IEICE TRANS. INF. & SYST., VOL.E98–D, NO.4 APRIL 2015

LETTER

Efficient In-Network Processing of Continuous Grouped Aggregation Queries in Sensor Networks

Inchul SONG[†a)], ***Member***, **Yohan J. ROH**[†b)], ***and*** **Myoung Ho KIM**[††c)], ***Nonmembers***

SUMMARY In this letter, we propose an energy-efficient in-network processing method for continuous grouped aggregation queries in wireless sensor networks. As in previous work, in our method sensor nodes partially compute aggregates as data flow through them to reduce data transferred. Different from other methods, our method considers group information of partial aggregates when sensor nodes forward them to next-hop nodes in order to maximize data reduction by same-group partial aggregation. Through experimental evaluation, we show that our method outperforms the existing methods in terms of energy efficiency.

key words: *grouped aggregation query, sensor network, multipath routing*

1. Introduction

In this letter, we propose an energy-efficient way of answering continuous grouped aggregation queries in wireless sensor networks (WSNs). A continuous grouped aggregation query (aggregation query in short) divides tuples of sensor readings into disjoint groups and reports an aggregate for each group in a predefined interval. For example, the user may pose the following query to monitor the occupancy of the rooms in a building: "Report the average loudness of each room on the sixth floor of a building every 60 seconds."

In-network processing of aggregation queries is a widely accepted technique to reduce energy consumption for wireless communication in WSNs [1]. The main idea is that aggregates are partially computed in the network to reduce data transferred as data flow through the sensor nodes. In previous work [1]–[3], in-network processing is typically conducted on a routing tree rooted at the base station. When receiving tuples or partial aggregates from its child nodes, an intermediate sensor node partially aggregates them into smaller parital aggregates and forwards them to its parent node. For grouped aggregation queries, only those tuples or partial aggregates belonging to the same group can be partially aggregated. The more the same-group partial aggregation occurs, the less data is transferred.

In this letter, we propose an energy-efficient in-network processing method called Group-aware Multipath Routing (GMR) for continuous grouped aggregation queries in WSNs. In all of existing methods, sensor nodes forward tuples or partial aggregates to pre-determined parent nodes without considering their group information. On the other hand, our method is built upon a novel group distance measure with which sensor nodes determine where tuples of a certain group may be generated. Based on this information, sensor nodes forward tuples or partial aggregates in different groups to different next-hop nodes such that the data reduction by same-group partial aggregation be maximized. Through experimental evaluation in a range of simulated sensor network environments, we show that our method outperforms the existing methods in terms of energy efficiency for query processing.

The rest of the letter is organized as follows. In Sect. 2, we define the problem of answering continuous grouped aggregation queries in WSNs. In Sect. 3, we describe our proposed method in detail. In Sect. 4, we compare the energy-efficiency of our method to those of existing approaches in various WSN environments through experimental evaluation. Finally, we conclude the letter in Sect. 5.

2. Preliminaries

A sensor network can be viewed as a distributed table, named `sensors(id, temp, light, loc, ...)`, which has the identifier of a sensor node and one attribute for each sensor. The user of a sensor network can query this `sensors` table by using an SQL-like query language. Continuous grouped aggregation queries for WSNs can be expressed as follows [1]:

```
SELECT {aggregates, selected-attributes}
  FROM sensors WHERE conditions-for-tuples
  GROUP BY {group-attributes}
  HAVING conditions-for-groups EVERY e.
```

The semantics of the above query is almost the same as the SQL aggregation query, except for the `EVERY` clause. A set of tuples, each of which is in the form of `<group_id,aggregate1,aggregate2,...>` per group, is produced with a timestamp for each epoch. The duration of each epoch is specified by the `EVERY` clause. We consider only standard SQL aggregation operators (i.e., `AVG`, `SUM`, `MIN`, `MAX`, and `COUNT`) in this letter.

Given a continuous grouped aggregation query, we want the results to be collected at the base station once for each epoch in such a way that data reduction by in-network processing be maximized.

Manuscript received November 21, 2014.
Manuscript publicized January 21, 2015.
[†]The authors are with Samsung Advanced Institute of Technology, Korea.
[††]The author is with KAIST, Korea.
a) E-mail: icsong@gmail.com
b) E-mail: yohan.roh@samsung.com
c) E-mail: mhkim@dbserver.kaist.ac.kr
DOI: 10.1587/transinf.2014EDL8237

3. Group-Aware Multipath Routing

We model a sensor network as a connected undirected graph $G = (V, E)$. There is one distinguished node, called the *root node* that is directly connected to the base station. There is an edge between two nodes if they can communicate with each other. The *distance* of v_i in graph G, denoted by $d_G(v_i)$, is the length (i.e., the number of edges) of any shortest path between v_i and the root node, v_0. The *stratified graph* $S = (V, E')$ of a graph $G = (V, E)$ is a subgraph of G, where an edge $\{v_i, v_j\} \in E$ is in E' if and only if $|d_G(v_i) - d_G(v_j)| = 1$. Figure 1 (a) shows a stratified graph with nine sensor nodes (v_0 is the root node). A node v is at level i if $d_G(v) = i$.

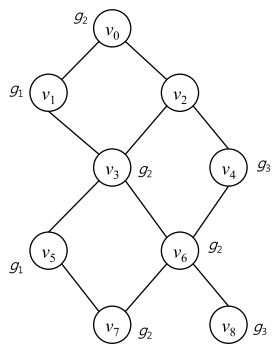
Definition 1. *Let $S = (V, E)$ be a stratified graph. We define a successor relation $\rightarrow$ on V as follows:*

$$\rightarrow = \{(v_i, v_j) | \{v_i, v_j\} \in E \text{ and } d_S(v_i) = d_S(v_j) + 1\}.$$

If a pair (v_i, v_j) is in $\rightarrow$, we use a notation $v_i \rightarrow v_j$. When $v_i \rightarrow v_j$, we say that v_j is a *successor* of v_i and v_i is a *predecessor* of v_j. If a node has no predecessor, it is called a *terminal node*. Otherwise, it is a *non-terminal node*. The transitive closure of $\rightarrow$ is denoted by $\rightarrow^+$. When $v_i \rightarrow^+ v_j$, we say that v_j is *reachable* from v_i. Note that for any non-root node v_i, $v_i \rightarrow^+ v_0$ holds, where v_0 is the root node. For example, in Fig. 1 (a), the successors of v_6 are v_3 and v_4, and the nodes that are reachable from v_6 are v_3, v_4, v_1, v_2, and v_0.

Given a query q, a node whose sensor readings satisfies the conditions in the `WHERE` clause of the query is called a *qualified node* (shortly, a *Q-node*) of q. For the query mentioned in Sect. 1, Q-nodes are the nodes on the sixth floor in a building. Each Q-node is assigned a group ID based on the group to which its sensor readings belong, which is determined by the `GROUP BY` clause of the query. The group ID of a Q-node v is denoted by $\mathrm{G}_q(v)$. For the aforementioned query, the group ID of a Q-node is the room number where the node is placed.

In the following, we define the *minimum aggregatable distance* (MD) that is assigned to every node for each group of a given query. An intermediate node v uses the MD values of its successor nodes to determine to which successor node to forward each partial aggregate during query processing.



(a) A stratified graph with nine sensor nodes for some grouped aggregation query q

	g_1	g_2	g_3
v_0	0	0	0
v_1	0	1	1
v_2	1	1	1
v_3	1	0	2
v_4	2	2	0
v_5	0	1	3
v_6	2	0	1
v_7	1	0	2
v_8	3	1	0

(b) MD values for the three groups

Fig. 1 Q-nodes for three groups and minimum aggregatable distances.

Definition 2. *Given a stratified graph $S = (V, E)$ and a query q, the* minimum aggregatable distance *(MD) of a node v_i for a group g, denoted by $\mathrm{MD}_q(v_i, g)$, is defined as follows:*

(1) If $\mathrm{G}_q(v_i) = g$, $\mathrm{MD}_q(v_i, g) = 0$.
(2) Otherwise, $\mathrm{MD}_q(v_i, g) = \min\{d_S(v_i, v_j) | v_i \rightarrow^+ v_j$, where $\mathrm{G}_q(v_j) = g$ or v_j is the root node$\}$.

In other words, $\mathrm{MD}_q(v_i, g)$ is the distance from node v_i to the closest reachable Q-node belonging to group g on the paths from v_i to the root node v_0. For example, in Fig. 1 (a), all nodes except node v_2 are Q-nodes for a given query q and a group ID is indicated next to each Q-node in the figure. Figure 1 (b) shows the MD values of nodes for groups g_1, g_2, and g_3. Note that when k number of groups are formed by a query, each node has k number of MD values.

Observation 1. *Let q be a given query. For the following two cases (1) and (2), the MD of node v for group g can be computed as follows: $\mathrm{MD}_q(v, g) = \min\{\mathrm{MD}_q(w, g) | v \rightarrow w\} + 1$.*

(1) v is a Q-node for query q, but $\mathrm{G}_q(v) \neq g$.
(2) v is not a Q-node for query q.

From Observation 1, we can see that when $\mathrm{MD}_q(v, g)$ is not zero, it can be calculated based on only the MDs of node v's successors.

3.1 Distributed MD Computation

GMR proceeds in two phases: the *setup* phase and the *query processing* phase. In the setup phase, which is executed before any query is issued, the stratified graph of a sensor network is constructed: that is, each node finds its distance from the root node (i.e., its *level*) and determines its successors. To this end, a *distance message* floods from the base station down to the network. This message contains an integer, called a *distance value*, which is initially zero in the root node and then incremented one by one as the message passes through sensor nodes. Every node v_j that sends a distance message to node v_i becomes a successor of v_i if the distance of v_j is less than the distance of v_i by one. Each node maintains its distance together with the identifiers of its successors.

The query processing phase, which begins when the user poses a query, consists of two steps: *query dissemination* and *result collection*. In the query dissemination step, a query message that contains the query and the MD values of the sender floods from the base station down to the network. Starting from the root node, the delivery of a query message together with the computation of MD values proceeds level by level in a stratified graph. MD values at each node are computed as follows:

- $\mathrm{MD}_q(v_0, g) = 0$ for group g such that $g = \mathrm{G}_q(v_0)$.
- Let v_i be a non-root node. For each group g_j, $j = 1, \cdots, k$:
  - if $\mathrm{G}_q(v_i) = g_j$, then $\mathrm{MD}_q(v_i, g_j) = 0$.
  - Otherwise, $\mathrm{MD}_q(v_i, g_j) = \min\left\{\mathrm{MD}_q(w, g_j) | v_i \to w\right\} + 1$.

Once the query dissemination and MD value computation are completed, the results of the query are collected every epoch, which we call the result collection step. GMR uses the slotted approach as in [4] for time synchronization among sensor nodes organized in the form of a stratified graph such that nodes are activated level-by-level from the highest level nodes towards lower level nodes. A node, in its time slot, first samples sensors, performs in-network processing with the data received from its predecessors and its own sensor readings, and produces several partial aggregates. Then it forwards these partial aggregates to its successors by the algorithm we describe next.

3.2 Multipath Routing with In-Network Processing

In this section we describe how a node forwards aggregates to its successors based on their MD values. The basic idea is that, for frequent and early partial aggregation, a node forwards each aggregate for group g to the successor node with the smallest MD for group g because that node is the closest to some node that generates tuples in group g. Suppose a given query is q. For node v, let W_g be a set of successors of v that have the smallest MD for group g, i.e., $W_g = \arg\min_u \left\{\mathrm{MD}_q(u, g) | v \to u\right\}$. Any node in W_g is called a *best successor* of node v for group g.

(1) Operations in a terminal node v: If v is not a Q-node for query q, do nothing and EXIT. Otherwise, let $\mathrm{G}_q(v) = g$. Create a message that contains a tuple of the form <`group_id,aggregate1,aggregate2`>. Here, `group_id` is g, and `aggregate1`, `aggregate2`, ... are simply the sensor readings of v. Send this message to any best successor node.

(2) Operations in a nonterminal node v:

- Collect all the messages from its predecessors.
- Perform in-network processing as much as possible to obtain partial aggregates. Suppose n partial aggregates ($n \geq 1$) $agg_1, agg_2, \ldots, agg_n$ are obtained. Let the group of agg_1 be g_1, the group of agg_2 be g_2, ..., the group of agg_n be g_n. For a given partial aggregate agg_i, let W_i be the set of best successors of node v for group g_i.
- Create messages as follows: Initially, create n messages with one message m_i for one partial aggregate agg_i. The number of these messages will be reduced through the following two-step merging process.
  - In the first merging step, we merge two messages if there is a node that is the best successor for both messages as follows: repeatedly merge two messages m_i and m_j into one message m_i if there is at least one node in both W_i and W_j. Update W_i such that $W_i = W_i \cap W_j$. If the merging of two messages causes a message overflow, do not merge them. Repeat until no merge occurs.
  - In the second merging step, we further merge messages by using a heuristic similar to the "first fit" strategy as follows: repeatedly merge any two messages m_i and m_j into one message m_i if this merging does not cause a message overflow. Update W_i such that $W_i = W_i \cup W_j$. Repeat until no merge occurs. Let k messages $m_1, \ldots, m_k$ remain.
- Send k messages to its successors as follows: For $i = 1, \ldots, k$, send each message m_i to a node w in W_i that is selected as follows: Select the node w in W_i that is the best successor for the maximum number of partial aggregates in message m_i.

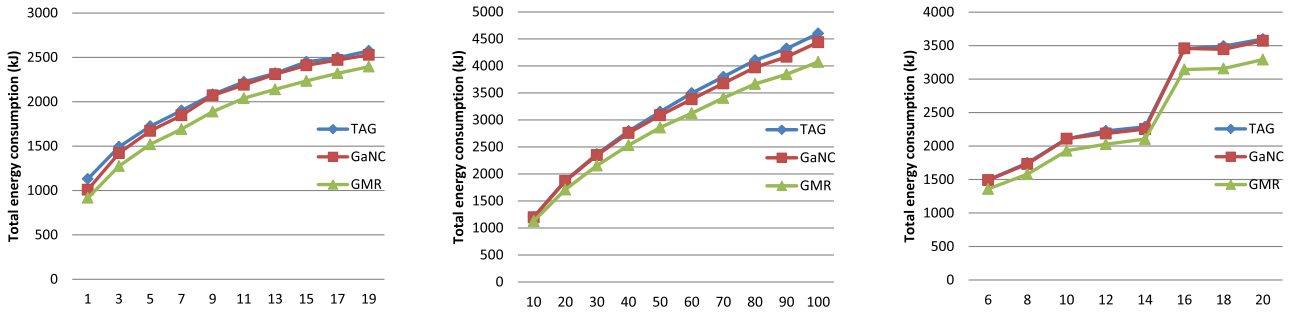
Note that the merging of two messages in the first merging step always allows every partial aggregate in both messages to be sent towards a closest reachable Q-node in the same group while the merging in the second merging step may not.

4. Evaluation

We conduct various experiments to compare our method with two previous representative routing-tree based approaches for in-network processing of continuous grouped aggregation queries in WSNs: TAG [1] and GaNC [3]. Both of them use a routing tree for in-network processing, as described in Sect. 1. What GaNC differs from TAG is that it forms a routing tree such that sensor nodes that produce tuples belonging to the same group are located close to each other. However, in both of the methods, each node blindly forwards tuples or partial aggregates to its pre-determined parent node, regardless of their groups.

As performance metric, we use the total amount of energy consumption for wireless communication in collecting the results of a grouped aggregation query in one epoch. We model per-message energy consumption by the following model used in [5]: $energy = m \times message\_size + b$, where m and b are device-specific constants, and $message\_size$ denotes the size of message in bytes. As in [5], when sending a message, m and b are set to 0.0144 mJ and 0.4608 mJ, respectively; when receiving a message, m and b are set to 0.00576 mJ and 0.1152 mJ, respectively.

In our experiments, sensor nodes are deployed randomly in a rectangular area whose size is 600m × 600m. The base station is placed at the center of the network. The communication range of each sensor node is set to 30m. A grouped aggregation query divides sensor readings into disjoint groups. There are 10 groups by default. The selectivity, whose default value is 25%, specifies what percentage of nodes are Q-nodes. Each Q-node belongs to a certain group with equal probability. The node density, which is set to 10 in all experiments, denotes the average number of neighbor nodes. The default size of a partial aggregate is 10 bytes



(a) Varying the number of groups (b) Varying the selectivity (c) Varying the tuple size

Fig. 2 Experimental results.

Table 1 Various parameters used in the experiments.

Parameter	Default value	Range
Network size (m^2)	600×600	—
Communication range (m)	30	—
Number of groups	10	1–19
Selectivity (%)	25	1–100
Node density	10	—
Size of partial aggregate (byte)	10	6–20

and a single message can contain up to 29 bytes. Table 1 summarizes the default values and ranges of the parameters used in the evaluation. We assume that wireless communication is lossless. All the values in the figures are obtained by computing the average of ten executions of a query over randomly generated sensor networks.

In Fig. 2 (a), we vary the number of groups formed by a query from 1 (aggregation query without grouping) to 19 (close to a non-aggregation query where most of the Q-nodes are in different groups). As shown in the figure, GMR outperforms the other methods in all cases because of its group-aware forwarding. Figure 2 (b) shows the results when we vary the selectivity from 1 (only few nodes are Q-nodes) to 100 (the entire nodes are Q-nodes). When the selectivity is low, the chances of in-network processing are low in all the methods. However, as the selectivity increases, GMR reduces more data by increased same-group partial aggregation through multipath routing. Lastly we vary the tuple size from 6 (partial aggregates of small sizes—few messages are used) to 20 (partial aggregates of large sizes—every aggregate is sent in a separate message) in Fig. 2 (c). The performance difference is noticeable when the tuple size is large. This is because every partial aggregate is sent in a separate message, and thus can be sent towards a closest reachable Q-node in the same group.

5. Conclusions

We proposed an energy-efficient group-aware query processing method for continuous grouped aggregation queries in WSNs. The key idea is that an intermediate node forwards tuples or partial aggregates such that the chances of same-group partial aggregation be maximized. We showed through experimental evaluation that our method outperformed the existing tree-based methods in various sensor network environments.

References

[1] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," SIGOPS Oper. Syst. Rev., vol.36, no.SI, pp.131–146, 2002.
[2] Y. Yao and J. Gehrke, "Query processing in sensor networks," CIDR, pp.233–244, 2003.
[3] A. Sharaf, J. Beaver, A. Labrinidis, and K. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks," The VLDB Journal, vol.13, no.4, pp.384–403, 2004.
[4] S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong, "TinyDB: An acquisitional query processing system for sensor networks," ACM Trans. Database Syst., vol.30, no.1, pp.122–173, 2005.
[5] B. Chen, W. Liang, and J. Yu, "Energy-efficient skyline query optimization in wireless sensor networks," Wireless Netw., vol.18, no.8, pp.985–1004, 2012.