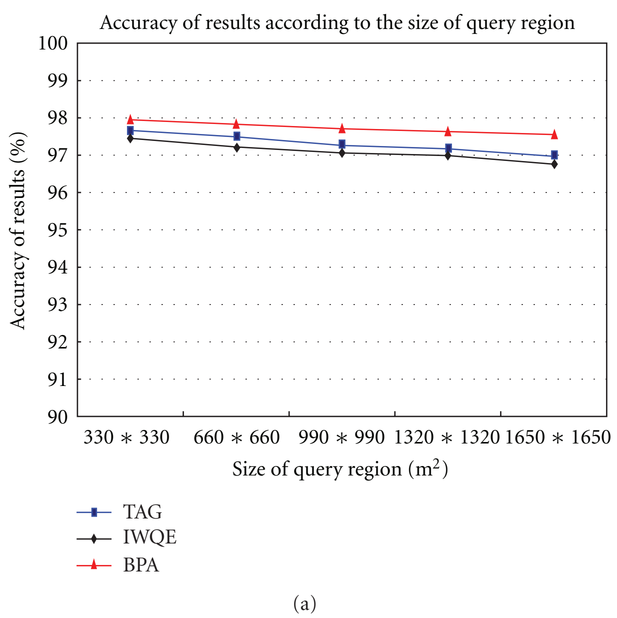
Figure 21: AVG aggregation according to the size of query region.

Figure 18 presents the results of performance evaluation in MEDIAN query processing for BPA, TAG-q-digest, TAG-SMC, IWQE-q-digest, and IWQE-SMC, showing energy consumption and query processing time according to the number of sensor nodes.

As in Figure 18, BPA showed 101% higher performance than TAG-q-digest, 88% higher than TAG-SMC, 66% higher than IWQE-q-digest, and 55% higher than IWQE-SMC in terms of energy consumption. In terms of query processing time as well, BPA showed 93% higher performance than IWQE-q-digest, 87% higher than IWQE-SMC, 66% higher than TAG-q-digest, and 59% higher than TAG-SMC.

Figure 19 presents the results of performance evaluation in AVG query processing for BPA, TAG, and IWQE, showing the accuracy of results, energy consumption, and query processing time according to the number of continuous queries.

As in Figure 19, BPA showed 28% higher performance than IWQE and 18% higher than TAG in terms of the accuracy of query processing results. In terms of energy consumption, BPA showed 66% higher performance than TAG, and 42% higher than IWQE. Also in terms of query processing time, BPA showed 56% higher performance than IWQE and 26% higher than TAG.



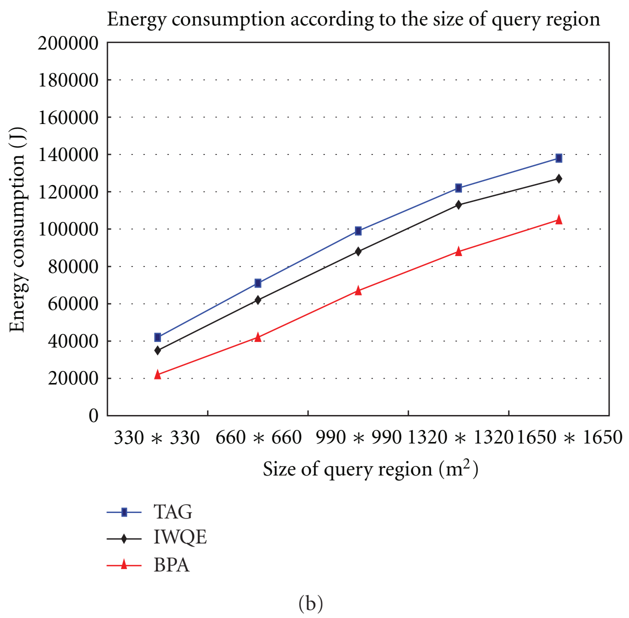


FIGURE 22: MEDIAN aggregation according to the size of query region.

```
Function  BPA_Process(QUERY q, SENSOR_NODE nodelist[ ])
Input     q: Query Information, nodelist[ ]: Information of Sensor Nodes
Output    BUCKET_LIST bkl: Information of bucket List
(1)       SENSOR_NODE rnode = RNode_Search(q, nodelist)
(2)       HROUTE hr = HRoute_Build(rnode, nodelist)
(3)       HR_SENSOR_NODE_INFO sni = SensorInfo_Collection(hr,MBR())
(4)       QT_NODE qn = QT_Build(sni, nodelist)
(5)       IROUTE ir = QT_CNode_Itinerary(qn, nodelist)
(6)       if(QNodeIs() || DNodeIs())
(7)       {
(8)           QNode_Query_Send(q, ir)
(9)               If(!QNode_Filtering())
(10)              {
(11)                  Bucket_Build(Bucket_Update())
(12)                  Compress_Value()
(13)                  BUCKET_LIST bkl = QNode_Duplicate_Result_Send(ir)
(14)                  return bkl;
(15)              }
(16)               If(!DNode_Filtering())
(17)              {
(18)                  Bucket_Build(Bucket_Update())
(19)                  Compress_Value()
(20)                  BUCKET_LIST bkl = DNode_Duplicate_Result_Send(ir)
(21)                  return bkl;
(22)              }
(23)      }
(24)      else
(25)      {
(26)            MissingNode_Send_Result()
(27)      }
```

ALGORITHM 1: Algorithm of BPA.

Table 1: Parameters for performance evaluation.

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

Figure 20 presents the results of performance evaluation in MEDIAN query processing for BPA, TAG-q-digest, TAG-SMC, IWQE-q-digest, and IWQE-SMC, showing energy consumption and query processing time according to the number of continuous queries.

As in Figure 20, BPA showed 104% higher performance than TAG-q-digest, 93% higher than TAG-SMC, 71% higher than IWQE-q-digest, and 59% higher than IWQE-SMC in terms of energy consumption. In terms of query processing time as well, BPA showed 90% higher performance than IWQE-q-digest, 80% higher than IWQE-SMC, 64% higher than TAG-q-digest, and 56% higher than TAG-SMC.

Figure 21 presents the results of performance evaluation in AVG query processing for BPA, TAG, and IWQE, showing the accuracy of results, energy consumption, and query processing time according to the size of query region.

As in Figure 21, BPA showed 28% higher performance than IWQE and 18% higher than TAG in terms of the accuracy of query processing results. In terms of energy consumption, BPA showed 45% higher performance than TAG, and 31% higher than IWQE. Also in terms of query processing time, BPA showed 75% higher performance than IWQE and 23% higher than TAG.

Figure 22 presents the results of performance evaluation in MEDIAN query processing for BPA, TAG-q-digest, TAG-SMC, IWQE-q-digest, and IWQE-SMC, showing energy consumption and query processing time according to the size of query region.

As in Figure 22, BPA showed 94% higher performance than TAG-q-digest, 84% higher than TAG-SMC, 68% higher than IWQE-q-digest, and 56% higher than IWQE-SMC in terms of energy consumption. In terms of query processing time as well, BPA showed 100% higher performance than IWQE-q-digest, 90% higher than IWQE-SMC, 66% higher than TAG-q-digest, and 54% higher than TAG-SMC.

4.3. Analysis of Performance Evaluation. When performance in AVG query processing was evaluated according to the number of sensor nodes, the number of continuous queries, and the size of query region, BPA showed higher performance than TAG, and IWQE in terms of the accuracy of processing results, energy consumption, and query processing time. This is probably because BPA solves the problem in TAG that data of sensor nodes not included in the query region are transmitted, resolves the shortcoming of IWQE by reducing the number of missing nodes happening in the routing process, and processes an aggregate query in parallel by dividing the query region.

Moreover, when performance in MEDIAN query processing was evaluated according to the number of sensor nodes, the number of continuous queries, and the size of query region, BPA showed higher performance than TAG-q-digest, TAG-SMC, IWQE-q-digest, and IWQE-SMC in terms of energy consumption and query processing time. This is probably because BPA does not use a data structure with a fixed range as in q-digest, and SMC but updates the data structure adaptively, sends only changed bucket information instead of sending all aggregate data each time, and performs compressing and filtering for data to be transmitted.

Particularly in AVG and MEDIAN query processing, BPA showed even higher performance than the existing techniques when the number of sensor nodes and the number of continuous queries were large and when the size of query region was large.

5. Conclusions

This study proposed BPA, a bucket-based parallel aggregate query processing technique for more efficient aggregate query processing in wireless sensor networks. In order to reduce the energy consumption of sensor nodes and query processing time, BPA builds a query region into a quad-tree and processes an aggregate query in parallel through the itinerary routing over the cell coverage of quad-tree nodes. In addition, it minimizes the occurrence of missing nodes for higher accuracy of query processing results and reduces data loss from transmission errors through the double data transmission by sensor nodes.

BPA also uses bucket-based data structure and the variable bit compression coding technique in order to reduce energy consumption by sensor nodes in processing aggregate queries MEDIAN and HISTOGRAM. Particularly for higher accuracy of query processing results, it divides and merges the bucket data structure adaptively according to the number of data in the bucket. What is more, data are transmitted only when they are outside the filtering range, and this reduces the energy consumption of sensor nodes. Lastly, we proved the superiority of BPA proposed as an aggregate query processing technique through various experiments using sensor data.

Acknowledgments

This work was supported by 00046658 Business for Cooperative R&D between Industry, Academy, and Research Institute funded by Korea Small and Medium Business Administration in 2011.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–105, 2002.

[2] D. Culler, D. Estrin, and M. Srivastava, "Overview of sensor networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[3] S. Motegi, K. Yoshihara, and H. Horiuchi, "DAG based in-network aggregation for sensor network monitoring," in *Proceedings of the 2006 International Symposium on Applications and the Internet (SAINT '06)*, pp. 292–299, January 2006.

[4] H. Cheng, L. Qin, and J. Xiaohua, "Heuristic algorithms for real-time data aggregation in wireless sensor networks," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC '06)*, pp. 1123–1128, July 2006.

[5] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[6] C. C. Hung and W. C. Peng, "Optimizing in-network aggregate queries in wireless sensor networks for energy saving," *Data and Knowledge Engineering*, vol. 70, no. 7, pp. 617–641, 2011.

[7] D. Wang, J. Xu, F. Wang, and J. Liu, "Mobile filter: exploring filter migration for error-bounded continuous sensor data collection," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 8, pp. 4093–4104, 2010.

[8] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for Ad-hoc sensor networks," in *Proceedings of the Symposium on Operating System Design and Implementation*, pp. 131–146, 2002.

[9] S. Roy, M. Conti, S. Setia, and S. Jajodia, "Secure median computation in wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1448–1462, 2009.

[10] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*, pp. 239–249, November 2004.

[11] Y. Xu, W. Lee, J. Xu, and G. Mitchell, "Processing window queries in wireless sensor networks," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 270–280, 2006.

[12] D. Pendarakis, N. Shrivastava, L. Zhen, and R. Ambrosio, "Information aggregation and optimized actuation in sensor networks: enabling smart electrical grids," in *Proceedings of the 26th IEEE International Conference on Computer Communications (IEEE INFOCOM '07)*, pp. 2386–2390, May 2007.

[13] J. J. Kim, H. K. Kang, D. S. Hong, and K. J. Han, "An efficient compression technique for a multi-dimensional index in main memory," in *Proceedings of the International Conference on Visual Information Systems (VISUAL '07)*, pp. 336–346, 2007.

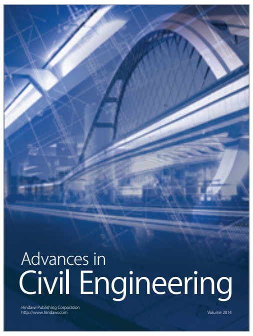
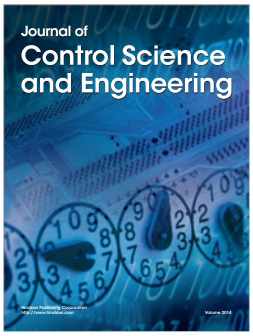
[14] J. Considine, F. Li, G. Kollios, and J. Byers, "Approximate aggregation techniques for sensor databases," in *Proceedings of the 20th International Conference on Data Engineering (ICDE '04)*, pp. 449–460, April 2004.

Journal of
Engineering

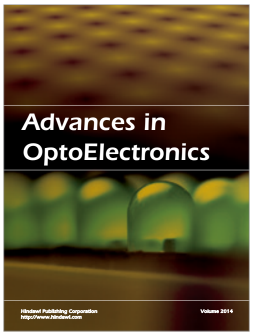
The Scientific World Journal

International Journal of
Rotating Machinery

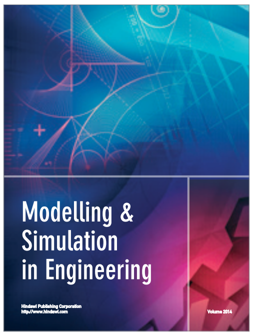
Journal of
Sensors

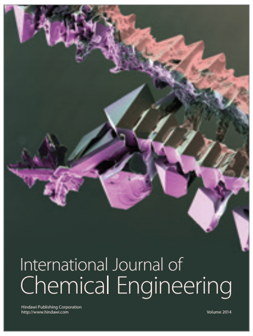
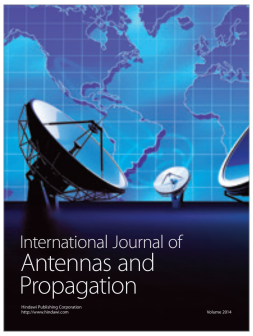
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

Journal of
Control Science and Engineering

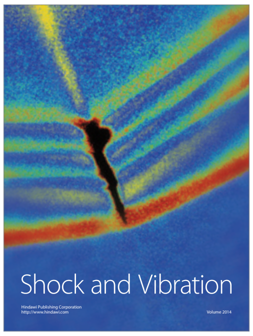
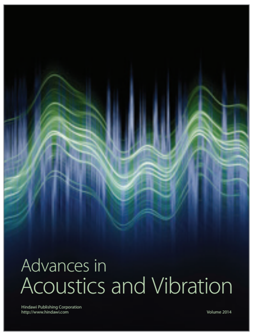
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive
Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration