MDS: Efficient Multi-dimensional Query Processing in Data-Centric WSNs

Hanhua Chen†, Mo Li‡, Hai Jin†, Yunhao Liu‡, Lionel M. Ni‡

†*School of Computer Science and Technology*
Huazhong Univ. of Science and Technology
{chenhanhua,hjin}@hust.edu.cn

‡*Dept. of Computer Science and Engineering*
Hong Kong Univ. of Science and Technology
{limo,liu,ni}@cse.ust.hk

Geographical hash table (*GHT*) *has been widely used to provide energy efficiency for data-centric storage in wireless sensor networks. Such a mechanism, however, suffers from high communication cost when we apply multi-dimensional event search in the network. In this work, we present MDS, a flexible, complete, and efficient multi-dimensional search mechanism atop traditional GHT based data-centric storage architecture. MDS utilizes bloom filters to reduce the communication cost of in-network intersection and union operations for multi-dimensional queries in wireless sensor networks. This scheme can be easily extended to support multi-dimensional range queries. Our mathematical analysis indicates the optimal settings for the bloom filters that maximize the traffic savings according to the information popularities. We conduct comprehensive simulations to evaluate our design. Results show that MDS achieves significant performance improvement in terms of energy consumptions and thus improves the applicability of the multi-dimensional search over the GHT based data-centric storage in sensor networks.*

1. Introduction

The emerging wireless sensor networks (WSNs) [4, 8, 13, 15, 21, 26] have been revolutionizing the ways of collecting information from the physical world. The community has envisioned a large variety of applications, such as environment monitoring, underground surveillance, and so on.

Although sensor networks provide us unprecedented access to the detailed observations of the physical world, directly delivering large amount of sensory data back to the base station (sink) might quickly exhaust the energy of the sensor nodes [20]. Recently proposed data-centric storage (DCS) schemes [22, 23] explore alternative ways that organize the sensory data and store them within the network. External queries are scheduled in a DCS system to retrieve corresponding data for query processing. For instance, a surveillance WSN monitoring animal migration often collects many individual animal sightings. All the sighting events can be organized and stored in some given places within the network instead of directly delivering them back to the base station. The future queries can thus be forwarded to corresponding places and yield the time and location of all such sightings [23].

The key issue affecting the efficiency of the DCS scheme is how to organize the storage of the sensory data at appropriate rendezvous nodes for the consumers to retrieve. Most existing schemes employ the Geographical Hash Table (GHT) [20] to disseminate different sensory data in the network, where event data are hashed by pre-defined event types to different geographical locations. A consumer node is able to apply corresponding hash functions to retrieve data from rendezvous nodes that store different types of events. The DCS scheme has been proven efficient for storing and processing information in large scale WSNs, especially for information search applications [14, 22]. For the queries concerning the events of a specific type of data, which we call singular events, DCS scheme provides much higher retrieval efficiency than previous data-centric routing schemes [12, 16], where data are stored at their origins and the discovery of the desired information relies on flooding the network.

Recent advances in sensor hardware designs have enabled sensor nodes the capability of detecting multiple types of environment elements [1]. It is expected that more functionalities can be added to sensors, fulfilling the needs of monitoring complex environments involving multiple types of sensory data. Information queried by users thus can become highly selective - they might be interested in different subsets of environment attributes. Under such conditions, desired events are likely being flexibly composed of multiple different data types, which we call multiplex event. Such search process is also referred to as a multi-dimensional search [14].

There are two types of methods to design the store

1052-8725/08 $25.00 © 2008 IEEE
DOI 10.1109/RTSS.2008.26

and query of multiplex events: 1) the first one is that we define a complex event as a whole. When a node detects these complex events it stores them to some storage nodes determined by specific algorithms. In the querying phase, queries are simply forwarded to the corresponding storage nodes to retrieve results; 2) the second method is that we still define the multiple data types in a multiplex event as separate singular events, and then the multiplex events are composed only in the querying phase.

The first method is efficient when the patterns of the event are known before deployment and never change. It has limitation, however, on the flexibility when users want to know a different undefined event. Typically schemes include DIM [14] and Double Ruling [22]. The second type of method aims at achieving non-predefined multiplex events retrieval. For example, in our sea environment surveillance applications [2, 25], scientists who analyze the growth of marine micro-organisms might be casually interested in the multiplex events that occurred within certain temperature and light conditions, e.g., "Find me all events that have temperatures above 20 celsius degree and light level above 15". A naïve scheme is to utilize GHT infrastructure to disseminate the two singular events to different GHT nodes across the network. During querying phase, each multiplex event is decomposed into two singular events, "events with temperature above 20" and "events with light level above 15". Thus, to obtain the search result for such multiplex events, the two single-data-type queries are processed separately with a consequent intersection operation at the base station. Although only two rendezvous nodes need to be involved during the search process, each sends a large amount of data through the network, introducing excessively heavy traffic overhead.

To address the above issue, we propose a novel protocol, MDS, atop traditional GHT based DCS schemes, for flexible, complete, and efficient multi-dimensional search. MDS utilizes Bloom Filter (BF) [6, 18, 24] to encode the data set to transmit and achieves efficient distributed intersection and union operations in WSNs. By transmitting the BF instead of the raw data among the rendezvous nodes together with reverse verification, MDS achieves exact intersection and union on multi-dimensional data in the network, largely reducing the traffic overhead.

Through mathematical analysis, we demonstrate that the optimal BF setting for a query q is determined by the popularity of items relevant to the data types involved in q. We accordingly derive an effective approach to achieve optimal settings for BF through numerical analysis. To further reduce the communication cost, we adopt caching strategy to avoid sending redundant BFs across the network. By using model-driven data acquisition schemes [10], we further extend MDS to handle multi-dimensional range queries.

We conduct comprehensive simulations to evaluate the efficiency of MDS. The results show that, compared with the baseline approaches, MDS achieves significant performance improvement in terms of energy consumption and thus largely improves the applicability of multi-dimensional search over existing DCS scheme.

The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 introduces the design of MDS. Section 4 shows how MDS can support a range query. We analyze the optimal settings of BF for MDS multi-dimensional search and evaluate the performance of this design in Section 5. We conclude the work in Section 6.

2. Related Work

There are mainly two different types of retrieval schemes in wireless sensor networks: data-centric routing [12, 16] and data-centric storage [14, 22, 23].

In data-centric routing, data are stored at their origins while the discovery of the desired information usually relies on flooding the network, requiring communication cost of $O(n)$ for a query, where n is the number of sensor nodes in the network. Clearly, that kind of approaches mainly target at infrequent queries for streaming data type where the cost of flooding can be amortized by the following long-term data delivery that followed [12]. However, for queries from multiple consumers for the same data source, the performance deteriorates as data sources might be rediscovered separately by multiple consumers. DCS-based retrieval schemes map events to proper rendezvous nodes for the base station to retrieve [14, 22, 23]. A query only needs to visit the rendezvous location to acquire data of a given type at the cost of packet transmissions $O\left(\sqrt{n}\right)$. DCS-based schemes greatly reduce the communication cost on event search by avoiding flooding. Using hash mapping, such schemes, however, have the limitation of exact-match, providing poor search capability.

To support complex queries for multiplex events, existing DCS-based schemes commonly use an attribute vector to define a multiplex event. For example, DIM [14] leverages a locality-preserving geographic hash to map events with comparable attributes values to places nearby. DIM embeds a *k-d* tree [5] like index in a sensor network, with which the bounding rectangle that contains all sensors within the network are

recursively divided into two halves until each zone contain only one sensor node. Each node acts as an index node of the zone. The zone and multi-dimensional events are coded based on the same *k-d* tree scheme. Thus, any event with the pre-defined fix attribute vector can be hashed to a zone for the base station to retrieve. With *k-d* tree index, events with comparable attributes values are stored nearby and DIM can support multi-dimensional range queries. However, DIM only supports the fixed attribute vector. The attributes of interested events are predetermined in order to construct the DCS structure. In practice, it is likely and perhaps often the case that a user is only interested in a selected set of the attributes instead of all in the pre-defined event type.

So far, there is not much research on addressing the flexible multi-dimensional search. One possible scheme is to organize a distributed global index which maps each data type to a set of events based on DCS mechanisms, such as GHT. Then, a multi-dimensional query is searched by retrieving the sets for each data type through the global index, and then performing an intersection/union operation. In such a scheme, although only a few sensor nodes need to be contacted, each sends a potentially large amount of data across the entire network. Our MDS approach largely improves the efficiency of the in-network intersection/union operations.

3. System Design

In this section, we first give a brief overview of the design of MDS. We then describe our approach of reducing the communication cost using bloom filters. We illustrate the optimization strategies for both "and" and "or" queries separately in Sections 3.1 and 3.2. In Section 3.3 we introduce the caching strategy to further reduce communication cost by avoiding transmitting redundant BFs among GHT nodes.

3.1. System model

Our design is based on the following search model. The base station searches for the multiplex events involving one or more data types and the system should send back the set of events containing all the requested types of data. Our system model is based on recent works on DCS systems [22, 23]. We assume the presence of a large number of data types. According to the GHT principle, the singular events of each data type are mapped into a random location by predefined hash functions. Sensors at different locations are responsible for accommodating the singular events composed of the data types mapped to them. While our approach is general to any of the concrete GHT techniques [20, 22], for simplicity, our following discussion assumes the architecture closely related to the basic GHT design [20]. We also assume that the sensors are densely deployed. Each sensor in the network detects different types of environmental data. Corresponding singular events of each data type are inserted into the global GHT index. The underlying geographic routing algorithms support the data dissemination and query dissemination.

A multi-dimensional query search based on such a GHT index includes looking up the sets for the singular events of different data types from multiple GHT nodes and returning the multiplex events by intersection/union on the accommodated events. In order to reduce the communication cost to achieve in-network intersection/union operations, each concerned rendezvous node sends a BF, a succinct data structure of a set, instead of the set itself to perform the intersection/union operation. The base station adjusts the BF parameters into optimal settings to achieve minimized communication cost.

3.2. MDS protocol

3.2.1. Bloom filter. Before we introduce our approach that reduces the communication cost, we briefly review the basis of bloom filters [6]. A BF is a bit vector *bitvec_m* with *m* bits, initially all set to 0. The BF facilitates membership test to a finite set $S = \{x_1, x_2, \ldots, x_n\}$ of *n* elements from a universe *U*. It uses a set of *k* uniform and independent hash functions $\{h_1, h_2, \ldots, h_k\}$ to map the universe *U* to the bit address space [1-*m*]. For each element *x* belonging to *S*, the bits $h_i(x)$ are set to 1 for $1 \le i \le k$. To check whether or not an item *y* is in *S*, we check whether all $h_i(y)$ are set to 1. If not, *y* clearly is not a member of *S*. If all $h_i(y)$ are set to 1, *y* is in *S* with high probability which can be controlled by the parameters of BF.

After all *n* elements of *S* are hashed and inserted into the BF, the probability that a specific bit of *bitvec_m* is still 0 is

$$p = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m} \quad (1)$$

The probability of a false positive after *n* elements inserted in the *bitvec_m* is the probability that a new element is not in *S,* but can be separately hashed by the *k* hash functions to some *k* bits of "1" in the *bitvec_m*.

$$f = (1 - p)^k = (1 - e^{-kn/m})^k \quad (2)$$

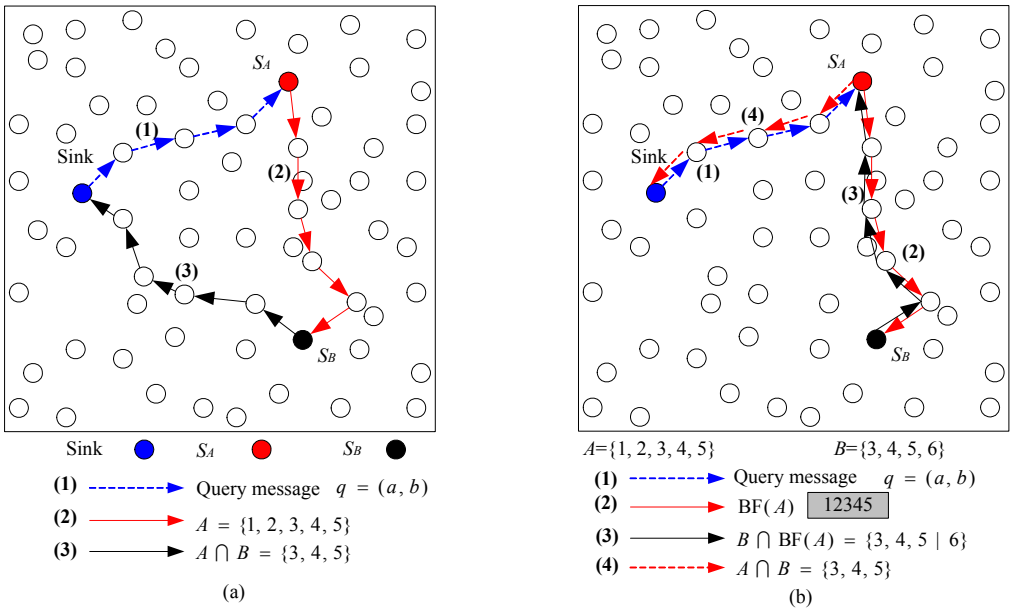

Figure 1. Bloom Filter based in-network intersection

Given an optimal choice of k hash functions, the false positive rate f can be minimized when $m = \frac{k \cdot n}{\ln 2}$ and the lower bound of the positive rate [18] is

$$f_{\min} = 0.6185^{\frac{m}{n}} \quad (3)$$

3.2.2. In-network intersection. A common requirement for multi-dimensional search in WSNs is to conduct a distributed intersection operation inside the network. Figure 1.(a) gives an example of a two-data-type query q = (a and b). The query is first routed to the GHT node S_A which is responsible to accommodate the singular events of data type a. Then A, the set of selected event items of data type a, is transmitted to the GHT node S_B which is responsible to accommodate the singular events of data type b. Node S_B is thus able to obtain $A \cap B$, where B is the set of selected event items of data type b. The final results are returned back to the base station.

Although in the above in-network intersection operation, only two nodes are involved, each of them sends a large amount of data across the WSN and the communication cost is expensive. For the same example discussed above, MDS reduces the communication cost by sending a BF that contains the element information of A, BF(A), instead of the raw set of A itself to node S_B. As illustrated in Fig.1.(b), the gray box represents BF(A), the Bloom filter of set A. Our approach allows $A \cap B$ to be determined with much fewer bits transmitted compared with S_A directly sending the entire set A. When BF(A) is transmitted to node S_B, it determines the intersection of A and B by checking each item in B according to the records in BF(A). Because the BF has no false negatives, the result set contains all the elements of in the true intersection set. Due to the false positives, however, the result set might contain elements that are not in A. Typically, a client wishes to retrieve only the exact intersection result of A and B. Thus, the result set, denoted by $B \cap$BF(A), is sent back to node S_A for verification. S_A removes the false positives from $B \cap$BF(A) by calculating $A \cap (B \cap$BF(A)), which is equivalent to $A \cap B$.

Table 1. Notations in MDS Algorithm

Notation	Description
A	The set of the events containing a
BF(A)	The bloom filter for set A
$B \cap$BF(A)	The estimated intersection of A and B based on BF(A) and B
n	Number of elements inserted into a BF
m	Size of the bit vector used as a BF
k	Number of hash functions used for a BF
f	False positive rate of a BF
f_{min}	Minimized false positive rate of a BF
S_A	The GHT nodes responsible for a
r	The number of bits each item takes

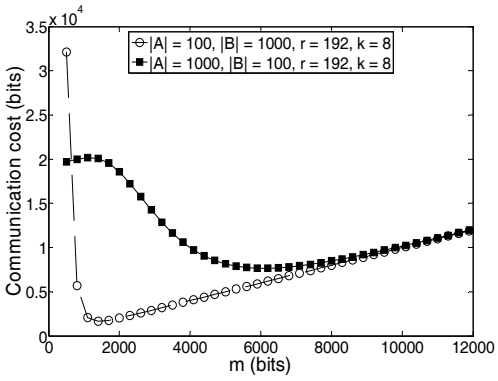

Figure 2. Asymmetry of communication cost of in-network intersection

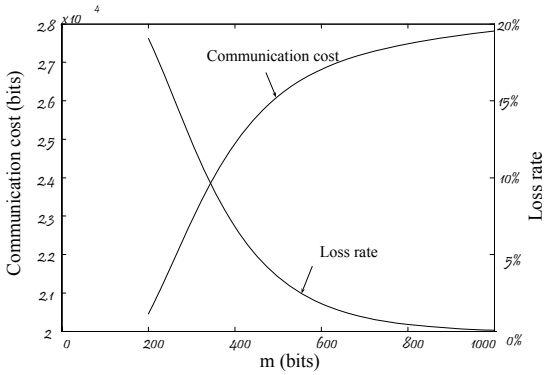

Figure 3. Trade-off between the loss rate and communication cost

In the following we will analyze the communication cost of our algorithm. We will further discuss how our algorithm can be extended to support range queries in Section 4 and how to achieve optimal BF settings in our algorithm in Section 5. Before we introduce the optimization scheme, we summarize the notations used in our algorithm in Table 1.

The communication cost of the BF based intersection is quantified as

$$m + | A \bigcap B | r + f | B | r \qquad (4)$$

We assume that each element in the set takes r bits. To evaluate the communication cost of BF-based distributed intersection, the cost of $|A \cap B|$ can be ignored since it represents the final intersection result, which anyhow will be sent back to the base station regardless of the choice of algorithms. We substitute f according to Eq.(2), and the extra communication cost for distributed intersection is given by

$$f(m,k) = m + (1 - e^{-k|A|/m})^k \mid B \mid r \qquad (5)$$

Given $|A|$ and $|B|$, the minimal value of $f(m,k)$ can be achieved by adjusting the parameters m and k of BF.

We find that the minimal communication cost is not symmetric when sets A and B differ in their sizes and thus the order of the intersection operation is critical in minimizing the communication cost. In Fig. 2, we observe that when $|A|$ =100, $|B|$ = 1000, and k = 8, the minimal extra communication cost is 1,647 bits with an optimal setting m = 1410, while when $|A|$ = 1000 and $|B|$ = 100, the minimal extra communication cost is 7,568 bits with an optimal setting m = 6,082. In such a case, we can achieve a 4.6× performance improvement in terms of communication cost if we send the BF of the event set of less popular data type to the GHT node accommodating the event set of more popular data type. In Section 5, we show how we can achieve BF optimal settings under different cases for a real system through further analysis.

In our algorithm, it is also possible to send $B \cap \mathrm{BF}(A)$ directly to the base station rather than first sending it back to node S_A. This further reduces more communication cost but suffers from a slight loss in the result precision due to the false positive of BF. Given reasonable values of $|A|$, $|B|$, k, and m, the number of the extra-transmitted elements is in proportion to the cardinality of set B. $B \cap \mathrm{BF}(A)$ contains $(1 - e^{-k|A|/m})^k \mid B \mid$ extra elements that do not belong to set A. The precision of the final result will be slightly decreased to

$$\frac{|A \bigcap B|}{|A \bigcap B| + (1 - e^{-k|A|/m})^k |B|} \times 100\% \qquad (6)$$

3.2.3. In-network union. In some applications, we may need to process the "or" queries, which demand the results containing all singular events of the appointed data type in the query. Fig. 4.(a) presents an example of the traditional strategy for processing a two-dimensional "or" query. First, two copies of the query are separately sent to the GHT nodes S_A and S_B,. S_A and S_B respectively send back their complete event sets. At last the search results of both data types are merged at the client. The total communication cost is $(|A|+|B|)\ r$. In our design, as shown in Fig. 4.(b), the query is first routed to node S_A, which sends BF(A) to node S_B, then S_B picks up the items that are not in A by checking each item in B against BF(A). Only the items picked up, denoted as B-BF(A), are returned to the client for a consequent union operation.

The communication cost of the in-network union operation for data type A and B can be quantified by

$$m + (| A | + (1 - f)(| B | - | A \bigcap B |))r \qquad (7)$$

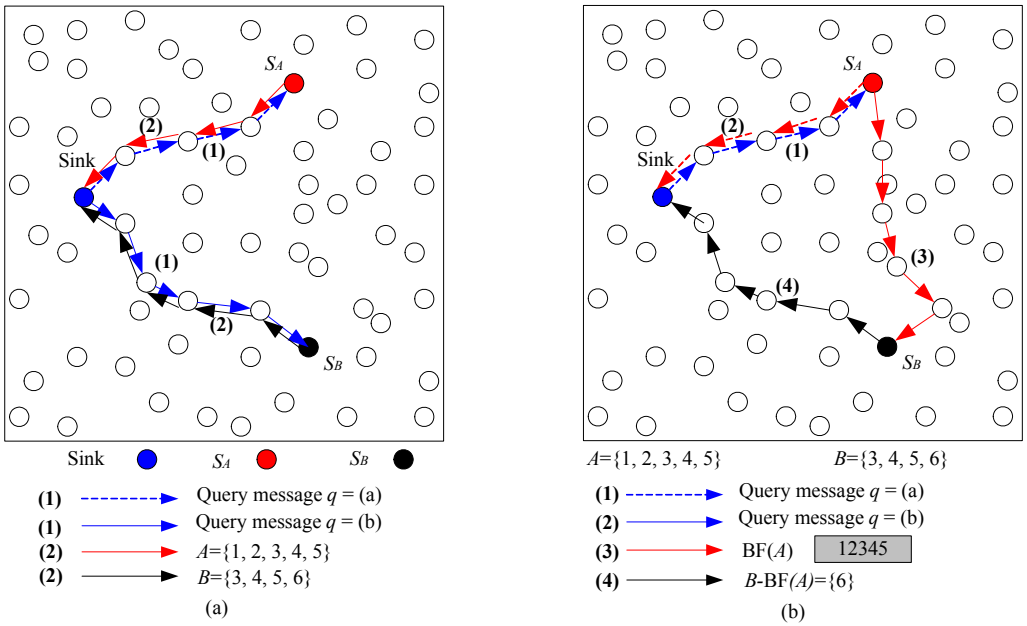

Figure 4. Bloom Filter based in-network union

By avoiding repeatedly sending the intersection part of sets A and B, our algorithm for the in-network union is designed efficient in traffic cost. Note that B-BF(A) is slightly different from B-A due to the false positive in the BF: some elements in $A \cup B$ are missed in the final results. Given $|A|$ and $|B|$, the loss rate can be quantified as follows.

$$\frac{f \times (|B| - |A \cap B|)}{|A| + |B|} \times 100\% \qquad (8)$$

Figure 3 plots the theoretically calculated curves that exhibit the trade-off between the loss rate and the communication cost, where $|A|$ =100, $|B|$ = 100, $|A \cap B|$= 60, k = 8, and r =192. It shows that when m is increased, the communication cost is increased while the loss rate is decreased. For example, the communication cost can be reduced from 38,400 bits to 23,500 bits at a loss rate of 9.5%.

The possible reduced communication cost by BF can be quantified as M_{saved}= ($|A|$+$|B|$) r- (m+$|A|$ r+(1-f) $|B$-$A|$ r) =(1-f) $|A \cap B|$ r+ $f|B|$ r-m, where the size of intersection $A \cap B$ can be estimated with the BF based algorithm first proposed in [19].

As described in Section 3.2.2, during the "and" query search, GHT nodes exchange BFs for their data types. Thus the intersection size can be calculated.

MDS piggybacks the intersection size to the base station for future use. In MDS design we use a threshold for strategy selection. If $\frac{M_{saved}}{(|A|+|B|)r} > \delta$, where δ is a predefined threshold, we use BF for distributed union operation, otherwise we use the straightforward strategy.

When we choose algorithms in the design for a real system, we may consider this trade-off between the search quality and system resource consumption. For example, we can minimize the false positive of BF to achieve the best recall rate. We substitute f in Eq. (6) according to Eq. (3), and the communication cost for distributed union is,

$$m + |A| r + (1 - 0.6185^{\frac{m}{|A|}})(|B| - |A \cap B|)r \qquad (9)$$

Where $m = \frac{k \cdot |A|}{\ln 2}$.

In MDS in-network union algorithm we do not consider a complete search mechanism using reverse verification like the in-network intersection algorithm, that achieves 100% recall but consumes even more communicational resources than the straightforward strategy that simply transmits all the sets directly to the base station.

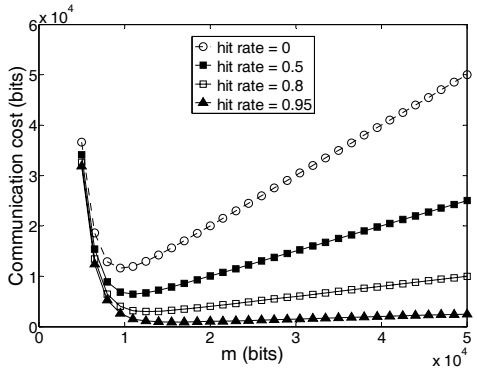

Figure 5. Cache for in-network intersection

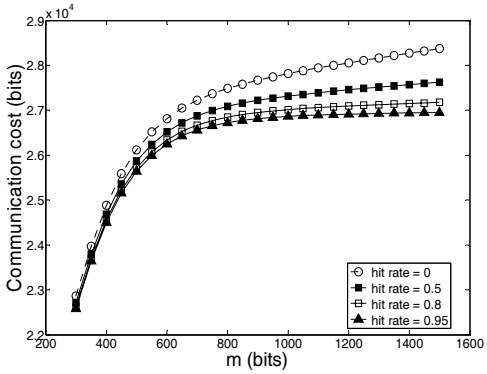

Figure 6. Cache for in-network union

3.3. Caches

GHT nodes may cache the posting list sent by BF to avoid transmitting them again for future queries. For example, in Fig. 1.(b), when node S_B already has BF(A) in its cache, a search operation for the data types a and b may skip the first step, in which node S_A sends its BF(A) to node S_B. We can obtain more benefit from caching BFs compared with caching the entire data element match lists because the small size of the BF representation facilitates a cache of fixed size to store more data types.

Assume a BF is in another node's cache with probability p. The communication cost formula for the in-network intersection in Eq. (5) can be altered, considering cache hit rate, p, as follows:

$$(1-p)\,m + (1-e^{-k|A|/m})^k \mid B \mid r \qquad (10)$$

The communication cost formula for the in-network union in Eq. (7) can be altered as follows:

$$(1-p)m + (\mid A \mid + (1-(1-e^{-k|A|/m})^k)(\mid B \mid - \mid A \cap B \mid))r \qquad (11)$$

Figure 5 illustrates the effect of cache hit rates on the communication cost for the in-network intersection operation in Eq. (10) where $|A|$ = 1000, $|B|$ = 1000. It shows that the minimized communication cost decreases when the cache hit rate increases. For example, when the hit rate, p, is 0.5, the minimum excess number of bits sent is 6,476 bits, representing 29.64:1 compression when compared with sending A or B, which is a 1.8× improvement on the performance without cache strategy. When p is increased to 0.8, the minimum communication cost is further reduced to 2,977 bits, which is a 3.9× improvement. Figure 6 shows a similar effect of the cache scheme for the in-network union operation.

Intuitively, the more frequently two data types are related in the search process, the more communication cost can be saved by caching the transferred BF. MDS can adaptively adjust the cache rate by learning from the query experiences. Specifically, the base station can learn the correlation among data types from the query logs and compute the statistical query rate. MDS piggybacks the statistical information on queries, and sends them to involved GHT nodes to adjust their cache rates. Designing an optimal cache strategy is out of the scope of this paper, as it is highly related to the concrete application running atop the system and many efforts have been done for cache strategy design under different application systems [11]. Note that the replicas of a BF can become slightly stale due to the update of local event records on corresponding rendezvous nodes. Although slight staleness on the BF information is acceptable, we handle the cache consistency with a TIMEOUT field in the cached BF. After a period of time, the BF for an updated set will be transmitted.

4. Supporting Range Queries

This section illustrates how MDS can also effectively support range queries. Using a range query, a user may prefer events with data type values in certain ranges rather than simply of certain properties, e.g., "finding out all events that have temperature between 40˚F and 50˚F and light level between 10 and 20". As the analysis will be shown in Section 5, the optimal settings of a BF are determined by the popularities of data types. For a range query, we should know how many items are in each specified ranges. Since the ranges for each data type may vary frequently in the queries issued by users, caching such numbers in the base station is not applicable here.

To solve this problem we adopt the model-driven method [10] to estimate such numbers. Specifically, the GHT nodes can derive the *probability density function* (PDF) *p(x)* for each data type *x* using standard algorithms (e.g., [17]). The PDF for data type *x* together with the popularity information $|X|$ is then cached in the base station. Thus the base station can compute the probability of a specified range $[t_1, t_2]$ by:

$$p(x \in [t_1, t_2]) = \int_{t_1}^{t_2} p(x)dx \quad (12)$$

The popularity of *x* in range $[t_1, t_2]$ can be estimated by

$$| X | \int_{t_1}^{t_2} p(x)dx \quad (13)$$

Using the PDF and the size of the event item list, the base station can estimate the number of items within any range. It is a cost-efficient way for a base station to get the optimal settings of a BF.

5. Performance Evaluation

In this section we first introduce our simulation setup and then we analyze how to achieve the optimial settings of a BF. Then we verify the efficiency of our algorithm. We compare the performance of MDS with that of the baseline algorithm which transmits the raw sets.

5.1. Simulation setup

In the simulation, we put 1296 nodes on the grid of a $600m \times 600m$ rectangle and then perturb each point by a random shift which has been widely treated as an approximation for the manual deployment of sensor nodes [7]. We use the unit disk graph model for sensor communications. The communication radius of each node is set to $30m$. Each node generates different singular events of three types of data. We assume that the data popularities within the sensor network follow a Zipf distribution with parameters $\alpha = 1.0$ and $n = 5000$. Each type of data is hashed to a random rendezvous node by the name of the data type. Upon an event is detected, it is published to the GHT by hashing the data type it belongs to. According to the TinyOS standard [3, 9], every message in our design has a limited length of 46 bytes with 28 bytes payload, 11 bytes header information and 7 bytes metadata.

We compute the energy consumption for transmitting data M_i by: $E_i = \frac{|M_i|}{L} H_i$, where $|M_i|$ is the size of the transmitted data, L is the length of the payload in each packet and H_i is the hops the data transmitted from the source to the destination.

During the searching process, we randomly select a node as the base station to issue queries containing at least three types of data.

5.2. Optimal settings of bloom filter

In this section, we show how we achieve the optimal parameter settings of BF. We analyze the communication cost quantified by Eq. (5) with Matlab tools.

We examine three situations (1) $|A|<|B|$, (2) $|A|>|B|$, and (3) $|A| = |B|$. In all the three cases, we find that the value of $f(m, k)$ is significantly influenced by the variable *m*, the length of BF, while it is slightly influenced by the value of *k*, the number of hash functions used in BF.

When $|A| \le |B|$, the minimal communication cost can be achieved when *m* is set as an optimal value. Based on the observation, given $|A|$, $|B|$ and *k*, the objective of our optimal in-network intersection algorithm is to choose an optimal *m* and the intersection order to achieve the minimal communication cost.

Because the minimal communication cost is not symmetric due to different sizes of *A* and *B*. Thus, in our design, we first sort the number of singular events of each data type in a query in an increasing order according to their popularities, $|A| \le |B|$.

By varying the values of $|A|$ and $|B|$, we get a set of sample values for an optimal *m.* We find that with the same values of $|B|/|A|$, the value of $m/|A|$ is a constant, where *m* is the optimal setting. For simplicity, we use *u* to denote $|B|/|A|$ and *v* to denote $m/|A|$. Thus, we can derive a function $v = f(u)$.

We used the least-squares polynomial curve-fitting tool in Matlab to find the best fits. Figure 7 shows the curves for the fits. The three cubed curve $v = 0.0004u^3 - 0.0193u^2 + 0.5493u + 10.0652$ is the best to fit the distribution of the optimal *m*. Note that such a function is quite important for a sensor node to configure the BF with optimal settings because it incurs no extra configuration cost. Thus the base station can calculate the optimal *m* by: $m = f(u)\,|A|$, with the popularity information $|A|$ and $|B|$.

The popularity information of data types *a* and *b* can be easily achieved though looking up GHT. By caching the popularity information in the base station, MDS avoids frequently retrieving such statistical information. In order to update the popularity information, MDS piggybacks the information in the search results to facilitate future queries.

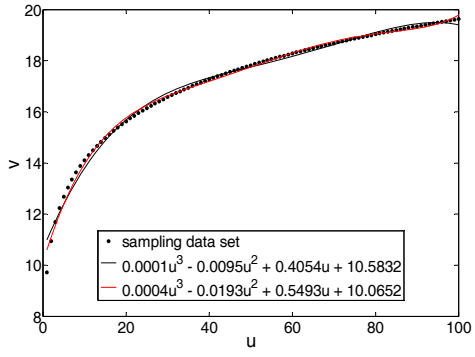

Figure 7. Curve-fitting for optimal settings

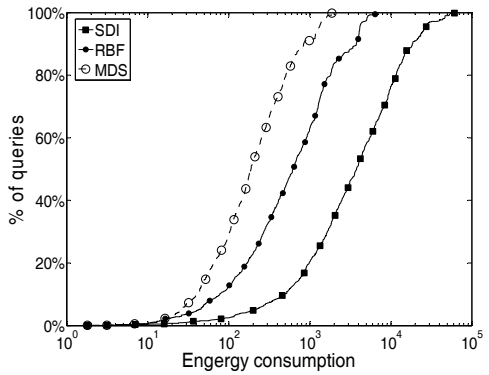

Figure 8. Energy cost for in-network intersection

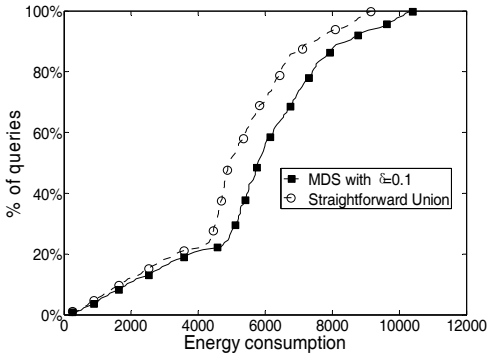

Figure 9. Energy cost for in-network union

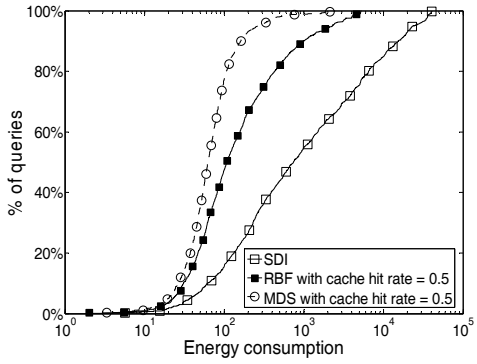

Figure 10. Improvement with cache strategy

5.3. Results

To evaluate the performance of our MDS mechanism, we compare the performance achieved by our BF based algorithm with the distributed intersection and union algorithms.

5.3.1. In-network intersection. Figure 8 plots energy consumptions of all the tested queries. In this experiment the straightforward distributed intersection (SDI) algorithm is used as a baseline approach. We examine how BF based algorithm can reduce the energy consumptions. We mainly consider two strategies, the BF based algorithm with random intersection order (RBF) and the BF based distributed intersection algorithm with optimal BF settings and intersection order according to the popularity information (MDS).

The results show that about 80.5% queries using MDS algorithm have energy consumptions less than 500, while only 9.8% queries of the baseline achieve such low energy consumption. About 44.1% queries using RBF algorithm have energy consumption less than 500. Such a result validates that our insight about the optimal BF settings based on popularity is quite effective.

We compare the values of average energy consumption per query in different algorithms. The results show that the energy consumption per query of SDI is 7,409, while the value of MDS is 337. This shows that MDS achieves a 22× performance improvement.

5.3.2. In-network union. In this experiment, we evaluate the performance of our BF based distributed union algorithm described in Section 3.2.3. We use the straightforward union algorithm as the baseline.

Figures 9 shows the energy consumption of distributed union algorithm in MDS, where the base station decides the search strategy with the threshold fixed at $\delta = 0.1$. The results show that about 48.9% involved queries using MDS have energy consumptions less than 5,000, while only 24.3% queries of the baseline achieves such energy consumption. Statistically the energy consumption per query is reduced by 12.52%.

5.3.3. Caches. We combine the strategy of BF and

cache together and examine the total performance improvement. Figure 10 plots energy consumptions of all the queries when the hit rate is set to 0.5. The results show that about 93.5% MDS queries have energy consumptions less than 200, while only 27.6% SDI queries and 67.2% RBF queries achieve such low energy consumptions. According to the results, the complete MDS mechanism achieves a 45× performance improvement.

6. Conclusions

We propose MDS, a multi-dimensional search infrastructure for WSNs. By utilizing bloom filters, MDS reduces the communication cost for distributed intersection/union operations during the multi-dimensional search process. We present the optimal settings for a BF through mathematical analysis, and a cache strategy is designed to further reduce the communication overhead. Our simulation results show that MDS achieves significant performance improvement in terms of energy consumptions and makes the multi-dimensional search more applicable for existing GHT schemes.

7. Acknowledgements

This work was supported in part by National 973 Basic Research Program of China under grant No.2006CB303000 and Hong Kong RGC grant HKUST 6169/07E.

8. References

[1] Crossbow, http://www.xbow.com/, 2008.

[2] OceanSense, http://www.cse.ust.hk/~liu/Ocean/, 2008.

[3] TinyOS, http://www.tinyos.net/, 2008.

[4] X. Bai, D. Xuan, Z. Yun, T.-H. Lai, and W. Jia, "Complete optimal deployment patterns for full-coverage and k-connectivity (k<=6) wireless sensor networks," in Proceedings of ACM MobiHoc, 2008.

[5] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[6] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Communication of the ACM*, vol. 13, no. 7, pp. 422-426, 1970.

[7] J. Bruck, J. Gao, and A. A. Jiang, "MAP: Medial Axis Based Geometric Routing in Sensor Network," in Proceedings of ACM MobiCom, 2005.

[8] M. Cardei, S. Yang, and J. Wu, "Algorithms for Fault-Tolerant Topology in Heterogeneous Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 4, pp. 545-558, 2008.

[9] N. Cooprider, W. Archer, E. Eide, D. Gay, and J. Regehr, "Efficient Memory Safety for TinyOS," in Proceedings of ACM SenSys, 2007.

[10] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-Driven Data Acquistion in Sensor Networks," in Proceedings of VLDB, 2004.

[11] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 282-293, 2000.

[12] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," in Proceedings of ACM MobiCom, 2000.

[13] A. Koubaa, M. Alves, and E. Tovar, "Modeling and Worst-Case Dimensioning of Cluster-Tree Wireless Sensor Networks," in Proceedings of IEEE RTSS, 2006.

[14] X. Li, Y. J. Kim, R. Govindan, and W. Hong, "Multidimensional Range Queries in Sensor Networks," in Proceedings of ACM SenSys, 2003.

[15] M. Liu, J. Cao, Y. Zheng, and L. Xie, "A Energy-Efficient Protocol for Data Gathering and Aggregation in Wireless Sensor Networks," *Journal of Supercomputing*, vol. 43, pp. 107-125, 2008.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks," in Proceedings of OSDI, 2002.

[17] T. Mitchell, M*achine Learning:* McGraw Hill, 1997.

[18] M. Mitzenmacher, "Compressed Bloom Filters," *IEEE/ACM Transactions on Networking,* vol. 10, no. 5, pp. 604-612, 2002.

[19] J. K. Mullin, "Estimating the Size of a Relational Join," *Information Systems,* vol. 18, no. 3, pp. 189-196, 1993.

[20] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A Geographic Hash Table for Data Centric Storage," in Proceedings of WSNA, 2002.

[21] S. Ren, Q. Li, H. Wang, X. Chen, and X. Zhang, "Design and Analysis of Sensing Scheduling Algorithms under Partial Coverage for Object Detection in Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems,* vol. 18, no. 3, pp. 334-350, 2007.

[22] R. Sarkar, X. Zhu, and J. Gao, "Double Rulings for Information Brokerage in Sensor Networks," in Proceedings of ACM MobiCom, 2006.

[23] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin, "Data-Centric Storage in Sensornets," in Proceedings of ACM HotNets, 2002.

[24] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, "Fast Hash Table Lookup Using Extended Bloom Filter: An Aid to Network Processing," in Proceedings of ACM SIGCOMM, 2005.

[25] Z. Yang, M. Li, and Y. Liu, "Sea Depth Measurement with Restricted Floating Sensors," in Proceedings of IEEE RTSS, 2007.

[26] H. Zhai and Y. Fang, "Impact of Routing Metrics on Path Capacity in Multirate and Multihop Wireless Ad Hoc Networks," in Proceedings of ICNP, 2006.