Sensors **2015**, *15*, 15033-15066; doi:10.3390/s150715033

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

A Novel Two-Tier Cooperative Caching Mechanism for the Optimization of Multi-Attribute Periodic Queries in Wireless Sensor Networks

ZhangBing Zhou [1,4,*], Deng Zhao [1], Lei Shu [2] and Kim-Fung Tsang [3]

[1] School of Information Engineering, China University of Geosciences (Beijing), Beijing 100083, China; E-Mail: zhaodpx@163.com
[2] Guangdong Provincial Key Lab. of Petrochemical Equipment Fault Diagnosis, Guangdong University of Petrochemical Technology, Maoming 525000, China; E-Mail: lei.shu@ieee.org
[3] Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China; E-Mail: ee330015@cityu.edu.hk
[4] Computer Science Department, TELECOM SudParis, Evry 91011, France

* Author to whom correspondence should be addressed; E-Mail: zhangbing.zhou@gmail.com; Tel.: +86-10-8232-3184.

Academic Editor: Leonhard M. Reindl

Received: 10 May 2015 / Accepted: 17 June 2015 / Published: 26 June 2015

Abstract: Wireless sensor networks, serving as an important interface between physical environments and computational systems, have been used extensively for supporting domain applications, where multiple-attribute sensory data are queried from the network continuously and periodically. Usually, certain sensory data may not vary significantly within a certain time duration for certain applications. In this setting, sensory data gathered at a certain time slot can be used for answering concurrent queries and may be reused for answering the forthcoming queries when the variation of these data is within a certain threshold. To address this challenge, a popularity-based cooperative caching mechanism is proposed in this article, where the popularity of sensory data is calculated according to the queries issued in recent time slots. This popularity reflects the possibility that sensory data are interested in the forthcoming queries. Generally, sensory data with the highest popularity are cached at the sink node, while sensory data that may not be interested in the forthcoming queries are cached in the head nodes of divided grid cells. Leveraging these cooperatively cached sensory data, queries are answered through composing these two-tier cached data.

Experimental evaluation shows that this approach can reduce the network communication cost significantly and increase the network capability.

Keywords: periodic query optimization; cooperative caching; wireless sensor networks

1. Introduction

With the rapid development of microelectronic, wireless communication, new and renewable energy technologies, smart sensor nodes become smaller in physical size, stronger in storage and computational capabilities, more powerful in battery capacity and less expensive in price. Sensor nodes form wireless sensor networks (WSNs), which have been adopted in widespread domain applications, including ambient assisted living [1], target tracking [2], bridge or traffic monitoring [3], *etc.* Sensor nodes are mostly battery-powered, which are difficult to be recharged and replaced, especially when deployed in harsh environments. Although energy harvesting from natural sources [4], energy replenishment [5,6] and radio optimization and charging [7,8] technologies have been developed to recharge the battery of sensor nodes, network lifetime maximization and prolongation is still essential, and energy efficiency is one of the most important research challenges in WSNs nowadays [7]. Note that the network lifetime can be defined as various semantics depending on the context of application domains, and a definition with wide acceptance is the time when the first sensor node depletes its energy [9]. Therefore, techniques that facilitate sensory data gathering efficiently for answering queries, while prolonging the network lifetime as much as possible, are fundamental.

As presented by Xu *et al.* [10], queries in WSNs are typically conducted in a periodic, rather than one-shot, fashion for supporting one or multiple applications. Note that queries in WSNs are different from those in query-based WSNs [11], where sensor nodes are producers (sources) and consumers (sinks) of resources simultaneously. In this article, sensor nodes gather sensory data, which are aggregated and routed to the sink node according to the requirement of certain applications. Usually, WSNs can be shared by multiple applications to improve the network utilization efficiency [12]. Consequently, multiple queries are performed in a certain time period, and multiple-attribute sensory data are often interested [13,14]. These queries may have overlapping sub-regions of interest. Besides, the points of interest for certain applications may be within a certain sub-region for a certain time duration, while evolving moderately to a neighboring sub-region [15,16]. In this setting, when the number of queries is relatively large and each query is to be processed independently, the capability required for processing these queries may be above the capability that the network can provide, and consequently, the delay for query answering may be towards infinity [10,17]. Therefore, the mechanism for the optimization of multi-attribute query processing, while prolonging the network lifetime, is an important research challenge.

Traditional techniques have studied the query processing in WSNs from various aspects, including in-network query processing [18], aggregated query processing [19], compressed data aggregation [20], spatial correlation data aggregation [21], range query processing [22], opportunistic sampling-based query processing [23], snapshot and continuous data aggregation [17,24], real-time

query processing [25], multiple dimensional or attributes query optimization [26–28], cooperative caching-based query processing [29,30], *etc*. Generally, these techniques are mostly exploring the one-shot query scheduling, where one single attribute is interested, whereas few efforts study periodic, aggregated and multi-attribute query processing [31]. Note that queries to be conducted in a certain time duration usually have overlapping sub-regions, where sensory data in these sub-regions gathered from the network can be shared for answering these concurrent queries. Besides, sensory data gathered at the current time slot may be reused for answering the forthcoming queries, when the variation of these sensory data is within an allowed threshold. In fact, sensory data may not vary dramatically in certain applications (like health or environmental monitoring), and many applications may work well when the bias of sensory data ((i) being used and (ii) being sensed in real time) is within a certain threshold [32]. Without loss of generality, a time duration is divided and represented by discrete time slots. Queries issued at a certain time slot are rewritten into one query. Hence, these concurrent queries are processed in batches. Besides, certain sensory data are cached in the network for answering the forthcoming queries and are retrieved from the network when the bias between the cached value and the current sensing value is above a certain threshold. Generally, this threshold is pre-specified as an appropriate value considering the characteristics of certain attributes and the requirements of certain applications. This strategy should reduce the network communication cost, improve the network capability, shorten the response time of query answering and, most importantly, prolong the network lifetime to some extent. Consequently, caching and refreshing multi-attribute sensory data in the network, while diminishing the cost of answering the forthcoming queries, is an important research problem to be explored further.

To remedy this issue, a two-tier popularity-based cooperative caching (PCC) mechanism is developed to support the periodic query processing, where multi-attribute sensory data are cached in the sink node and leaf head nodes of an index tree. Our main contributions are presented as follows:

- Given a network represented as square grid cells with inverted files, an index tree is constructed, where grid cells correspond to the leaf nodes in this tree. A query expects to return sensory data for certain attributes in a set of neighboring grid cells. For simplicity, sensory data in a grid cell with an attribute is considered as an atomic unit for query answering and caching manipulation. A query is answered through composing sensory data that are: (i) cached in the sink node; and (ii) gathered from the network in real time.
- The sink node is usually limited in storage and computational capabilities and can hardly cache sensory data of all sensor nodes in the whole network. Besides, a sub-region, rather than the whole network, is usually interested in applications within a certain time duration. Therefore, a two-tier cooperative caching mechanism is proposed, such that sensory data of the most popular are cached at the sink node, and these data can be reused for answering the forthcoming queries. This strategy can reduce the energy consumption of answering concurrent queries to a large extent. Specifically, the popularity of sensory data, which are interested in most queries in the most recent time slots, is the highest. These sensory data are assumed to be interested mostly in the forthcoming queries and are cached in the sink node. On the other hand, as for a grid cell that may not be interested in queries at this moment, sensory data in this grid cell are cached locally in the memory space of the corresponding head node. A flag, which indicates that sensory data have been varied significantly,

is cached in the head node. When this grid cell is interested in queries, sensory data of these sensor nodes, whose flags indicate a dramatic variation, are retrieved from the network in real time.

- Extensive simulations are conducted for evaluating the effectiveness and efficiency of the proposed algorithms. The experimental results show that the technique proposed in this article outperforms another technique proposed by Zhou *et al.* [33] where multiple-attribute query processing is also explored. Generally, our technique is more efficient than [33] in reducing the communication energy consumption and increasing the network capability, especially when the number of queries and the number of attributes interested in these queries are relatively large.

The rest of this article is organized as follows. Section 2 introduces the energy model used in the following. Section 3 presents the index tree construction algorithm and the network caching model. Section 4 proposes our cooperative caching mechanism for facilitating query answering. Section 5 evaluates the technique developed in the previous sections. Section 6 reviews and compares traditional techniques, and Section 7 concludes this work.

2. Preliminary: Energy Model

Several protocols for wireless sensor networks have been proposed leveraging the assumption made for the radio characteristics in the transmission and receiving modes. Without loss of generality, we apply the well-adopted first order radio model [34] in this article, for the computation of the energy consumption, where the parameters are presented in Table 1. In this model, the energy consumed for running the transmitter or receiver circuitry E_{elec} is set to 50 nJ/bit, while that for the transmit amplifier ϵ_{amp} is set to 100 pJ/bit/m^2. The energy consumption(s) $E_{Tx}(k,d)$ (or $E_{Rx}(k)$) for transmitting (or receiving) a packet of k bits within a distance d is (are) specified by the following equations:

$$E_{Tx}(k,d) = E_{elec} \times k + \epsilon_{amp} \times k \times d^n \tag{1}$$

$$E_{Rx}(k) = E_{elec} \times k \tag{2}$$

Table 1. Parameters in the energy model [34].

Name	Description
E_{elec}	Energy consumption constant of the transmitterand receiver electronics
ϵ_{amp}	Energy consumption constant of the transmission amplifier
k	The number of bits in one pocket
d	The distance of transmission
n	The attenuation index of transmission
r	The communication radius of sensor nodes
$E_{Tx}(k,d)$	The energy consumed to transmit a k bit packet to a distance d
$E_{Rx}(k)$	The energy consumed to receive a k bit packet
$E_{ij}(k)$	Energy consumption for transmitting a k bit packet from a node i to a neighboring node j

Consequently, the energy consumption $E_{ij}(k)$ for transmitting a packet of k bits from a sensor node i to a neighboring sensor node j is computed as follows:

$$E_{ij}(k) \quad = E_{Tx}(k,d) + E_{Rx}(k) = \begin{cases} E_{elec} \times k + \epsilon_{amp} \times k \times d^n & \text{if } j \text{ is } SN \\ 2 \times E_{elec} \times k + \epsilon_{amp} \times k \times d^n & \text{otherwise} \end{cases} \tag{3}$$

Note that the energy consumption of transmitting a packet to a sensor node is different from that to the sink node (SN, or called the base station), since SN is assumed to have no energy constraint, and the cost of receiving a packet is ignored in this model [34]. The parameter d refers to the distance between sensor nodes i and j (or SN). The energy of transmitting a packet from a sensor node i to another j is assumed the same as that of transmitting a packet from j to i, *i.e.*, $E_{ij}(k) = E_{ji}(k)$. The parameter n, which refers to the attenuation index of transmission as presented in Table 1, is determined by the surrounding environment. If sensor nodes in the network are barrier-free when forwarding packets, n is set to two. Otherwise, n is set to a value between three and five, when sensor nodes for long-distance transmission are distributed in the area of buildings and a vegetation cover. Without loss of generality, the network is assumed to be deployed in an area that is barrier-free, and n is set to two in our experiments in Section 5.

As an example, Figure 1 shows part of our sample network region, as shown in Figure 2, where 13 sensor nodes are deployed in this sub-region. The lines of arrows reflect the fact that sensor nodes in neighboring grid cells are within their communication radius r. Consequently, the energy consumed for forwarding a packet with the size of k bits from a sensor node (e.g., 47) to a neighboring one (e.g., 49) is computed as $E_{ij}(k) = 2 \times E_{elec} \times k + \epsilon_{amp} \times k \times d^n = 2 \times 50 \times k + 0.1 \times k \times d^2$, where the parameter d represents the geographical distance between sensor Nodes 47 and 49.

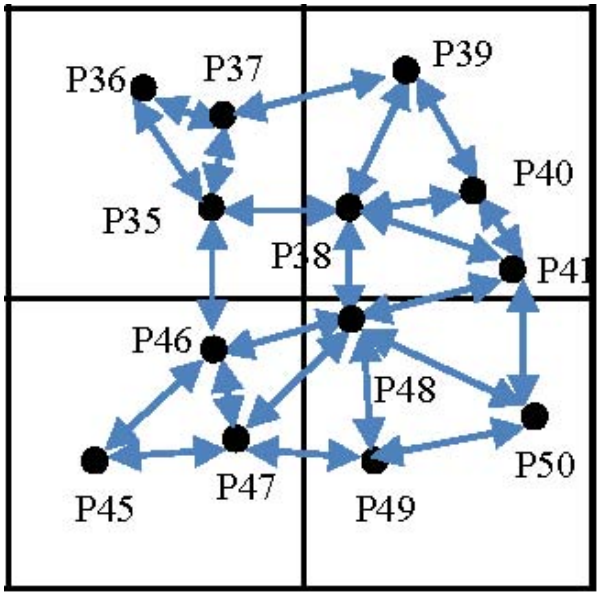

Figure 1. An example of energy consumption for the transmission of packets, where the sub-region is part of the network region, as shown in Figure 2.

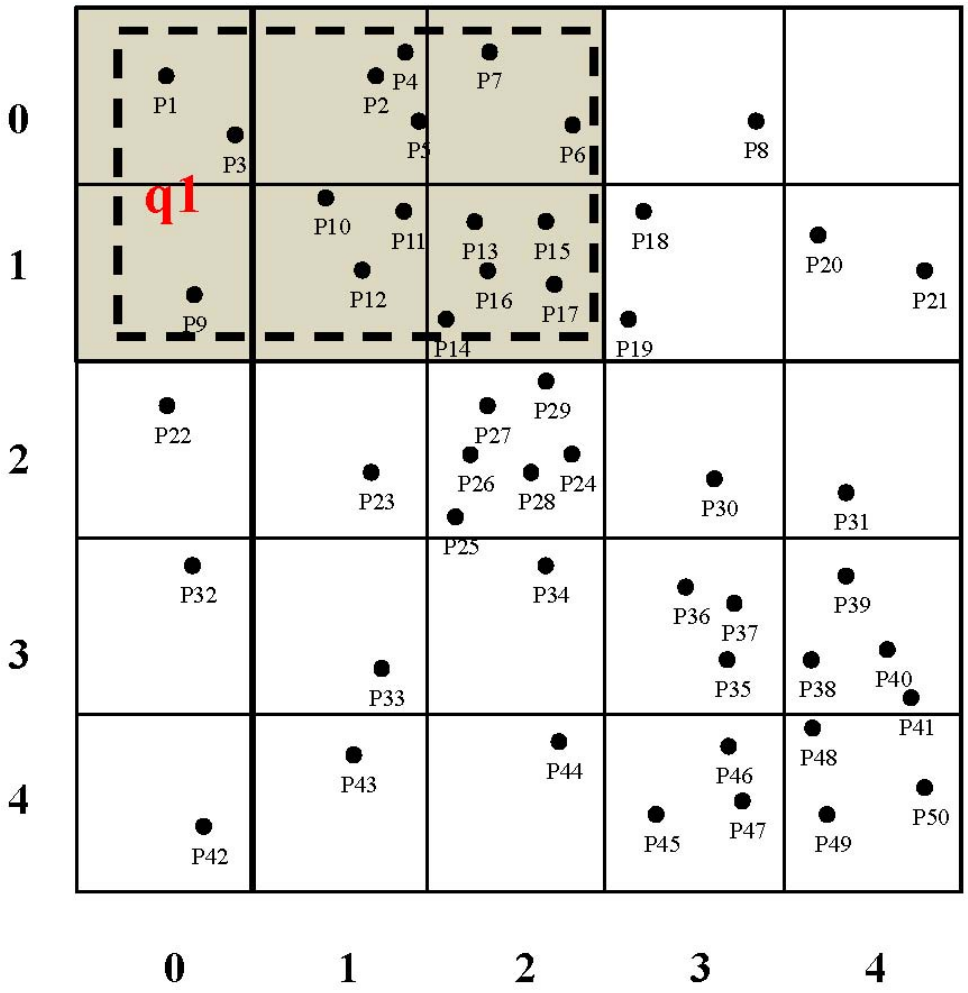

Figure 2. An example of grid division, where 50 sensor nodes are deployed unevenly in the network region and several kinds of attributes are assumed sensed by these sensor nodes. The network region is divided into 25 square grid cells, which are the same in geographical size. The region of a query (for instance, $q1$) is rewritten into a set of grid cells. For instance, $q1.qr$ can be rewritten into a set of grid cells of $\{gc_0, gc_1, gc_2, gc_5, gc_6, gc_7\}$.

3. Index Tree Construction and Network Caching Model

This section proposes an index tree construction algorithm leveraging the method developed by Zhou *et al.* [33] and introduces the network caching model, which is used to facilitate the query processing in Section 4.

3.1. Index Tree Construction

A tree-based routing structure has been used widely in WSNs for supporting the data gathering, aggregation and transmission in a multi-hop manner [35,36]. Leveraging the index tree construction algorithm developed in our previous work [33], we develop a novel index tree for organizing sensor nodes in a balanced manner, which is better at facilitating the query processing when multiple kinds of attributes are interested according to certain requirements of domain applications. Similar to the assumptions made in our previous work [33], sensor nodes are assumed in a skewed distribution, where sensor nodes are

dense in some sub-regions of the network, while they are sparse in the others. In fact, sensor nodes are distributed unevenly in real-world applications, including bridge or traffic monitoring scenarios [3]. An example of sensor nodes in a skewed distribution is shown in Figure 2, where sensor nodes located in the upper center and the right bottom of the network region are dense, while the others are sparse. A sensor node is assumed accompanied by one sensing equipment for sensing a certain attribute, such as humidity, temperature, gas flow, *etc.* Various sensor nodes with different sensing equipment are able to sense diverse attributes. The index tree construction algorithm proposed by Algorithm 1 in our previous work [33] works as follows:

- The network region is divided into square grid cells whose side-length is set to $\sqrt{2}r$, and an inverted file [37] is attached to each grid cell for specifying the attributes to be sensed by the sensor nodes therein. An example of grid cell division is shown in Figure 2, where the network is divided into 25 grid cells, and a two-dimensional matrix is adopted to represent the coordinates of these grid cells. Actually, grid cells correspond to the leaf nodes of the index tree to be constructed.
- The weight between two neighboring grid cells or sub-regions is calculated according to the mechanism proposed by the Algorithm 2 in our previous work [33]. This weight specifies the energy consumption of forwarding the same size of message between neighboring grid cells. Intuitively, sensor nodes in neighboring grid cells contribute to this weight calculation when the Euclidean distance between them is no more than the communication radius of sensor nodes r.
- Neighboring grid cells or sub-regions are merged in a pair-wise fashion when the weight between the candidates is the greatest, and the inverted file is processed accordingly. This merging procedure iterates until the root node of the index tree, which is a binary tree in shape, is constructed. Note that this merging strategy makes adjacent sub-regions, which may induce relatively larger energy consumption when forwarding messages in between, to be included by different children. Consequently, the energy consumption for routing data packets along the paths specified by this index tree is balanced somehow.

Generally, the Algorithm 1 in our previous work [33] constructs a tree that may be unbalanced, especially when sensor nodes are distributed unevenly in the network. Since [33] does not cache sensory data in leaf head nodes, leaf head nodes gather sensory data from sensor nodes and route these data to the SN. An unbalanced tree for a skewed distribution of sensor nodes prolongs the network lifetime, as evidenced by the experimental evaluation conducted in [33]. On the other hand, leaf head nodes require caching sensory data locally, as presented by Section 4 in this article, and sensory data, whose variation is beyond an allowed threshold, are not required to be routed to the SN. Therefore, when sensory data of most sensor nodes do not vary dramatically, the effort for routing sensory data to the SN should be lighter than that of [33] in the network, but the effort of leaf head nodes should be heavier due to the caching of sensory data locally. Intuitively, an unbalanced tree should a have much greater number of head nodes whose children include sensor nodes and head nodes, while all head nodes should be leaf head nodes for a half tree [38]. Therefore, more hops are to be involved when routing sensory data to the sink node in a hop-by-hop manner, especially when the path is longer in hops. To facilitate the caching mechanism upon leaf head nodes as presented in Section 4, a relatively balanced tree is constructed for organizing sensor nodes in this article.

Algorithm 1 Index tree construction.

Require: LN_{set}: the set of leaf nodes with inverted files

Ensure: rt: the root of the index tree

1: $TN_{set} \leftarrow$ set of leaf nodes and initially set to LN_{set}
2: $nTN_{set} \leftarrow$ set of tree nodes recording newly-merged nodes
3: **while** $|TN_{set}| > 1$ **do**
4: $wgt(nd_1, nd_2) \leftarrow calNgbNdWgt(nd_1, nd_2)$ as presented by Algorithm 2 in our previous work [33], where nd_1 and nd_2 are neighboring nodes in TN_{set}
5: **while** $\exists\ nd_1, nd_2 \in TN_{set}$: nd_1 and nd_2 are neighboring nodes **and** $wgt(nd_1, nd_2)$ is the biggest **do**
6: $tn \leftarrow$ merge tree nodes nd_1 and nd_2
7: $nd_1, nd_2 \leftarrow$ children of tn
8: $tn.IvtF \leftarrow nd_1.IvtF \cup nd_2.IvtF$
9: $TN_{set} \leftarrow TN_{set}$ - $\{nd_1, nd_2\}$
10: $nTN_{set} \leftarrow nTN_{set} \cup \{tn\}$
11: **end while**
12: $TN_{set} \leftarrow nTN_{set} \cup TN_{set}$
13: $nTN_{set} \leftarrow \emptyset$
14: **end while**
15: $rt \leftarrow tn$

Algorithm 2 Data synchronization.

Require: SN_{IN}: a set of sensor nodes that are contained in an intermediate node (IN)

Ensure: sensory data cached in IN are updated

1: **for** each $v_{sn} \in SN_{IN}$ **do**
2: $v_{sn}.val_{vsn}^{df} \leftarrow |v_{sn}.val_{vsn}^{cur}$ - $v_{sn}.val_{vsn}|$
3: **if** $v_{sn}.val_{vsn}^{df} > thrh_{atr}$ **then**
4: $v_{sn}.val_{vsn} \leftarrow v_{sn}.val_{vsn}^{cur}$
5: $IN.vec_{IN}^{v_{sn}}.ts_{fmS} \leftarrow$ current time slot
6: **if** $v_{sn}.sd_{stt}$ is active **then**
7: $IN.vec_{IN}^{v_{sn}}.val_{vsn} \leftarrow v_{sn}.val_{sn}^{cur}$
8: **else**
9: $IN.vec_{IN}^{v_{sn}}.val_{vsn} \leftarrow null$
10: **end if**
11: **end if**
12: **end for**

As previously mentioned, the index tree to be constructed in this article should be a relatively balanced tree for a skewed distribution, rather than an unbalanced tree, as developed in our previous work [33]. The difference lies mainly in the tree node merging strategy. As presented in Algorithm 1, after dividing the network region into square grid cells whose side-length is $\sqrt{2}r$, we

firstly get the set of leaf nodes (denoted TN_{set}) with inverted files as presented by Algorithm 1 (Lines 1–13) in our previous work [33] (Line 1). Note that these leaf nodes corresponds to grid cells, rather than sensor nodes in the network. The weight (denoted $wgt(nd1, nd2)$) between neighboring nodes (denoted nd_1 and nd_2), which reflects the energy consumption when routing sensory data from one node to another, is calculated using the function $calNgbNdWgt(nd_1, nd_2)$. As presented by Algorithm 2 in our previous work [33] (Line 4), this function returns the sum of weights among all pairs of neighboring grid cells in the corresponding neighboring sub-regions. If there are neighboring nodes (*i.e.*, nd_1 and nd_2) in TN_{set} that can be merged, in other words, $wgt(nd1, nd2)$ is the biggest (Line 5), nd_1 and nd_2 are merged as a newly-generated node tn (Lines 6–7), and the inverted files (denoted $tn.IvtF$, for instance) are processed accordingly (Line 8). Note that tn corresponds to a sub-region in the network, which is the merger of sub-regions for nd_1 and nd_2. After this merging procedure, nd_1 and nd_2 are removed from TN_{set} (Line 9), while tn is inserted into nTN_{set} as a candidate for the next merging procedure (Line 10). This procedure iterates until all neighboring nodes in TN_{set} have been examined and processed. Thereafter, TN_{set} and nTN_{set} are updated accordingly (Lines 12–13). The symbol $\emptyset$ in Line 13 means an empty set. This tree construction procedure terminates one there is only one node left in TN_{set} (Line 3), which corresponds to the root node of the index tree (Line 15). An example of the index tree constructed through this algorithm is shown in Figure 3, which is a binary tree and more balanced than the tree as shown in Figure 5 in our previous work [33]. Specifically, 25 grid cells, as shown in Figure 2, correspond to the leaf nodes, and sub-regions composed of multiple grid cells correspond to non-leaf nodes, in the index tree. This index tree construction is to facilitate the popularity-based cooperative caching mechanism to be presented in the following sections. The time complexity of Algorithm 1 is $O(n \times \log_2 n)$, where n is the number of leaf nodes in, while $\log_2 n$ is the height of the index tree.

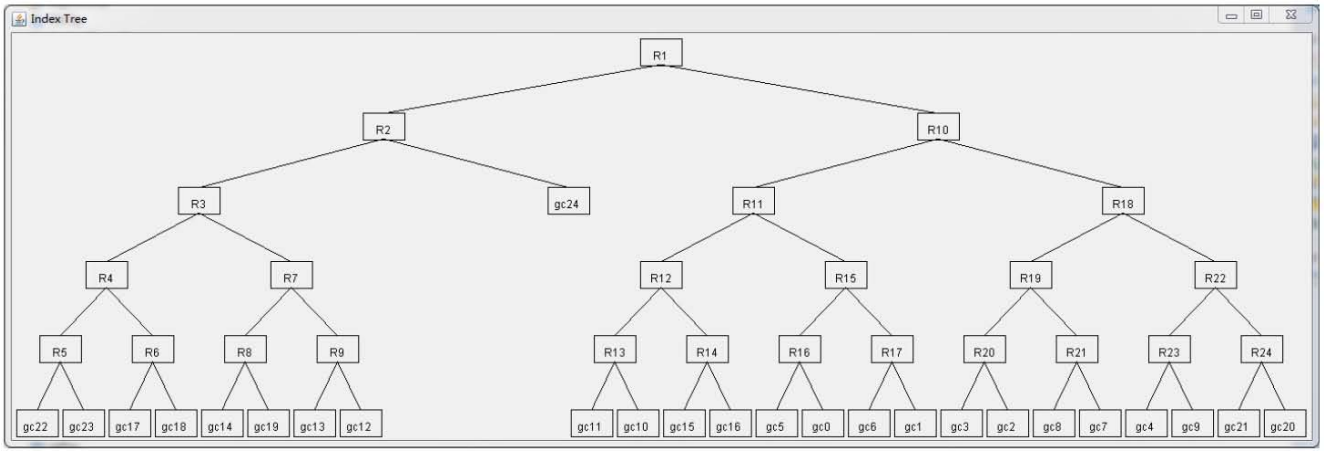

Figure 3. An example of the index tree constructed through Algorithm 1, where 1000 sensor nodes are deployed in the network region, and the skewness degree is set to 60%. The leaf nodes in the index tree correspond to the grid cells (e.g., gc_{24}), and non-leaf nodes (e.g., R_5) correspond to sub-regions containing multiple grid cells (e.g., gc_{22} and gc_{23}).

Generally, a node in an index tree corresponds to a sub-region containing a subset of tree nodes and/or sensor nodes, and these tree nodes are responsible for query propaga-

tion and sensory data routing to the *SN*. The low-energy adaptive clustering hierarchy protocol (*LEACH*) [39] is adopted for the head node selection in these sub-regions or grid cells, where these sub-regions or grid cells correspond to the clusters as presented by Heinzelman *et al.* [39]. LEACH incorporates randomized rotation of high-energy sensor nodes as the head nodes to avoid draining the energy of head nodes. Consequently, the energy consumption of being a cluster head node is distributed and balanced among sensor nodes. This strategy facilitates the prolongation of the network lifetime to some extent. Optimal head nodes should be located at the center of a cluster [40,41]. Therefore, a higher priority is given to sensor nodes that are closer to the center of sub-regions or grid cells when voting for head nodes.

3.2. Two-Tier Cooperative Caching Model

Applications leveraging WSNs may require gathering sensory data in a short latency, while minimizing the energy consumption of the network. Queries issued periodically and continuously for gathering sensory data of multiple attributes should induce high communication cost, which may be above the network capability. Without loss of generality, we divide the time duration into time slots as our previous work [42], and queries are assumed to be conducted in batches in each time slot. Note that sensory data in some applications, such as health or wildlife monitoring, may not change dramatically within a certain time duration. Besides, some applications may tolerate a bias of sensory data, when the variation of these data can satisfy a certain constraint. These suggest that sensory data may be valid for the applications within some time slots after the sensing time point. Therefore, these sensory data may be appropriate to reuse for answering the forthcoming queries, rather than fetching from the network in a real-time fashion [42]. To facilitate this sensory data reusability strategy, this article proposes a cooperative data caching mechanism, where sensory data are cached in the memory of: (i) the SN; and (ii) head nodes of grid cells, which correspond to the leaf nodes in the index tree. We use the notion of intermediate nodes (INs) to represent these leaf head nodes in the following. Generally, the SN, which has a larger memory space and more capability in computation than sensor nodes, is responsible for caching the bulk (may be not all) of sensory data from the network. An IN is required to cache sensory data of sensor nodes in the corresponding grid cell. Since there are usually a limited number of sensor nodes in each grid cell, an IN is assumed to have enough memory space and computational capability for processing sensory data of all sensor nodes in the corresponding grid cell. To facilitate the query processing mechanism in the following sections, we define a wireless sensor network as follows:

Definition 1 (WSN). *A wireless sensor network is a tuple* $wsn = (SN, V_{IN}, V_{sn}, ATR)$*, where:*

- *SN is the sink node of this network.*
- V_{IN} *is a set of intermediate nodes, which are responsible for handling sensory data in grid cells.*
- V_{sn} *is a set of sensor nodes in the network.*
- ATR *is a set of attributes to be sensed by* V_{sn}*.*

Given a sensor node $v_{sn} \in V_{sn}$, v_{sn} is defined in terms of the vector:

$$v_{sn} =< sd_{id}, sd_{atr}, sd_{stt}, val_{vsn} > \quad (4)$$

where sd_{id} means the identifier of this sensor node v_{sn}, $sd_{atr} \in ATR$ represents the single attribute to be sensed by v_{sn}, sd_{stt} specifies the status of v_{sn}, which can be active or inactive, and val_{vsn} is the sensory data that may be cached in the corresponding intermediate node $v_{IN} \in V_{IN}$ and the SN. Note that val_{vsn} is not the sensory data val_{vsn}^{cur} at this moment, and val_{vsn} is to be replaced by val_{vsn}^{cur} only when the bias between val_{vsn} and val_{vsn}^{cur} is above an allowed threshold $thrd$. In this case, v_{sn} should:

- report val_{vsn}^{cur} to the corresponding v_{IN} for the update of this sensory data in v_{IN} when v_{sn} is in the status of active, which means that v_{IN} is interested in queries in recent time slots.
  Note that the SN should synchronize with v_{IN} for retrieving the last sensory data when a query interest a grid cell v_{IN}. When v_{IN} is not interested in queries for a certain number of recent time slots, v_{IN} is assumed not interested currently, and all sensor nodes in v_{IN} are set to the status of inactive. On the other hand, when v_{IN} is interested in a query, all sensor nodes in v_{IN} are reset to the status of active immediately.
- Otherwise, v_{sn} should report a sensory data change notification message to v_{IN} when v_{sn} is in the status of inactive, and thereafter, v_{IN} will invalidate the sensory data in the cache.

Note that $thrd$ is determined according to: (i) the kind of attribute to be sensed by v_{sn}; and (ii) the specific requirement of certain applications. As for the sensory data synchronization procedure, we refer to Section 4.1 for the details.

An intermediate node v_{IN} should cache sensory data of all sensor nodes in the corresponding grid cell, in terms of the vector for a sensor node v_{sn}:

$$vec_{IN}^{vsn} =< sd_{id}, sd_{atr}, sd_{stt}, val_{vsn}, ts_{fmS}, ts_{toSN} > \tag{5}$$

where sd_{id}, sd_{atr}, sd_{stt} and val_{vsn} are the same as those of v_{sn}, respectively. ts_{fmS} records the time slot when val_{vsn} is reported from the sensor node, while ts_{toSN} records the time slot when val_{vsn} is retrieved by SN. When ts_{fmS} is the same as ts_{toSN}, the last sensory datum of v_{sn} has been cached in the SN if the SN has enough memory space. Otherwise, the cached sensory data in the SN is not synchronized with that in the corresponding IN. Note that $vec_{IN}^{vsn}.val_{vsn}$ should be set to null when v_{sn} reports a sensory data change notification message to the IN.

The SN caches sensory data of sensor nodes in terms of the vector:

$$vec_{SN}^{vsn} =< sd_{id}, gc_{id}, sd_{atr}, val_{vsn} > \tag{6}$$

where sd_{id}, sd_{atr} and val_{vsn} are the same as those of sensor node v_{sn}, respectively. gc_{id} refers to the ID of the corresponding grid cell where v_{sn} lies. Since the SN has a limited storage capability, sensory data of sensor nodes, which are the most popular according to the recent query history, are to be cached in the SN. The popularity computation of grid cells is presented in Section 4.2.

4. Query Processing with Cache Mechanism

Given a network deployed in a certain region where multiple kinds of attributes are interested, queries are to be issued periodically and continuously according to the requirement of certain applications. Generally, a multiple-attribute query can be described in terms of a three-dimensional vector, including:

(i) a query region; (ii) a set of interested attributes; and (iii) a certain time slot. A query region is typically represented by a rectangle. As mentioned in Section 3.1, the network region is divided into grid cells. Consequently, a query region is transferred to a set of grid cells with the minimum number, which can cover the rectangle prescribed by this query. Formally, a query is defined as follows:

Definition 2 (Query). *A query is a tuple* $q = (tm, qr, ATR)$*, where:*

- tm *is the time slot when the query* q *is issued.*
- qr *is the query region, which is represented by a set of corresponding square grid cells.*
- ATR *is a set of attributes interested in* q*.*

Given a query $q = <tm, qr, ATR_q>$ where $q.qr$ is rewritten as a set of grid cells GC_q (*i.e.*, $q.qr = GC_q$), q returns sensory data at $q.tm$ of sensor nodes: $\forall\ v_{sn} \in GC_q$: ($v_{sn}.sd_{id}$ is contained by a grid cell within GC_q) $\wedge$ ($v_{sn}.sd_{atr} \in ATR_q$).

It is worth mentioning that in certain domain applications, including health monitoring, queries are typically issued continuously and periodically. The points of interest should be within certain sub-regions for certain contiguous time slots, while evolving moderately to neighboring sub-regions [15]. In this context, when the sub-regions for the queries in contiguous time slots have overlapping sub-regions, query results in the previous time slots should be valid and (partially) reusable for answering the forthcoming queries. On the other hand, queries are answered independently by the SN. Generally, for a certain query q to be processed by the SN, q is answered by a cooperative caching mechanism, including the following steps:

- Step 1: The SN determines a set of grid cells GC_q of a minimum number that can cover the query sub-region $q.qr$.
- Step 2: The SN synchronizes with the intermediate nodes (INs), which are actually the head nodes of $gc_q \in GC_q$ in the index tree, for retrieving the last sensory data. It is worth mentioning that the SN may cache sensory data of some, but not all, INs. Intuitively, a flag is adopted for specifying whether sensory data of a certain IN has been cached in the memory space of SN or not.
- Step 3: An IN examines the freshness of cached sensory data for each sensor node (denoted v_{sn}) in the corresponding grid cell gc_q.
  - When $v_{sn}.ts_{fmS} = v_{sn}.ts_{toSN}$, which suggests that sensory data cached in IN is up-to-date, and is synchronized with that cached in the SN, a flag indicating this scenario is sent to the SN, whereas sensory data of v_{sn} do not need to be forwarded to the SN.
  - Otherwise, sensory data cached in the IN are not up-to-date. Consequently, the head node of the IN requests to retrieve the last sensory datum from v_{sn}, which is to be routed to the SN afterwards.

  Note that the status of v_{sn} is set to active when v_{sn} is interested in queries currently and/or in recent time slots. The reader is referred to Section 4.1 for the details of the sensory data synchronization mechanism between an IN and the sensor nodes contained therein.
- Step 4: When the SN gets sensory data of all sensor nodes in each grid cell $gc_q \in GC_q$, the answer to the query q is aggregated. Sensory data, which were cached in the memory space of the SN and are not consistent with the last ones retrieved from INs, are updated accordingly. Note that the

SN usually has a constraint in storage and computational capabilities. As presented in Section 4.2, when sensory data of some grid cells are retrieved from INs, sensory data cached in the SN, which may not be reused for answering the forthcoming queries, should be removed, and the released storage capability can be adopted for caching more popular sensory data.

As discussed at Steps 2–4, the technical details of our two-tier cooperative caching mechanism on the SN and IN for query answering are presented in Section 4.3.

4.1. Sensory Data Synchronization for INs and Corresponding Sensor Nodes

As discussed, in certain applications of WSNs, the points of interest may be within certain regions in certain time durations, while evolving to neighboring sub-regions moderately and continuously. This suggests that a certain grid cell may be interested in applications of some time durations, while not in the others. To reduce the energy consumption, sensor nodes, which do not contribute to answering the queries in a few recent time slots, are set to the status of inactive. Therefore, these sensor nodes are not required to send sensory data to the IN at each time slot afterwards. Instead, when their sensory data have been changed dramatically and the variation is not tolerable for applications, a flag indicating this situation is sent to the IN. It is worth mentioning that the strategy of active and inactive is different from adaptive sleeping [43,44] and duty-cycle [45,46] strategies, where sensor nodes do not sense environmental variables and respond to queries. For simplicity, we assume that sensor nodes decide to send either sensory data or a flag (data of a bit usually) to the IN according to the status of active or inactive. This strategy decreases the amount of data to be sent and thus reduces the energy consumption. In fact, adaptive sleeping and duty-cycle strategies complement our technique, and when applied, energy consumption should be further reduced.

The sensory data synchronization strategy for the IN and contained sensor nodes is presented in Algorithm 2. For each sensor node v_{sn} in each IN, v_{sn} examines whether the sensory data at this moment (denoted $v_{sn}.val_{vsn}^{cur}$) have been changed dramatically with respect to the sensory data reported to the IN (denoted $v_{sn}.val_{vsn}$) in previous time slots (Line 2). If the variation $v_{sn}.val_{vsn}^{df}$, which is represented by the absolute value of the difference between $v_{sn}.val_{vsn}^{cur}$ and $v_{sn}.val_{vsn}$, is above an allowed and pre-specified threshold (denoted $thrh_{atr}$), v_{sn} should report this data change situation to the IN (Line 3). In this case, v_{sn} sets its reported sensory data $v_{sn}.val_{vsn}$ as the current value $v_{sn}.val_{vsn}^{cur}$ (Line 4). $IN.vec_{IN}^{v_{sn}}.ts_{fmS}$ is set to the current time slot (Line 5). $v_{sn}.val_{vsn}^{cur}$ is reported to the IN for updating the cached data when v_{sn} is in the status of active. To be specific, $IN.vec_{IN}^{v_{sn}}.val_{vsn}$ is replaced by $v_{sn}.val_{sn}^{cur}$ (Lines 6–7). Otherwise, a flag indicating this dramatic data change situation is reported to the IN, and the corresponding sensory data for v_{sn} cached in IN is set to null (Lines 8–9). Consequently, sensory data cached in the IN are synchronized with those of sensor nodes contained in this IN. Note that this data synchronization procedure is performed at each time slot, which facilitates the answering of queries as presented in Section 4.3.

The time complexity of Algorithm 2 is $O(m_{gc})$, where m_{gc} is the maximum number of sensor nodes contained in a grid cell. The worst case occurs when sensory data of all sensor nodes have been changed dramatically, and thus, all sensory data cached in the IN have to be updated consequently.

4.2. Popularity-Based Sensory Data Replacement Mechanism in the SN

As mentioned in Section 3.2, sensory data of certain sensor nodes may not change dramatically in certain time durations, and domain applications may work well when the bias of sensory data is beyond a certain threshold with respect to the data in real time. In this setting, sensory data cached in the memory space of the SN may be appropriate to be (partially) reused for answering the forthcoming queries, rather than retrieving from the network in real time. This mechanism should reduce the effort of sensory data sensing, gathering and routing and, thus, should prolong the network lifetime to some extent. Note that the SN usually has limited storage and computational capabilities, which may not be capable of caching sensory data of all sensor nodes in the network. Therefore, sensory data that have a high possibility of being reused for answering the forthcoming queries should be cached. Generally, queries are issued at each time slot. When fresh sensory data of sensor nodes are retrieved from the network, these fresh sensory data should be used for updating the obsolete counterparts if cached in the SN previously. On the other hand, sensory data cached in the SN, which may not contribute to the forthcoming queries, should be removed, and the released memory space is used for caching the fresh sensory data. This section proposes a mechanism for sensory data replacement in the cache of the SN depending on the popularity of sensory data. Note that the network is divided into square grid cells in this article, and a grid cell (denoted gd), with an interested attribute (denoted atr), is considered as an atomic unit for query processing. Intuitively, the vector $vc = <gd, atr>$ is applied to represent the index of the set of sensory data to be cached in the SN.

Given a set of sensory data represented by the vector $vc_i = <gc_i, atr_i>$, where these data are (i) cached in the SN or (ii) were not cached in the SN and are retrieved from the network at this moment, we calculate the popularity pop^i_{vc} of vc_i according to (i) the history of queries conducted in the previous k time slots and (ii) the size of sensory data contained by vc_i. A large value of pop^i_{vc} means that vc_i is higher in possibility of being cached in the SN and of being interested in the queries in the consequent time slots. Specifically, pop^i_{vc} is calculated using the following formula:

$$pop^i_{vc} = \frac{1}{sz_{vc_i}} \times \sum_{j=1}^{k} (\alpha^j \times \frac{f^j_i}{f^j_{all}}) \tag{7}$$

where sz_{vc_i} means the size of sensory data in vc_i, which is proportional to the number of sensor nodes contained in gc_i accompanying the attribute atr_i and the size of sensory data for the attribute atr_i. This indicates that the larger the size of vc_i is (*i.e.*, $\frac{1}{sz_{vc_i}}$ is smaller), the higher the possibility that vc_i is removed from the SN. f^j_i represents the number of queries that are interested in vc_i at the time slot j, while f^j_{all} represents the number of all vectors $vc = <gd, atr>$ interested in all queries issued at the time slot j. Generally, the more frequently that the vc_i is interested in queries, the larger the popularity pop^i_{vc} is. The parameter α corresponds to an attenuation coefficient, which is set to a value between zero and one. In fact, α reflects the importance of queries conducted in a recent, while different, time slot. Intuitively, the more recent queries the vc_i is interested in, the larger the popularity pop^i_{vc} is.

Given a set of $VC = \{vc_1, vc_2, \cdots\}$ that are gathered at a certain time slot, the popularity pop^i_{vc} of $vc_i \in VC$ is calculated using Formula 7. pop^i_{vc} are ranked according to their values. Sensory data of vc_i are cached in the SN until not enough storage capability in the SN is available for the next. These cached sensory data are used for facilitating the cooperative caching mechanism, as detailed in the following.

4.3. Query Processing with Two-Tier Cooperative Caching Mechanism

Leveraging sensory data cached in the memory space of INs and the SN, we propose a two-tier cooperative caching mechanism for answering periodic queries. As presented in our previous work [33], the network region is divided into square grid cells. A two-dimensional matrix is used to represent grid cells, where each grid cell is accompanied by an identifier (denoted gc_{id}). gc_{id} is computed based on the value of (i) row (denoted *row*) and column (denoted *col*) coordinates of the grid cell and (ii) the columns of the grid cell in the matrix (denoted *cols*). Specifically, $gc_{id} = row \times cols + col$. An example is shown in Figure 2, where pi (i = 1, 2, ...) represents sensor nodes sensing various kinds of attributes. As for the grid cell containing sensor node $p17$, its grid cell ID is computed as 7 = 1 $\times$ 5 + 2, and this grid cell is denoted as gc_7. The first grid cell, which contains $p1$, as shown in Figure 2, is denoted as gc_0.

Given a query q to be answered, q is rewritten for determining a minimum set of grid cells that $q.qr$ covers. When a grid cell is intersected with $q.qr$, this grid cell is counted. An example is shown in Figure 2, where the query region is represented using a rectangle with dotted lines, and grid cells for $q.qr$ are marked in gray. The attributes that $q.ATR$ includes are represented using a bitmap. An example is shown in Table 2, where one means that the query interests in the corresponding attribute, while zero means it is not. The parameter k represents the number of attributes to be sensed in a certain network.

Table 2. An example of a bitmap for attributes interested in a certain query q, where atr_i specifies the i-th attribute. The value 1 means that a certain attribute (e.g., atr_3) is interested in this query q, while 0 means it is not.

Attribute	atr_1	atr_2	atr_3	atr_4	...	atr_k
Bit	1	0	1	0	...	0

When answering a query q, sensory data of multiple attributes in a certain grid cell should be gathered and aggregated once. An example is the attributes atr_1 and atr_3 with respect to grid cells gc_0, gc_1, gc_2, gc_5, gc_6 and gc_7, as shown in Figure 2 and Table 2. To facilitate the query processing, a GC-attribute table is adopted to clearly represent this relation for grid cells and interested attributes. For instance, Table 3 shows the relation for grid cells (*i.e.*, gc_0, gc_1, gc_2, gc_5, gc_6 and gc_7) and interested attributes (*i.e.*, atr_1 and atr_3). It is worth mentioning that a grid cell with a single attribute of interest is considered as the atomic unit for query processing and the sensory data caching mechanism as presented in Section 4.2. Generally, Table 3 is an example for the relation of grid cells and attributes for a single query, where one means that the corresponding attribute in the certain grid cell is interested in the query, whereas zero means it is not. A table in this format can also be applied to specify the relation of grid cells and attributes aggregated for multiple queries that are to be processed concurrently at a certain time slot. For simplicity, as presented by Algorithm 3, a single query is considered for the query processing procedure leveraging the cooperative caching mechanism, whereas multiple queries can be handled in a similar fashion. Besides, a table, which is named the GC-SNCache table and an example shown in Table 4, is used to represent sensory data of grid cells that are cached in the SN currently, where one means that the corresponding attribute in the certain grid cell has been cached in the memory space of the SN, whereas zero means it is not. Note that the SN is usually limited in storage and computational

capabilities and should cache sensory data of grid cells, which have a high possibility of being covered by the forthcoming queries.

Table 3. An example of the GC-attribute table, where gc_i specifies the i-th grid cell and atr_j specifies the j-th attribute. The value 1 means that the sensor nodes in a certain grid cell (e.g., gc_1) are interested in a certain attribute (e.g., atr_3), while 0 means they are not.

	gc_0	gc_1	gc_2	gc_5	gc_6	gc_7
atr_1	0	1	0	0	1	1
atr_2	0	0	0	0	0	0
atr_3	1	1	1	1	1	0
...	...	...	...	...	...	...
atr_k	0	0	0	0	0	0

Table 4. An example of the GC-SNCache table, where gc_i specifies the i-th grid cell and atr_j specifies the j-th attribute. The value 1 means that sensory data for a certain grid cell (e.g., gc_1) with a certain attribute (e.g., atr_1) are cached in the memory of the SN, while 0 means they are not.

	gc_0	gc_1	gc_2	gc_5	gc_6	gc_7
atr_1	1	1	0	0	1	0
...	...	...	...	...	...	...
atr_i	0	1	0	1	1	0
...	...	...	...	...	...	...
atr_k	0	1	0	1	0	1

As presented by Algorithm 3, when a query q is issued at a certain time slot, q is rewritten and a set of grid cells (denoted GC_{set}^{q}) are retrieved that covers $q.qr$, as illustrated by Figure 2 (Line 1). If the attributes interested in the query q (denoted $q.ATR$) are not sensed by the sensor nodes in the sub-region corresponding to the tree node tn or the query region (*i.e.*, GC_{set}^{q}) and the sub-region contained in the tree node (denoted $tn.GC$) have no overlapping areas (Line 2), no sensory data should be returned, and the tree node tn does not contribute to the query q (Line 3). The symbol $\emptyset$ in Line 2 specifies an empty set. Otherwise, tn contains sensor nodes that can contribute to the answering of the query q.

Generally, two scenarios are considered leveraging the fact of whether tn corresponds to a leaf node in the index tree or not. When tn does not reflect a leaf node in the index tree (Line 5), the left and right children of tn are processed independently, and their results are represented as q_{rt}^{lf} and q_{rt}^{rt}, respectively (Lines 6–7). The querying result of q (denoted q_{rt}) is assembled as the aggregation of q_{rt}^{lf} and q_{rt}^{rt} (Line 8).

On the other hand, when tn reflects a leaf node in the index tree, tn corresponds to a grid cell actually. In this setting, the intermediate node (IN) with respect to tn is identified (Line 10), and the set of sensor nodes (denoted V_{sn}^{all}) contained in tn, which contribute to the answering of q, are retrieved (Line 11). The status of sensor nodes in V_{sn}^{all} is changed to active, when it is inactive currently (Line 12).

Algorithm 3 QueryProcCoopCaching

Require: q: a query issued at a certain time slot
tn: a node in the index tree
Ensure: q_{rt}: the result for the query q

1: $GC^{q}_{set} \leftarrow$ set of grid cells that are covered by $q.qr$
2: **if** $q.ATR \cap tn.IvtF = \emptyset$ **or** $tn.GC \cap GC^{q}_{set} = \emptyset$ **then**
3: **return**
4: **end if**
5: **if** tn has children **then**
6: $q^{lf}_{rt} \leftarrow$ *QueryProcCoopCaching*(q, $tn.lfCld$) when the left child $tn.lfCld$ exists
7: $q^{rt}_{rt} \leftarrow$ *QueryProcCoopCaching*(q, $tn.rtCld$) when the right child $tn.rtCld$ exists
8: $q_{rt} \leftarrow$ aggregation of q^{lf}_{rt} and q^{rt}_{rt}
9: **else**
10: IN $\leftarrow$ get the intermediate node that contains tn
11: $V^{all}_{sn} \leftarrow$ set of sensor nodes in tn, where $\forall\, v_{sn} \in V^{all}_{sn}$ **and** $v_{sn}.sd_{str} \in q.ATR$
12: $v_{sn}.sd_{stt} \leftarrow$ active, where $\forall\, v_{sn} \in V^{all}_{sn}$ **and** $v_{sn}.sd_{stt}$ = inactive
13: **for each** $v_{sn} \in V^{all}_{sn}$ **and** (*IN*.$v_{sn}.ts_{fmS} \neq$ *IN*.$v_{sn}.ts_{toSN}$ **or** *IN*.$v_{sn}.ts_{toSN} = null$) **do**
14: $V_{sn} \leftarrow V_{sn} \cup \{v_{sn}\}$
15: **end for**
16: IN retrieves sensory data from grid cell tn for V_{sn}, replaces cached sensory data in IN for V_{sn} and sets *IN*.$v_{sn}.ts_{fmS}$ and *IN*.$v_{sn}.ts_{toSN}$ to the current time slot for $\forall\, v_{sn} \in V_{sn}$
17: **if** grid cells for tn and $q.ATR$ are not cached in the SN according to the GC-SNCache table **then**
18: **for each** $v_{sn} \in V^{all}_{sn}$ **do**
19: $V_{sn} \leftarrow V_{sn} \cup \{v_{sn}\}$
20: **end for**
21: mark all grid cells for tn and $q.ATR$ as being cached in the GC-SNCache table
22: **end if**
23: **for each** $v_{sn} \in V_{sn}$ **do**
24: *SN*.$vec^{vsn}_{SN}.val_{vsn} \leftarrow$ *IN*.$vec^{vsn}_{IN}.val_{vsn}$, where $v_{sn}.sd_{id}$ = *SN*.$vec^{vsn}_{SN}.sd_{id}$ = *IN*.$vec^{vsn}_{IN}.sd_{id}$
25: **end for**
26: **for** $\forall\, v_{sn} \in V^{all}_{sn}$ **and** $v_{sn}.sd_{id}$ = *SN*.$vec^{vsn}_{SN}.sd_{id}$ **do**
27: $q_{rt} \leftarrow$ insert *SN*.$vec^{vsn}_{SN}.val_{vsn}$
28: **end for**
29: **end if**

For each sensor node $v_{sn} \in V^{all}_{sn}$, when the sensory data for v_{sn} cached at the corresponding IN are not aligned with those cached in the SN (indicated by *IN*.$v_{sn}.ts_{fmS} \neq$ *IN*.$v_{sn}.ts_{toSN}$ or *IN*.$v_{sn}.ts_{toSN}$ = $null$, where *IN*.$v_{sn}.ts_{fmS} \neq$ *IN*.$v_{sn}.ts_{toSN}$ suggests that the last sensory data provided by v_{sn} have not been synchronized with those cached in the SN, while *IN*.$v_{sn}.ts_{toSN}$ = $null$ suggests that the sensory data for v_{sn} have never been reported to the SN) (Line 13), the last sensory datum for v_{sn} should be retrieved from the network, and the time slots, which reflect the time when (i) sensor nodes report their sensory data to the IN (denoted ts_{fmS}) and (ii) the IN reports sensory data of certain sensor nodes to the SN (denoted ts_{toSN}), are updated accordingly

(Line 16). It is worth mentioning that, when the status of v_{sn} is inactive, the IN may not cache the sensory data for v_{sn}, according to our data synchronization mechanism presented by Algorithm 2. As for the sensor nodes whose cached sensory data are consistent between the SN and the corresponding IN, according to the GC-SNCache table, these sensory data are not required to be retrieved from the network and to be routed to the SN. Instead, the sensory data cached in the SN are adopted for answering the query q. This strategy should reduce the energy consumption, which may be unnecessary somehow. Otherwise, the GC-SNCache table is updated for showing the sensory data consistency between the SN and the corresponding IN at this moment (Lines 17 and 21). Note that when the SN does not cache the sensory data for sensor nodes in V_{sn}^{all} (Line 17), the sensory data of all sensor nodes in V_{sn}^{all} should be forwarded to the SN for answering the query q (Lines 18–20). Consequently, sensory data of all sensor nodes in V_{sn}^{all} are synchronized between the SN and the corresponding IN (Lines 23–25). The result of the query q, which is represented by q_{rt}, is assembled leveraging the sensory data cached in the SN accordingly (Lines 26–28).

The time complexity of Algorithm 3 is $O(n \times m)$, where n is the number of tree nodes to be examined in the index tree leveraging the inverted file, and m is the maximum number of sensor nodes in grid cells. The worst case occurs when all sensor node in the whole network are to be traversed. In this setting, all sensor nodes are required to report their sensory data to the SN.

5. Implementation and Evaluation

A prototype has been implemented in a Java program, and experiments have been conducted for evaluating our technique. In the following, we introduce the environment settings and discuss the results of our experiments.

5.1. Environmental Settings

Experiments are designed for evaluating our technique, and the factors considered include various skewness distributions for the network setting and various cache sizes in the SN. In the network setting, 1000 sensor nodes are generated and distributed unevenly with skewness degrees varying from 20%–80%. Intuitively, a skewness degree (denoted sd) is computed using the formula: $sd = \frac{dn-sn}{N}$, where dn and SN refer to the number of sensor nodes in dense and sparse sub-regions, respectively, while N is the number of sensor nodes deployed in the network (*i.e.*, $N = dn + sn$) [33]. For instance, assuming a network contains N (e.g., $N = 1000$) sensor nodes, the network region is divided into four sub-regions, which are rectangular in shape and the same in geographical size. A skewness degree is set to sd (e.g., $sd = 60\%$). Consequently, $n \times (1 - sd)$ ($1000 \times (1 - 60\%) = 400$) sensor nodes are deployed evenly in the whole network, while the remaining $n \times sd$ ($1000 \times 60\% = 600$) sensor nodes are distributed to any two dense sub-regions in a random fashion. Therefore, a network region with N sensor nodes, whose distribution follows a skewness degree sd, is constructed.

Experiments are performed on a desktop with an Intel(R) Core(TM) i5-2400 CPU at 3.10 GHz, a 4-GB memory and a 32-bit Windows system. Parameter settings for our experiments are presented in Table 5. Note that several, but typically not too many, kinds of attributes are interested in domain applications. Without loss of generality, 10 kinds of attributes are assumed interested in the experiments.

A sensor node is randomly assigned an attribute to be sensed, and each kind of attribute is sensed by around 1000/10 = 100 sensor nodes. The network region is set to 350 m $\times$ 350 m, which is divided into square grid cells with the same geographical size. The side-length of grid cells is set to $\sqrt{2}r$ [40], where r = 50 m is the communication radius of sensor nodes. The size of cache in the SN is set to a number between 500 and 900, which means that the SN can accommodate sensory data for 50–900 sensor nodes, respectively. The attenuation coefficient α and the number of preceding time slots k used in Equation (7) are set to 0.6 and four, respectively. Queries are issued every 2 min, and sensory data between INs and the corresponding sensor nodes are synchronized every 5 min, when the status of these sensor nodes is inactive. These parameters can be set to other values if appropriate.

Table 5. Parameters settings in the experiments.

Parameter Name	**Value**
Network region	350 m $\times$ 350 m
Skewness degree	20%–80%
Number of sensor nodes	1000
Number of attributes	10
Cache size in the SN	500–900
Communication radius (r)	50 m
Attenuation index of transmission (n)	2
Energy consumption constant for the transmit and receiver electronics (E_{elec})	50 nJ/bit
Energy consumption constant for the transmit amplifier (ϵ_{amp})	100 pJ/(bit $\times$ m^2)
Attenuation coefficient (α)	0.6
Number of time slots considered for computing the popularity of vectors (k)	4
Length of a time slot for query processing	2 min
Time interval for data synchronization	5 min
Time interval for head node reselection	20 min

5.2. Experimental Evaluation

An index tree is constructed through recursively merging neighboring sub-regions when forwarding messages of the same size in between is minimum in energy consumption. An example is shown in Figure 3, where the skewness degree is set to 60%. Leaf nodes are denoted by the IDs of correspond grid cells, as mentioned in Section 3.2, and R_i (i = 1, 2, ...) represents non-leaf nodes, which are (sub-)regions composed of several neighboring grid cells. For instance, R_5 represents a sub-region whose grid cells are gc_{22} and gc_{23}, while R_2 is composed of R_3 and grid cell gc_{24}. Query answering is performed leveraging this index tree for data gathering, aggregation and routing to the SN.

Our technique is evaluated with respect to four types of queries [33] leveraging the sub-region and attributes of interest, and a query is denoted as q = $<tm, qr, ATR_q>$:

- Single-attribute query in the whole region (SAQWR): q retrieves sensory data of all sensor nodes in the whole network region (*i.e.*, $q.qr = nr$) for a certain attribute (*i.e.*, $|q.ATR|$ = 1). nr specifies

the whole network region. Sensory data of the attribute $q.ATR$ at nr are gathered and routed to the SN at a certain time slot $q.tm$.

- Single-attribute query in a sub-region (SAQSR): The difference between SAQSR and SAQWR lies in the query region. SAQSR retrieves sensory data of sensor nodes in a sub-region of the network (*i.e.*, $q.qr \subsetneqq nr$) with a certain attribute $q.ATR$ (*i.e.*, $|q.ATR| = 1$).
- Multi-attribute query in the whole region (MAQWR): $q.qr$ is the whole network region (*i.e.*, $q.qr = nr$). $q.ATR$ contains some, but not all, attributes. Generally, sensory data of the attributes in $q.ATR$ at nr are gathered and routed to the SN at $q.tm$.
- Multi-attribute query in a sub-region (MAQSR): The difference between MAQWR and MAQSR lies in the query region where MAQSR retrieves sensory data of sensor nodes in a sub-region of the network (*i.e.*, $q.qr \subsetneqq nr$) with multiple attributes.

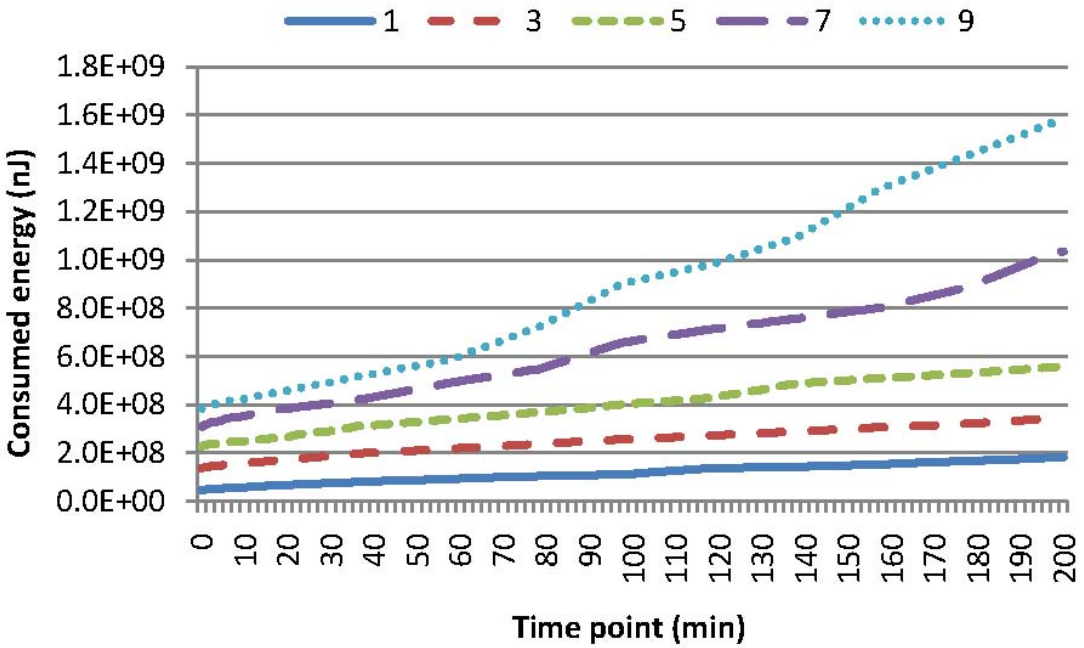

Figure 4. Comparison of the accumulated energy consumption for multi-attribute query in the whole region (MAQWR), where the number of attributes are set to 1, 3, 5, 7 and 9, respectively. The gradient of the curves represents the ratio of energy consumption for query answering. This figure shows that when the cache size in the SN is relatively small and is not capable of caching all sensory data requested by a certain query, more sensory data should be replaced in the SN according to our data replacement mechanism, as presented in Section 4.2, and more energy is required for answering the query. For instance, in comparison with the case when the number of attributes is three, 198% (or 355%) more energy is consumed for the case when the number of attributes is seven (or nine).

Experiments are conducted to evaluate the performance of these four kinds of queries leveraging our two-tier cooperative caching mechanism. Figure 4 compares the energy consumption of MAQWR, where the number of attributes varies as 1, 3, 5, 7 and 9, respectively. The cache size of the SN is set to 600, and the skewness degree is set to 60%. It is worth mentioning that Figure 4 shows the energy consumed in total from scratch, rather than that at a certain time point. The gradient of a curve

corresponds to the energy consumed at a certain time point. The same principle holds for the energy consumption shown in Figures 5–7. Intuitively, more energy is consumed when more attributes are of interest, since more sensor nodes are involved in sensory data gathering and aggregation. The energy consumption for the scenarios, where the number of attributes is 1, 3 or 5, is relatively small and stable. In contrast, the energy consumption for the scenarios, where the number of attributes is seven or nine, increases to a certain extent. Note that sensor nodes involved in the query answering should be 700 (or 900), when the number of interested attributes is seven (or nine). Since the cache size is 600, the sensory data of around 100 (or 300) sensor nodes cannot be cached in the SN. Hence, some sensory data have to be gathered from the network at each time slot, and the sensory data replacement mechanism is always enacted to cache sensory data with the highest popularity. These are the main causes for the increase of energy consumption. For instance, in comparison with the case when the number of attributes is three, 198% (or 355%) more energy is consumed for the case when the number of attributes is seven (or nine). Generally, the larger the cache size of the SN, the less the energy consumption in the network is. Caching in the SN should reduce the energy consumption to a certain extent, especially when the query region is relatively large and the number of interested attributes is relatively big.

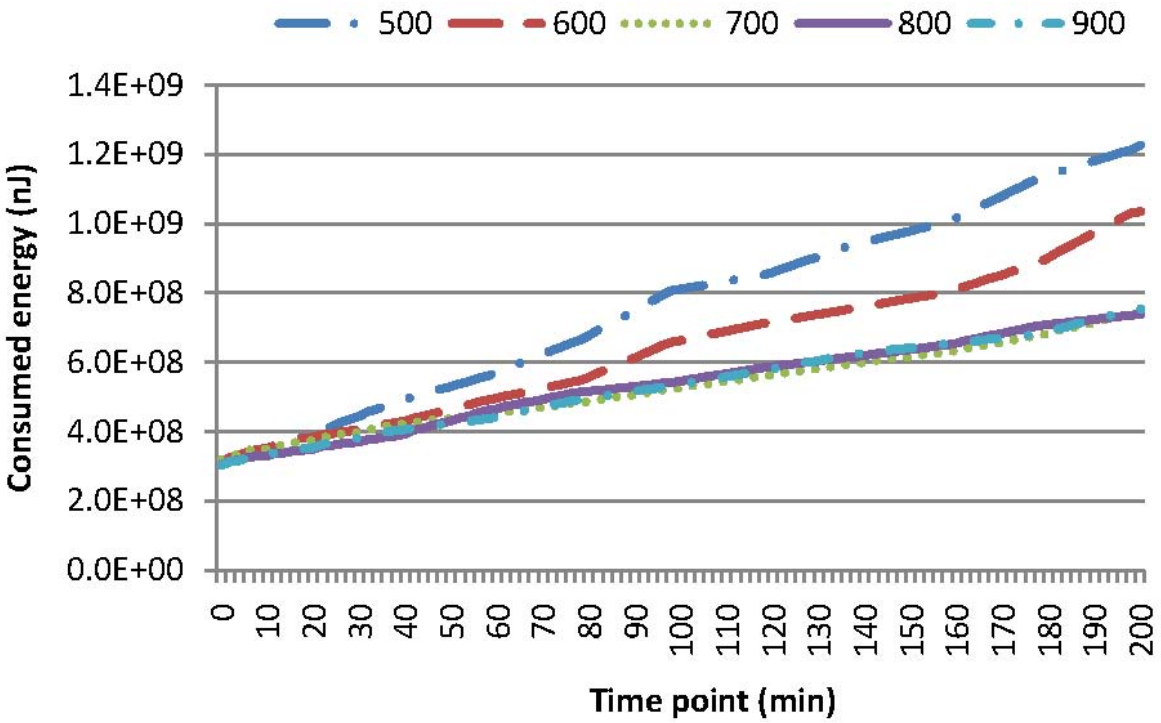

Figure 5. Comparison of the accumulated energy consumption for MAQWR, where the cache size of the SN is set to 500, 600, 700, 800 and 900, respectively. The gradient of the curves represents the ratio of energy consumption for answering the query. This figure shows that when the cache size of the SN is large enough to cache all sensory data requested by the query, the energy consumption is relatively small and steady. Otherwise, the energy consumption is much more and increases significantly. For instance, in comparison with the case when the cache size is set to 800, 66% (or 40%) more energy is consumed for the case when the cache size is set to 500 (or 600).

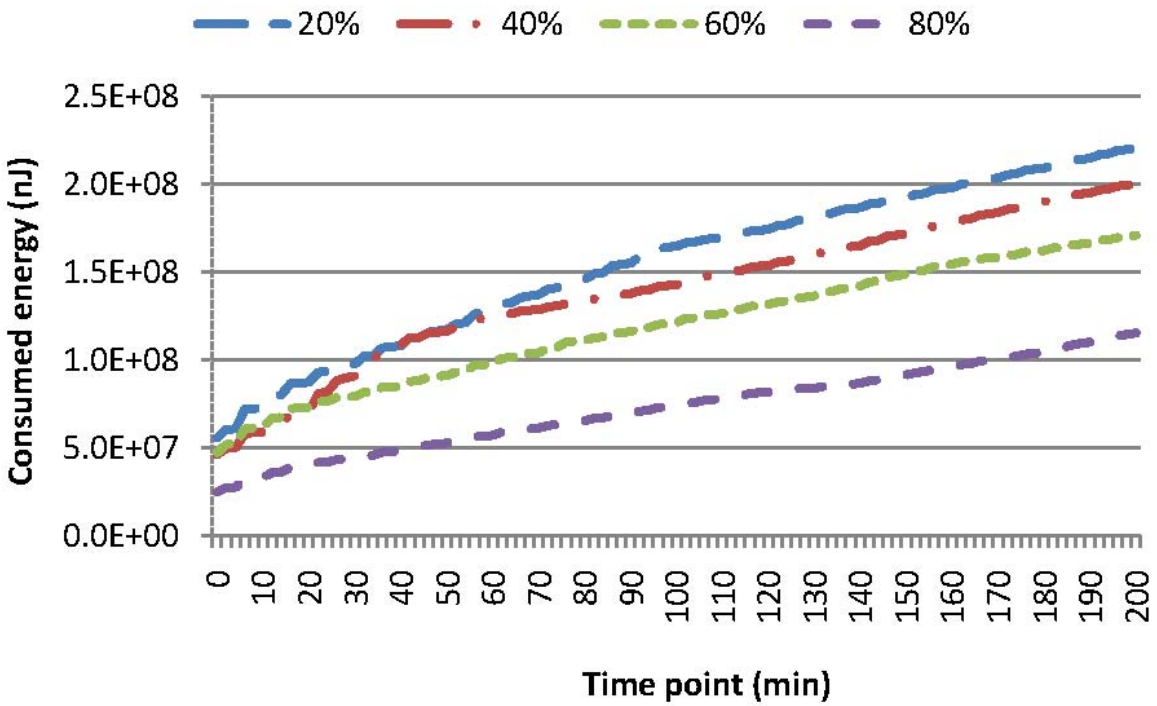

Figure 6. Comparison of the accumulated energy consumption for single-attribute query in the whole region (SAQWR), where the skewness degree for the sensor node distribution in the network is set to 20%, 40%, 60% and 80%, respectively. The gradient of the curves represents the ratio of energy consumption for answering the query. This figure shows that the bigger the skewness degree is, the less energy is consumed for answering the query. For instance, in comparison with the case when the skewness degree is set to 80%, 73% (or 91%) more energy is consumed for the case when the skewness degree is set to 40% (or 20%).

Figure 8 shows cache hit rates hrt_{cah} for MAQWR. hrt_{cah} is calculated as the ratio of $|SD^q_{cah}|$ / $|SD^q|$, where (i) SD^q_{cah} is the set of sensory data cached in the SN that contributes to answering of the query q and (ii) SD^q is the set of sensory data of q inquiries. Without loss of generality, the value of sensory data is assumed to vary according to the formula: val_{vsn} = log (k $\times$ t_{cur} + 1) + C, where: (i) t_{cur} is the current time; and (ii) k and C are constants, which are initially set to random values, and vary according to a normal distribution. Therefore, sensory data of sensor nodes are mostly different and change moderately. Figure 8 shows that cache hit rates for the scenarios, where the number of attributes is 1, 3 or 5, are quite high (roughly 95%), since the SN can have enough storage capability to cache almost all sensory data gathered in recent time slots. As for the scenarios where the number of attributes is seven or nine, cache hit rates are relatively lower (roughly 70%). Similar to the situation for energy consumption, a certain amount of sensory data has to be gathered from the network for query answering in real time. In addition, sensory data, which may be reused for answering the forthcoming queries, have to be removed from the cache by the data replacement mechanism, due to the limitation of the storage capability of the SN. Figure 8 shows that cache hit rates drop every 5 min. Since INs synchronize with sensor nodes in the corresponding grid cells every 5 min, sensory data, which have been changed remarkably, are retrieved from the network. These variations are routed to the SN, but these sensory data are not counted in SD^q_{cah}, which induces the dropping of hrt_{cah} consequently.

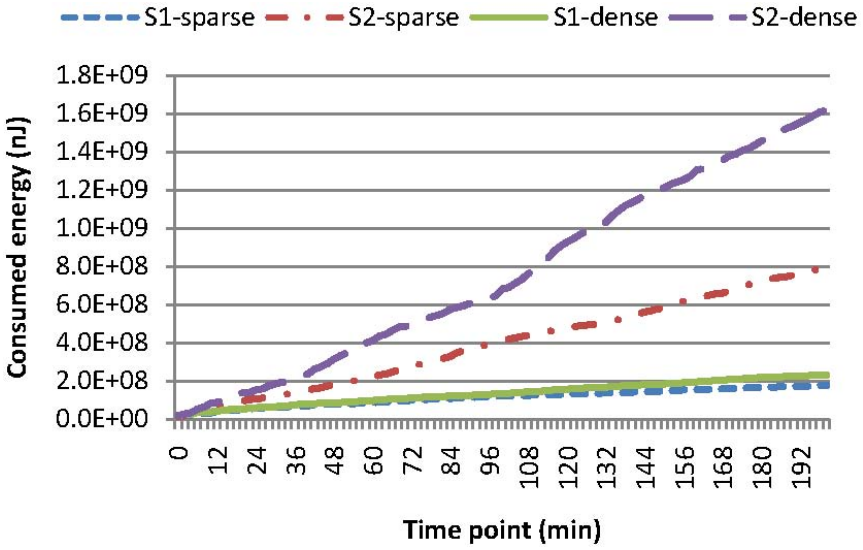

Figure 7. Comparison of the accumulated energy consumption for single-attribute query in a sub-region (SAQSR) with various query configurations, where *S1-sparse* means the sparse sub-regions with our cache mechanism, *S2-sparse* means sparse sub-regions without our cache mechanism, *S1-dense* means the dense sub-regions with our cache mechanism and *S2-dense* means the dense sub-regions without our cache mechanism. The gradient of the curves represents the ratio of energy consumption for answering the query. This figure shows that the energy consumption is decreased dramatically when our cooperative caching mechanism is adopted, especially when the sensors nodes are densely deployed in the network. For instance, 348% (or 599%) more energy is consumed for the case of *S2-sparse* (or *S2-dense*) than the case of *S1-sparse* (or *S1-dense*).

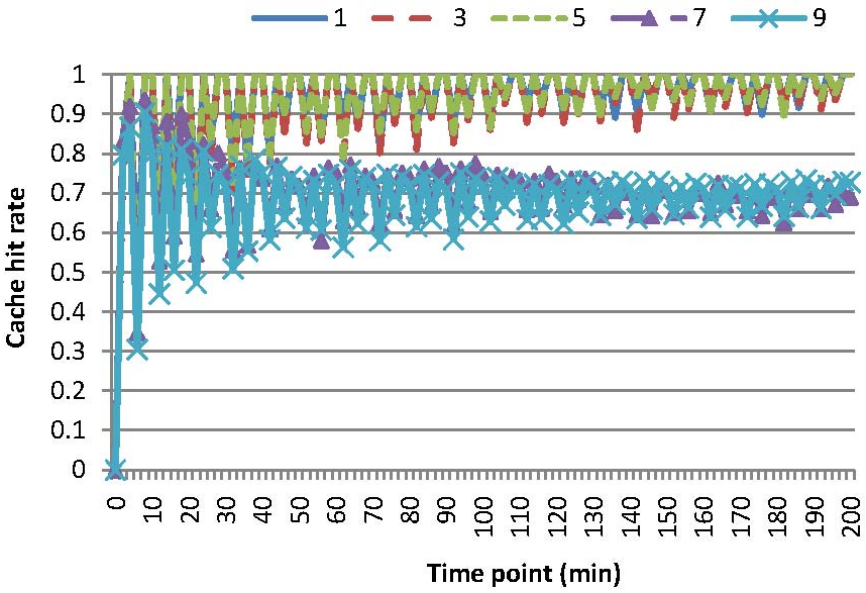

Figure 8. Comparison of cache hit rates for MAQWR, where the number of attributes are set to 1, 3, 5, 7 and 9, respectively. Similar to Figure 4, when the cache size of the SN is relatively small and is not capable of caching all sensory data requested by a certain query, the cache hit rates for sensory data cached in the SN decrease significantly (roughly from 95% down to 70%).

Figures 5 and 9 show the energy consumption and cache hit rates for MAQWR, where: (i) the cache size of the SN is set to 500, 600, 700, 800 and 900, respectively; and (ii) the number of attributes interested in queries is seven. Sensory data of around 700 sensor nodes are able to be cached in the SN. When the cache size is more than 700, much less energy (roughly 40%∼66% of a decrease) is consumed, as shown in Figure 5, and cache hit rates are much higher (roughly 25% of an increase) as shown in Figure 9. As discussed, the sensory data replacement mechanism and real-time data gathering from the network are the main causes of more energy consumption and cache hit rates dropping.

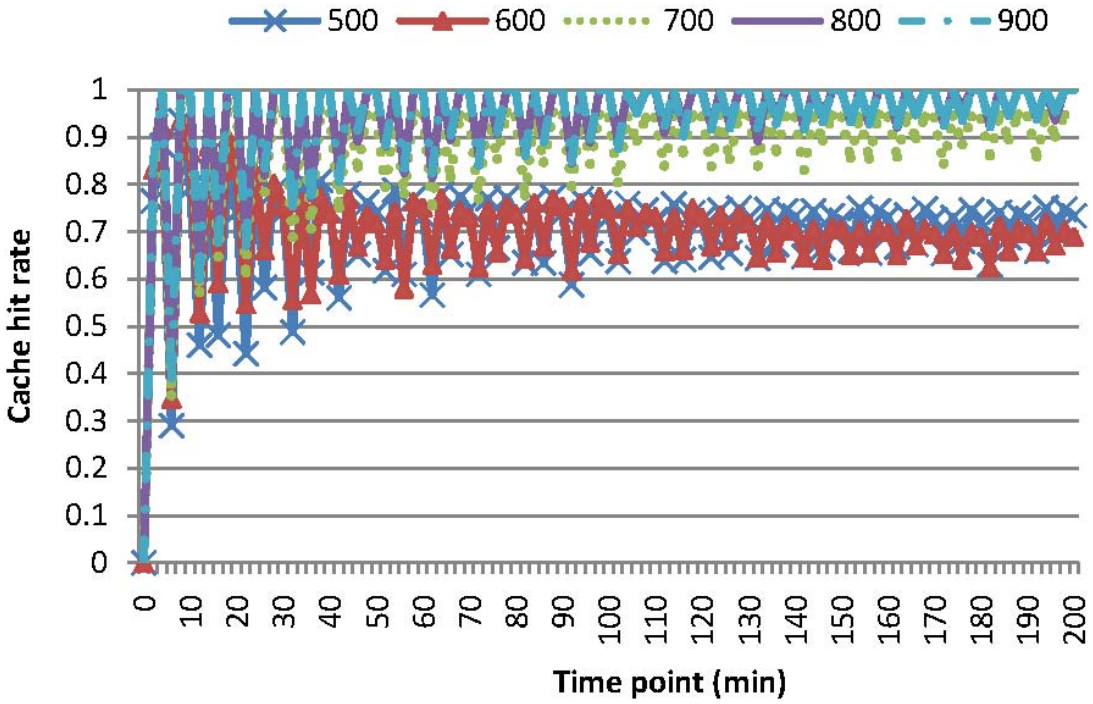

Figure 9. Comparison of cache hit rates for MAQWR, where the cache size of the SN is set to 500, 600, 700, 800 and 900, respectively. Similar to Figure 8, when the cache size of the SN is relatively small and is not capable of caching all sensory data requested by a certain query, the cache hit rates for sensory data cached in the SN decrease significantly (roughly from 95% down to 70%).

Figure 6 shows the energy consumption for SAQWR, where: (i) the skewness degree is set to 20%, 40%, 60% and 80%, respectively; and (ii) the cache size of the SN is set to 600. Note that around 100 sensor nodes are responsible for the query answering, and the cache of the SN can accommodate all of these sensory data. This figure shows that the bigger the skewness degree, the less the energy consumption is for the query answering in the whole network. For instance, in comparison with the case when the skewness degree is set to 80%, 73% (or 91%) more energy is consumed for the case when the skewness degree is set to 40% (or 20%). As presented in Section 3.1, our index tree is constructed through merging two neighboring sub-regions, where the energy consumption of forwarding the same size of messages is the least. Besides, our index tree is a relatively balanced tree, which reduces the path length of dense sub-regions for routing sensory data to the SN. Generally, our technique is more efficient, when sensor nodes are distributed in a skewness fashion.

As mentioned, the points of interest (POI) of domain applications are usually within a certain sub-region for a certain time duration, while evolving to neighboring sub-regions moderately. Queries are often issued periodically and continuously; a query region is typically part of a POI region, and concurrent queries often have overlapping sub-regions. Experiments are conducted for SAQSR, where a skewness degree is set to 60% and the cache size is set to 600. A POI region is assumed to be a rectangle in shape and is set to be dense or sparse sub-regions of the network. A query region is encoded as a set of grid cells contained in the POI region. These grid cells may not be neighboring, for simulating the scenario where multiple queries are issued concurrently. Grid cells at a certain time slot are randomly determined, and the number of grid cells may be different in contiguous time slots. The settings of experiments include: (i) *S1-sparse*: sparse sub-regions with our cache mechanism; (ii) *S2-sparse*: sparse sub-regions without our cache mechanism; (iii) *S1-dense*: dense sub-regions with our cache mechanism; and (iv) *S2-dense*: dense sub-regions without our cache mechanism. Figure 7 shows the energy consumption for these four types of experimental configurations. Intuitively, our cache mechanism can reduce the energy consumption to a large extent, when the query region is within sparse or dense sub-regions, *i.e.*, *S2-sparse* is 348% more in energy consumption than *S1-sparse*, and *S2-dense* is 599% more in energy consumption than *S1-dense*. Note that the difference of energy consumption for *S1-sparse* and *S1-dense* is quite small. This indicates that our cache mechanism is efficient in reducing energy consumption and increasing the network capability, especially when the cache of the SN can accommodate almost all sensory data interested in queries.

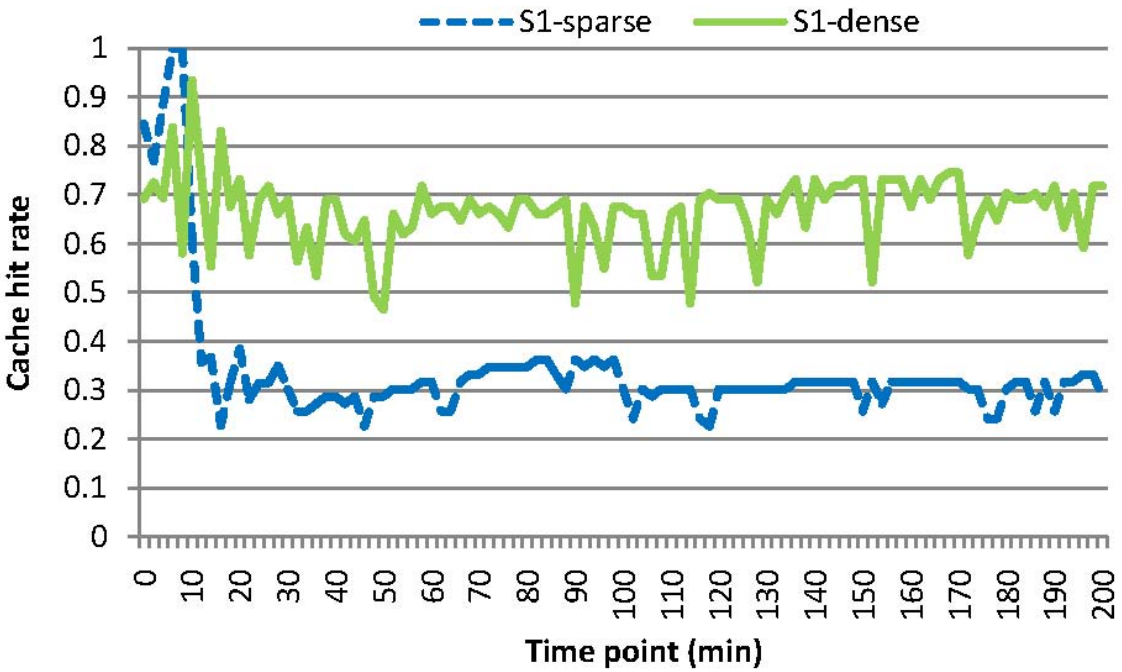

Figure 10. Comparison of cache hit rates for *S1-sparse* and *S1-dense*, where *S1-sparse* means sparse sub-regions with our cache mechanism, and *S1-dense* means dense sub-regions with our cache mechanism. Similar to Figure 6, this figure shows that our cooperative caching mechanism benefits the cache hit rates for cached sensory data of the SN (roughly from 30%–65%), especially when sensors nodes are densely deployed in the network.

Figure 10 shows the cache hit rates for the scenarios *S1-sparse* and *S1-dense*. Generally, the cache hit rate of *S1-sparse* (roughly 30%) is lower than that of *S1-dense* (roughly 65%), although the energy consumption for *S1-sparse* and *S1-dense* is almost the same, as shown in Figure 7. Since grid cells are chosen randomly for representing a query region, common grid cells between continuous queries are relatively less in number than those of Figure 6, which induces a smaller value of the cache hit rates. Note that relatively few sensor nodes are involved in *S1-sparse*, and a minor change of the number of sensor nodes may have a relatively big impact on cache hit rates, which results in a relatively smaller value of cache hit rates for *S1-sparse*. Consequently, our cache mechanism is more efficient, especially when periodic and continuous queries have more overlapping sub-regions.

5.3. Comparison with Relevant Techniques

This section presents the results of our experiments comparing the efficiency and performance of our popularity-based cooperative caching mechanism (denoted PCC) with respect to those of our multiple-attribute query processing (MQP) mechanism, as presented by Zhou *et al.* [33]. Different from PCC, the cooperative caching mechanism has not been adopted in MQP for facilitating the query processing with the same environmental settings.

Experiments have been conducted for the comparison of the energy consumption for PCC and MQP [33] with respect to the query types of MAQWR and SAQWR. The number of attributes are set to 1, 3, 5, 7 and 9, respectively. The cache size of the SN is set to 900, and the skewness degree is set to 60%. Figure 11 shows the experimental results, where the energy consumption at certain time points (e.g., TP2, TP4, *etc*.) is illustrated. Note that the symbol *TPi* (e.g., $i = 2$) means the i-th (second) time point. It is worth mentioning that the energy consumption for MQP is almost the same for all time points, whose values are illustrated at the left side of Figure 11. As for our PCC, the energy consumption is quite large at the first time point (denoted INITin Figure 11), since no sensory data have been cached in the SN for reducing the data gathering from the network in real time, and all intermediate nodes (INs) are required to gather sensory data from the corresponding sensor nodes and to cache them locally. The energy consumption at the succeeding time points decreases to a large extent due to the reusability of sensory data cached in the SN for supporting the forthcoming query answering. This figure shows that the energy to be consumed will be in a steady state after around 18 time slots when the number of attributes is one and around 40 time slots when the number of attributes is nine, due to the fact that sensory data cached in the SN and INs can hardly reduce the energy consumed for query processing any further. Generally, our PCC outperforms MQP on energy consumption, especially when the query attributes are relatively large in number.

Figure 12 illustrates the energy consumption for our PCC and MQP [33] at different time points. As previously mentioned, the energy consumption for MQP is the same for all time points. Figure 12 shows that more energy is consumed at the first time point (denoted INIT). The energy consumption for our PCC is much less than that of MQP in the consequent time points. It is evident from this figure that our PCC is more energy efficient than MQP, especially when the query attributes are relatively large in number.

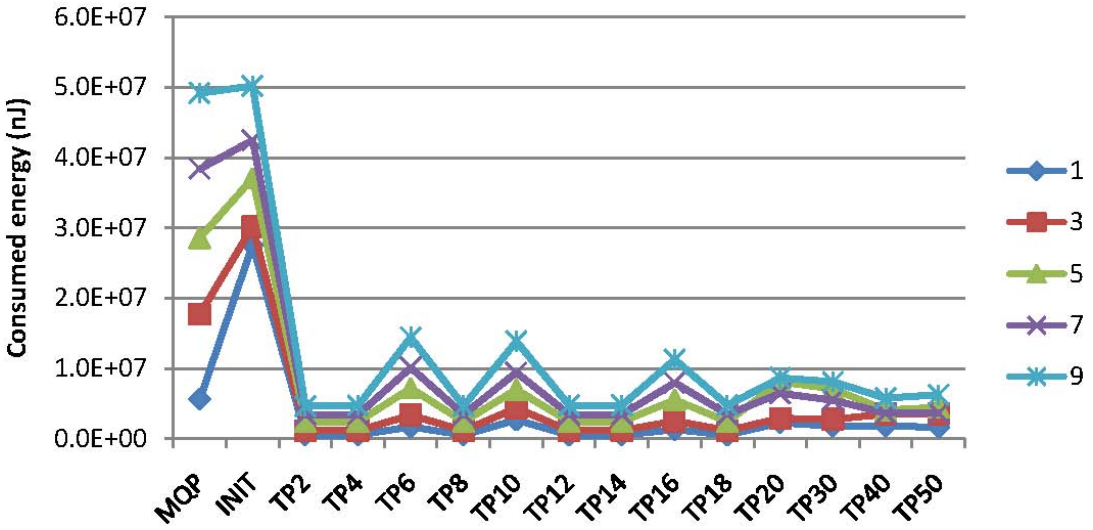

Figure 11. Comparison of energy consumption of our popularity-based cooperative caching (PCC) and multiple-attribute query processing (MQP) [33] for the query types of MAQWR and SAQWR, where the number of attributes is set to 1, 3, 5, 7 and 9, respectively. The energy consumption at various time points (TP2, TP4, *etc*.) are illustrated. Generally, the energy consumption of our PCC is much smaller than that of MQP (roughly 30% less on average for PCC than MQP), especially when the number of query attributes is relatively large. Note that the energy consumption of PCC is quite large at the first time point, since no sensory data have been cached in the SN for reducing the data gathering from the network in real-time to facilitate the query processing.

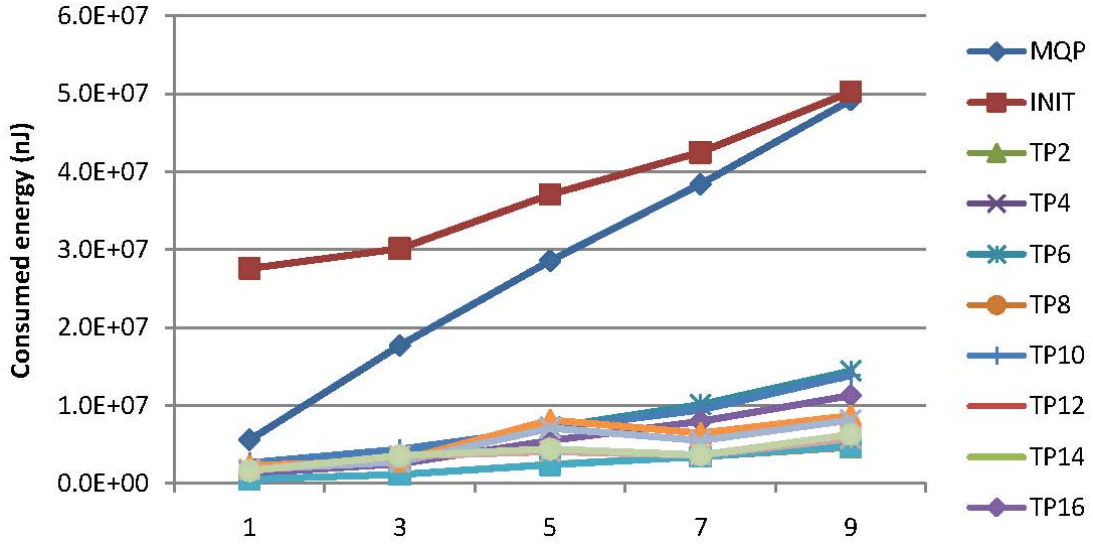

Figure 12. Comparison of energy consumption of our PCC and MQP [33] for the query types of MAQWR and SAQWR, where the number of attributes are set to 1, 3, 5, 7 and 9, respectively. This figure shows that more energy is consumed for our PCC than MQP at the first time point (denoted INIT), while much less energy is consumed afterwards.

6. Related Work and Comparison

Traditional techniques have been developed for facilitating the query processing in wireless sensor networks (WSNs) leveraging the cooperative caching mechanism. We have proposed a popularity-based caching mechanism for optimizing periodic queries in WSNs [42]. One kind of attribute is sensed by sensor nodes in the network. Sensory data are cached only in the memory space of the sink node, and these data are assumed to be valid for answering the forthcoming queries within a certain number of time slots. Usually, the point of interest evolves moderately to neighboring sub-regions, whose sensory data may have become stale already and are not being cached at the sink node at this moment. To facilitate this query processing procedure, grid cells, which may be covered by the forthcoming queries, are derived from the previous queries according to the popularity of interested grid cells. Sensory data are pre-fetched from the network for these grid cells whose popularity is among the highest. Generally, the technique developed in this article is inspired by our previous work [42]. However, multiple kinds of attributes are considered to be sensed by sensor nodes in this technique, and the staleness of sensory data is determined according to whether sensory data have been changed significantly. Besides, this technique removes the assumption made by Zhou *et al.* [42] that the sink node has enough storage capability for caching sensory data of sensor nodes in the whole network. Instead, only sensory data, which have a high possibility of being reused for answering the forthcoming queries for certain attributes, are cached in the sink node. Grid cells, which are not covered by recent queries, cache sensory data of certain attributes or just a flag indicating a dramatic change of sensor nodes. This two-tiered cooperative mechanism, which caches sensory data at the sink node and the head nodes of grid cells, is efficient for facilitating the query answering, as evidenced by the experimental evaluation in Section 5.3.

A cluster-based cooperative caching mechanism is developed by Chauhan *et al.* [47] for supporting query processing. The network is divided into non-overlapping clusters, and each sensor node is assumed to have some cache space. Sensory data are stored in the cache space of sensor nodes that are near the sink node. When a query request is to be responded to, sensory data are retrieved through a cache discovery process. Generally, a sensor node that is responsible for this query is examined for determining whether the required sensory data are saved in its cache space. If not, the cluster of the source sensor node is examined, and the source sensor node is visited for routing the required sensory data to the sink node. This method claims to reduce the requirement for bandwidth, energy and storage of the network. However, the sink node is not responsible for caching sensory data. In fact, sensor nodes near the sink node should cache sensory data and are responsible for routing data to the sink node. These sensor nodes have much more energy consumption and should deplete their energy quickly. In our technique, sensory data are cached in the sink node and the head nodes of grid cells, and the popularity of sensor nodes is considered when determining which sensory data of certain attributes should be cached. A cooperative caching mechanism is developed by Sharma *et al.* [48], where sensory data are cached in the sink node and sensor nodes. A cache zone is formed as a region around a sensor node, where the storage of surrounding sensor nodes can be used to build a larger cumulative cache. A cache discovery mechanism is proposed for identifying required data items, and a cache replacement policy is developed for evicting data items of less importance. This is an interesting work and inspired us to develop our technique. Generally, this work tries to increase data availability nearer the sink and to reduce unnecessary energy

consumption. A query processing mechanism is not discussed specifically when multiple queries are issued periodically and continuously.

To better support the cooperative caching in WSNs, sensor nodes, which can take the role of coordinating and packet caching and forwarding, are essential. A metric of energy betweenness centrality is proposed by Dimokas *et al.* [49] to evaluate the significance of sensor nodes and to examine whether these sensor nodes can take the special role of cooperation concerning the caching decisions. Consequently, a new energy-efficient cooperative caching protocol is developed. An in-network distributed query processor called Corona is developed by Khoury *et al.* [32], which aims to cluster sensor readings into a local storage buffer in sensor nodes. When the freshness of these sensory data is within a certain threshold, they can be used for answering concurrent queries directly, rather than being fetched from the network in real time. Therefore, sensor activation can be minimized. A survey is presented by Kumar *et al.* [30] about cache-based policies in WSNs for reducing the network traffic and bandwidth usage. Besides, cooperative caching has been used in other domains, like mobile *ad hoc* networks, to increase data availability and reduce data access delays [50], and cooperative caching policies have been applied in social wireless networks for minimizing electronic content provisioning cost [51]. Generally, these techniques explore the routing of data packets in the network, and a single query processing is of interest mostly. The strategy of caching sensory data in the intermediate nodes is the main focus; whereas in this article, we propose a two-tiered cooperative caching mechanism for reducing the energy consumption of query answering in the forthcoming time slots.

To establish efficient paths for routing sensory data to the sink node, a cache-based routing metrics is proposed by Grilo *et al.* [29], where intermediate nodes in WSNs are used for caching packets and transmitting them to the sink node. Note that in a heterogeneous Internet of Things, sensor nodes may differ to some extent in terms of storage and computational capabilities. Hence, sensor nodes with larger capabilities are good candidates for caching the packets. Therefore, cache utilization is set as a novel metric applied in this technique, and routing paths should be selected considering cache-rich intermediate nodes. Generally, this work is interesting and inspires caching packets at the intermediate nodes and determining appropriate routing paths; whereas we explore a cooperative caching mechanism at the sink node and the head nodes of grid cells, according to the popularity of sensory data leveraging the recent query history. These two techniques can complement each other for better facilitating the query processing and, thus, improving the energy efficiency. A caching platform is presented by Léone *et al.* [52] for reducing the network communication cost. A gateway based on a constrained application protocol (CoAP)-HTTP proxy is proposed, where cross-layer data are cached in the proxy [53]. When a query is to be answered, the gateway will delegate for query answering. Only when the requested sensory data are not fresh enough or missed in the cache, this query should be transferred to the network for fetching data. This work is similar to what we have developed. However, mainly a caching model is presented, while technical details about the caching strategy are not clear.

Besides, some methods study periodic query processing. In [17,24], the authors study the achievable network capability of snapshot and continuous data collection for a probabilistic WSN model. Cell-based path scheduling and zone-based pipeline scheduling algorithms are proposed for improving the concurrency of snapshot and continuous data collection, respectively. This work inspired us to partition the network region into square grid cells and to use grid cells as elementary units for caching sensory

data. Node localization is an important research problem, especially for large-scale WSNs [54,55]. The network is divided into overlapped local networks, and the corresponding local maps are constructed using a local semidefinite programming method. These local maps are merged into a global map, which contains the exact position of the nodes of interest. It is argued that in certain WSN applications, query requests come periodically with stringent delay constraints [10]. Therefore, a periodic aggregation query scheduling is performed by the designed routing strategy along with packet scheduling protocols. The quality of queries in WSNs is discussed by Brayner *et al.* [56], which aims to deliver a reasonable level of data quality as expected, while ensuring the intelligent consumption of limited network resources. Generally, these methods mainly explore the strategies of supporting (periodic) query processing in WSNs, in order to consume less resources while prolonging the network lifetime. The sharing of sensory data retrieved at a certain time slot and between concurrent queries and the reuse of sensory data gathered by recent queries for answering the forthcoming queries are not discussed extensively.

7. Conclusions

Wireless sensor networks, which act as important interfaces between physical environments and computational systems, have been used extensively to support widespread application domains. Usually, multiple attributes should be sensed in a network, and multiple-attribute sensory data are queried from the network continuously and periodically for facilitating domain applications. Note that sensory data may not change significantly within a certain time duration, and applications may tolerate a variation of adopted sensory data with accurate ones to a certain extent. Consequently, sensory data gathered at this moment can be shared for answering concurrent queries and may be reused for answering the forthcoming queries. To remedy this issue, a two-tier cooperative caching mechanism is proposed in this article. Specifically, the popularity of sensory data, which reflects the possibility of reusing these data by the forthcoming queries, is calculated according to the queries issued in recent time slots. Sensory data of a higher popularity are cached in the sink node, and these data can be used for query answering directly. Sensory data of a lower popularity are cached in the head nodes of grid cells. This two-tier cooperative caching strategy promotes the reuse of sensory data for answering the forthcoming queries significantly. The results of experimental evaluation show that our technique is efficient in the reduction of energy consumption for query answering, especially when the number of queries is relatively large. Specifically, the energy consumption for the case when the sink node is lacking caching space is around 40%∼66% more, as shown in Figure 5, than that for the case when the sink node is capable of caching all sensory data requested by the queries, while the cache hit rate increases significantly, as well (roughly from 70%–95%, as shown in Figures 8 and 9) for these two cases. Compared with our previous technique [33], this two-tier cooperative caching strategy can reduce around 30% of the energy consumption for answering the queries, as shown in Figure 11.

As to the future directions, we are adopting this cooperative caching mechanism to the scenario where a wireless sensor network is shared by multiple applications. The challenge includes the sharing and reusing of query results of these applications for answering forthcoming queries, in order to reduce the energy consumption. Besides, sensory data pre-fetching from the network should be beneficial for the

decrease of energy consumption, especially when the forthcoming queries can be (partially) predicted according to the queries in the past. A prediction model is under the construction.

Acknowledgments

This work was supported partially by the National Natural Science Foundation of China (Grant Nos. 61379126 and 61401107), by the Scientific Research Foundation for Returned Scholars, Ministry of Education of China, by the Guangdong University of Petrochemical Technology's Internal Project (Grant No. 2012RC106), by the Educational Commission of Guangdong Province, China (Grant No. 2013KJCX0131), and by the Fundamental Research Funds for the Central Universities.

Author Contributions

All authors were involved in conceiving the proposed ideas presented in this work. All authors were responsible for writing this manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Lampoltshammer, T.J.; de Freitas, E.P.; Nowotny, T.; Plank, S.; da Costa, J.P.C.L.; Larsson, T.; Heistracher, T. Use of Local Intelligence to Reduce Energy Consumption of Wireless Sensor Nodes in Elderly Health Monitoring Systems. *Sensors* **2014**, *14*, 4932–4947.
2. Zheng, J.; Bhuiyan, M.Z.A.; Liang, S.; Xing, X.; Wang, G. Auction-based adaptive sensor activation algorithm for target tracking in wireless sensor networks. *Future Gener. Comput. Syst.* **2014**, *39*, 88–99.
3. Saukh, O.; Sauter, R.; Marrón, P.J. Time-Bounded and Space-Bounded Sensing in Wireless Sensor Networks. In Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems, Santorini Island, Greece, 11–14 June 2008; pp. 357–371.
4. Renner, C.; Unterschutz, S.; Turau, V.; Romer, K. Perpetual Data Collection with Energy-Harvesting Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *11*, 12.
5. Guo, S.; Wang, C.; Yang, Y. Joint Mobile Data Gathering and Energy Provisioning in Wireless Rechargeable Sensor Networks. *IEEE Trans. Mob. Comput.* **2014**, *13*, 2836–2852.
6. Shi, L.; Han, J.; Han, D.; Ding, X.; Wei, Z. The dynamic routing algorithm for renewable wireless sensor networks with wireless power transfer. *Comput. Netw.* **2014**, *74*, 34–52.
7. Rault, T.; Bouabdallah, A.; Challal, Y. Energy efficiency in wireless sensor networks: A top-down survey. *Comput. Netw.* **2014**, *67*, 104–122.
8. Dai, H.; Chen, G.; Wang, C.; Wang, S.; Wu, X.; Wu, F. Quality of Energy Provisioning for Wireless Power Transfer. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 527–537.
9. Yun, Y.; Xia, Y. Maximizing the Lifetime of Wireless Sensor Networks with Mobile Sink in Delay-Tolerant Applications. *IEEE Trans. Mob. Comput.* **2010**, *9*, 1308–1318.

10. Xu, X.; Li, X.Y.; Wan, P.J.; Tang, S. Efficient Scheduling for Periodic Aggregation Queries in Multihop Sensor Networks. *IEEE/ACM Trans. Netw.* **2012**, *20*, 690–698.
11. Degirmenci, G.; Kharoufeh, J.P.; Prokopyev, O.A. Maximizing the Lifetime of Query-Based Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *10*, 56.
12. Gao, H.; Fang, X.; Li, J.; Li, Y. Data Collection in Multi-Application Sharing Wireless Sensor Networks. *IEEE Trans. Parallel Distrib. Syst.* **2015**, *26*, 403–412.
13. Wang, Y.C. A Two-Phase Dispatch Heuristic to Schedule the Movement of Multi-Attribute Mobile Sensors in a Hybrid Wireless Sensor Network. *IEEE Trans. Mob. Comput.* **2013**, *13*, 709–722.
14. Zhang, Y.; Yang, W.; Han, D.; Kim, Y.I. An Integrated Environment Monitoring System for Underground Coal Mines Wireless Sensor Network Subsystem with Multi-Parameter Monitoring. *Sensors* **2014**, *14*, 13149–13170.
15. Erdelj, M.; Loscrì, V.; Natalizio, E.; Razafindralambo, T. Multiple point of interest discovery and coverage with mobile wireless sensors. *Ad Hoc Netw.* **2013**, *11*, 2288–2300.
16. Pan, M.S.; Liu, P.L.; Lin, Y.P. Event data collection in ZigBee tree-based wireless sensor networks. *Comput. Netw.* **2014**, *73*, 142–153.
17. Ji, S.; Beyah, R.; Cai, Z. Snapshot and Continuous Data Collection in Probabilistic Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2014**, *13*, 626–637.
18. Can, Z.; Demirbas, M. A survey on in-network querying and tracking services for wireless sensor networks. *Ad Hoc Netw.* **2013**, *11*, 596–610.
19. Sarkar, R.; Gao, J. Differential Forms for Target Tracking and Aggregate Queries in Distributed Networks. *IEEE/ACM Trans. Netw.* **2013**, *21*, 1159–1172.
20. Xie, R.; Jia, X. Transmission-Efficient Clustering Method for Wireless Sensor Networks Using Compressive Sensing. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 806–815.
21. Villas, L.A.; Boukerche, A.; de Oliveira, H.A.; de Araujo, R.B.; Loureiro, A.A. A spatial correlation aware algorithm to perform efficient data collection in wireless sensor networks. *Ad Hoc Netw.* **2014**, *12*, 69–85.
22. Zhang, X.; Dong, L.; Peng, H.; Chen, H.; Zhao, S.; Li, C. Collusion-Aware Privacy-Preserving Range Query in Tiered Wireless Sensor Networks. *Sensors* **2014**, *14*, 23905–23932.
23. Umer, M.; Tanin, E.; Kulik, L. Opportunistic sampling-based query processing in wireless sensor networks. *Geoinformatica* **2013**, *17*, 567–597.
24. Ji, S.; He, J.S.; Uluagac, A.S.; Beyah, R.; Li, Y. Cell-Based Snapshot and Continuous Data Collection in Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2013**, *9*, 47.
25. Xu, X.; Li, X.Y.; Song, M. Distributed Scheduling for Real-Time Data Collection in Wireless Sensor Networks. In Proceedings of the IEEE Global Communications Conference, Budapest, Hungary, 25–26 June 2013; pp. 426–431.
26. Li, G.; Guo, L.; Gao, X.; Liao, M. Bloom filter based processing algorithms for the multi-dimensional event query in wireless sensor networks. *J. Netw. Comput. Appl.* **2014**, *37*, 323–333.
27. Kimand, D.; Uma, R.; Abay, B.H.; Wu, W.; Wang, W.; Tokuta, A.O. Minimum Latency Multiple Data MULE Trajectory Planning in Wireless Sensor Networks. *IEEE Trans. Mob. Comput.* **2014**, *13*, 838–851.

28. Hariharan, S.; Bisdikian, C.; Kaplan, L.M.; Pham, T. Efficient Solutions Framework for Optimal Multitask Resource Assignments for Data Fusion in Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *10*, 48.
29. Grilo, A.M.; Heidrich, M. Routing metrics for cache-based reliable transport in wireless sensor networks. *EURASIP J. Wirel. Commun. Netw.* **2013**, *139*.
30. Kumar, H.; Rai, M.K. Caching in Wireless Sensor Networks: A Survey. *Int. J. Eng. Trends Technol.* **2014**, *10*, 549–553.
31. Mahajan, P.C. A New Approach for Scheduling Periodic Aggregation Queries in Wireless Sensor Network with Aggregation Delay. *Int. J. Adv. Res. Comput. Commun. Eng.* **2013**, *2*, 1712–1717.
32. Khoury, R.; Dawborn, T.; Gafurov, B.; Pink, G.; Tse, E.; Tse, Q.; Almi'Ani, K.; Gaber, M.; Rhm, U.; Scholz, B. Corona: Energy-Efficient Multi-query Processing in Wireless Sensor Networks. In Proceedings of the15th International Conference on Database Systems for Advanced Applications, Database Systems for Advanced Applications 15th International Conference, DASFAA 2010, Tsukuba, Japan, 1–4 April 2010; pp. 416–419.
33. Zhou, Z.; Zhao, D.; Shu, L.; Chao, H.C. Efficient Multi-Attribute Query Processing in Heterogeneous Wireless Sensor Networks. *J. Internet Technol.* **2014**, *15*, 699–712.
34. Heinzelman, W.R.; Chandrakasan, A.; Balakrishnan, H. Energy-Efficient Communication Protocol for Wireless Microsensor Networks. In Proceedings of the 33rd Annual Hawaii International Conference on System Sciences, Maui, HI ,USA, 4–7 January 2000; pp. 1–10.
35. Lachowski, R.; Pellenz, M.E.; Penna, M.C.; Jamhour, E.; Souza, R.D. An Efficient Distributed Algorithm for Constructing Spanning Trees in Wireless Sensor Networks. *Sensors* **2015**, *15*, 769–791.
36. Shan, M.; Chen, G.; Luo, D.; Zhu, X.; Wu, X. Building Maximum Lifetime Shortest Path Data Aggregation Trees in Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *11*, 11.
37. Zobel, J.; Moffat, A.; Ramamohanarao, K. Inverted files versus signature files for text indexing. *ACM Trans. Datab. Syst.* **1998**, *23*, 453–490.
38. Marx, D.; Razgon, I. Constant Ratio Fixed-Parameter Approximation of the Edge Multicut Problem. In Proceedings of the 17th Annual European Symposium on Algorithms, Copenhagen, Denmark, 7–9 September 2009; pp. 647–658.
39. Heinzelman, W.B.; Chandrakasan, A.P.; Balakrishnan, H. An Application-Specific Protocol Architecture for Wireless Microsensor Networks. *IEEE Trans. Wirel. Commun.* **2002**, *1*, 660–670.
40. Forster, A.; Forster, A.; L.Murphy, A. Optimal Cluster Sizes for Wireless Sensor Networks: An Experimental Analysis. In Proceedings of the 1st International Conference on Ad Hoc Networks, Niagara Falls, ON, Canada, 16–18 October 2010; pp. 49–63.
41. Zhu, J.; Lung, C.H.; Srivastava, V. A hybrid clustering technique using quantitative and qualitative data for wireless sensor networks. *Ad Hoc Netw.* **2015**, *25*, 38–53.
42. Zhou, Z.; Zhao, D.; Xu, X.; Du, C.; Sun, H. Periodic Query Optimization Leveraging Popularity-Based Caching in Wireless Sensor Networks for Industrial IoT Applications. *Mob. Netw. Appl.* **2014**, doi:10.1007/s11036-014-0545-4.
43. Wu, M.; Tan, L.; Xiong, N. A Structure Fidelity Approach for Big Data Collection in Wireless Sensor Networks. *Sensors* **2015**, *15*, 248–273.

44. Zhu, C.; Leung, V.C.M.; Yang, L.T.; Shu, L. Collaborative Location-based Sleep Scheduling for Wireless Sensor Networks Integrated with Mobile Cloud Computing. *IEEE Trans. Comput.* **2014**, doi:10.1109/TC.2014.2349524.
45. Bagchi, A.; Pinotti, C.; Galhotra, S.; Mangla, T. Optimal Radius for Connectivity in Duty-Cycled Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *11*, 36.
46. Carrano, R.C.; Passos, D.; Magalhaes, L.C.S.; Albuquerque, C.V.N. Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 181–194.
47. Chauhan, N.; Awasthi, L.K.; Chand, N. Cluster Based Efficient Caching Technique for Wireless Sensor Networks. In Proceedings of the International Conference on Latest Computational Technologies, Dubrovnik, Croatia, 4–7 September 2012; pp. 85–89.
48. Sharma, T.; Joshi, R.; Misra, M. Dual Radio Based Cooperative Caching for Wireless Sensor Networks. In Proceedings of the16th IEEE International Conference on Networks, Orlando, FL, USA, 19–22 October 2008; pp. 1–7.
49. Dimokas, N.; Katsaros, D. Detecting Energy-Efficient Central Nodes for Cooperative Caching in Wireless Sensor Networks. In Proceedings of the IEEE 27th International Conference on Advanced Information Networking and Applications, Barcelona, Spain, 25–28 March 2013; pp. 484–491.
50. Abbani, N.; Artail, H. Protecting data flow anonymity in mobile ad hoc networks that employ cooperative caching. *Ad Hoc Netw.* **2015**, *26*, 69–87.
51. Taghizadeh, M.; Micinski, K.; Ofria, C.; Torng, E.; Biswas, S. Distributed Cooperative Caching in Social Wireless Networks. *IEEE Trans. Mob. Comput.* **2013**, *12*, 1037–1053.
52. Leone, R.; Medagliani, P.; Leguay, J. Optimizing QoS in Wireless Sensors Networks using a Caching Platform. In Proceedings of the 2nd International Conference on Sensor Networks, Barcelona, Spain, 19–21 February 2013; pp. 23–32.
53. Ludovici, A.; Calveras, A. A Proxy Design to Leverage the Interconnection of CoAP Wireless Sensor Networks with Web Applications. *Sensors* **2015**, *15*, 1217–1244.
54. Li, S.; Wang, X.; Zhao, S.; Wang, J.; Li, L. Local Semidefinite Programming-Based Node Localization System for Wireless Sensor Network Applications. *IEEE Syst. J.* **2014**, *8*, 879–888.
55. Naraghi-Pour, M.; Rojas, G.C. A Novel Algorithm for Distributed Localization in Wireless Sensor Networks. *ACM Trans. Sens. Netw.* **2014**, *11*, Article No. 1.
56. Brayner, A.; Coelho, A.L.; Marinho, K.; Holanda, R.; Castro, W. On query processing in wireless sensor networks using classes of quality of queries. *Inf. Fusion* **2014**, *15*, 44–55.