Hint-based Routing in WSNs using Scope Decay Bloom Filters

Xiuqi Li and Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
{xli,jie}@cse.fau.edu

Jun (Jim) Xu
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
jx@cc.gatech.edu

***Abstract*— In existing query-based routing protocols in wireless sensor networks (WSNs), a node either keeps precise route information to desired events, such as in event flooding, or does not keep any route to desired events such as in query flooding. In this paper, we propose a routing protocol, called *Hint-based Routing by Scope Decay Bloom Filter (HR-SDBF)*, that employs probabilistic hints. In HR-SDBF, each node maintains some probabilistic hints about events and utilizes these hints to route queries intelligently. We also put forward a data structure, *Scope Decay Bloom Filter (SDBF)* to encode the probabilistic hints. With SDBF, the amount of information about an event is propagated, without any loss, within the k-hop neighborhood of an event source but decreases outside the k-hop neighborhood as the distance from the event source increases. Compared to existing query-based protocols, HR-SDBF greatly reduces the amortized network traffic without compromising the query success rate and achieves a higher energy efficiency. To the best of our knowledge, this is the first query routing protocol in WSNs that utilizes probabilistic hints encoded in a variant of the bloom filter. Both the analytic and the experimental results support the performance improvement of our protocol.**

Keywords: bloom filter, data-centric, hint-based, query-based, routing, wireless sensor networks (WSNs).

I. Introduction

Wireless sensor networks (WSNs) have been used in applications such as the health industry, military, warehouse, and home environment [1]. Sensors are typically low-cost, low power, and multi-functional. They communicate with each other through wireless media and form a wireless distributed network.

In WSNs, routing is data-centric, i.e. finding data with specific attribute values [2]. In many WSN applications, routing is query-based. A *sink* initiates a query for some desired data, which is forwarded towards the hosting sensors [3]. Sinks can be static or dynamic. In this paper, we are interested in the latter case, where any sensor could issue a query. We refer to the queried data as events. Because sensors have limited power, one of the major challenges in designing WSN routing protocols is energy efficiency. One way to achieve this is to reduce the total routing traffic [4].

Existing query-based routing protocols can be classified into two types. The first type, called *query flooding based*, is blind forwarding and does not proactively maintain any hints. Queries are flooded over the WSNs. Query flooding can find desired events quickly but is also costly because many query messages are generated. This is evident when many events are frequently queried. The second type, called *event flooding based*, employs precise routing hints to route queries. The second type can reduce query messages at the expense of heavy routing overhead. Specifically, keeping precise routing hints for many events is expensive. This is because each node keeps a precise list of events that may be found through each neighbor. The cost to create and update this list is prohibitive when the list is large.

This work was supported in part by NSF grants ANI 0073736, CCR 0329741, CNS 0422762, CNS 0434533, EIA 0130806, and CNS 0531410.

In this paper, we propose a routing protocol, called *Hint-based Routing by Scope Decay Bloom Filter (HR-SDBF)*, that utilizes probabilistic hints. Each sensor maintains probabilistic hints about events that may be found through its neighbors. Hints are encoded using the proposed variant of the bloom filter (BF) [5] [6], called *scope decay bloom filter (SDBF)*.

A BF is a lossy compression of a set for supporting membership queries. It consists of a bit string and a group of hash functions. To generate a BF for a set, each set element is mapped by each hash function to a bit position in the bit string. All mapped bits are set. To determine the membership of an item, the item is hashed similarly. If any of the hashed bits is not set, then the item definitely does not belong to the set. If all bits are set, then the item is *possibly* in the set. If in fact the set does not contain the item, a false positive occurs. Nevertheless, the space savings usually offset this shortcoming when the false positive rate is significantly low. Bloom filters have been used in database applications [5], web caching [7], and searching in peer-to-peer networks [8] [9]. Unlike BFs, a SDBF can denote different amount of information about an element and represent probabilistic membership.

The HR-SDBF protocol uses SDBFs to advertise the routing hint about an event. The advertisement is designed such that the hint does not decay within the k-hop neighborhood of an event source but decays outside the k-hop neighborhood as the distance from the boundary of the k-hop neighborhood increases. By trading off precise routing hints for probabilistic ones, HR-SDBF achieves a higher query success rate with the same or less amortized routing overhead.

Sinks may conduct different types of searching based on

SDBFs. They can specify the minimum amount of information that a neighbor must have in order to receive queries.

- *1-thread best HR-SDBF*. A query is always forwarded to the best neighbor that has the maximum amount of information among all neighbors. Ties can be broken by random selection or based on some SDBF component. This option is intended to find at least one desired event with minimum cost.
- *N-threads HR-SDBF*. A query is forwarded to all neighbors that have the full amount of information about the desired event (i.e. all bits for the desired event are set). This choice is designed to find all events within the no-decay scope, k-neighborhood.

We make the following contributions in this paper.

- We propose a hint based routing protocol, HR-SDBF, which combines the advantages of both blind and precise-hint based schemes. To the best of our knowledge, HR-SDBF is the first query routing protocol in WSNs that utilizes probabilistic hints.
- We present a novel data strucure, *scope decay bloom filter (SDBF)*. SDBFs improve the conventional BFs by being capable of representing probabilistic membership and various amount of information about elements. SDBFs are flexible and can include different decay models.
- We discuss different design tradeoffs in HR-SDBF. These include tie breaking by random selection and some SDBF component, and different decay models such as the exponential decay and the linear decay.
- We conduct extensive performance analysis and simulation of HR-SDBF.

This paper is organized as follows. In Section II, we review the related work. In Section III, we give an overview of the HR-SDBF protocol. In Section IV, we present the detailed design of HR-SDBF. In Section V, we provide an analytical study of the HR-SDBF. In Section VI, we present experimental results. At the end, we summarize the paper and point out the future work.

II. Related work

A. Query flooding and its variants

The simplest way to route queries is to flood queries from the sink over the entire WSN and set up the reverse paths for desired data to be sent back to the sink. Various query flooding schemes differ in the manner in which they set up and use reverse paths. Directed diffusion [10] tries to find an optimal path between the sink and the event sources by flooding an exploratory query that is initiated at a sink. Each node sets gradients between neighboring nodes, and reinforces the best route for real data while transferring the exploratory events on the reverse query path. The gradients are only used for sending the real data from the discovered event source to the sink that initiates the exploratory query.

Gradient-based routing [11] is another scheme based on query flooding. It associates each node with a height, which is the minimum distance in terms of the number of hops from the sink. The scheme also assigns a gradient to the link between a node and its neighbor. A gradient is defined as the height difference between a node and its neighbor. A node always forwards desired data through the link with the highest gradient among all links to its neighbors. Energy aware routing [12] is also based on query flooding. This scheme tries to maintain multiple paths between a data source and the sink. Desired data is propagated through a route that is probabilistically selected. The probability of a route is set based on its energy consumption.

To reduce the cost of query flooding, gossiping [13] can be used for query-based routing in WSNs. It is essentially a random walk where each node forwards a received query to a randomly chosen neighbor.

B. Event flooding and its variants

Another option to correct the deficiency of query flooding is event flooding when the number of events in the WSN is small. We can use the minimum cost forwarding algorithm in [14] to set up the minimum cost path from every node to the event source. The event source broadcasts an event with cost 0. Each node updates its cost estimate and forwards a received message if the message leads to a lower cost path. Rumor routing [15] combines query flooding and event flooding. It uses long-lived packets (called agents) to spread event hints. An agent randomly walks in the WSN and updates its event routes along its path. An event query is forwarded to a neighbor that knows a route to the desired event. If no hint is available, a query is randomly forwarded.

Another related work is SQR searching [8] in peer-to-peer networks that uses an exponential decay bloom filter (EDBF) to advertise hints. SDBF is more general than EDBF. It can incorporate different decay models and can be utilized to perform more types of routing.

To the best of our knowledge, only one existing query-based WSN routing protocol, called resilient data-centric storage [16], uses bloom filters for routing. It stores information about all event sources of an event type in some replica nodes. A bloom filter is used to represent the collection of attribute value pairs for all event types at each replica node. This protocol differs from our approach because bloom filters are not used for offering probabilistic hints.

III. HR-SDBF Protocol - Overview

This section outlines the overall design of the HR-SDBF routing protocol. The design details will be discussed in the next section. The basic idea in the HR-SDBF protocol is to route the query based on probabilistic hints about the desired events. To obtain these probabilistic hints, we spread the knowledge about an event from the event source such that the amount of information about the event does not decay within the k-hop neighborhood but decreases outside the k-hop neighborhood as the distance increases.

HR-SDBF protocol includes two types of searching, *1-thread best HR-SDBF* and *N-thread HR-SDBF*. The first approach is pictured in Figure 1(a). A query is always forwarded

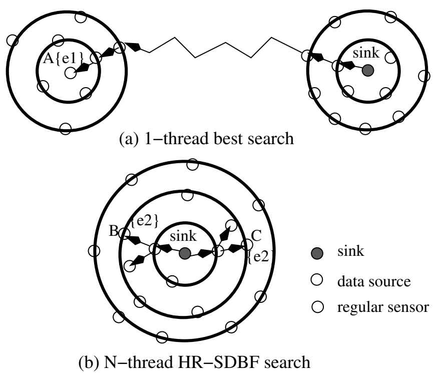


Fig. 1. HR-SDBF routing overview. e_1, e_2: detected events. A, B, C, D: event sources.

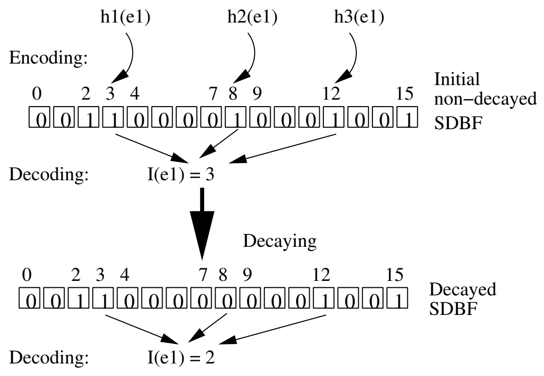


Fig. 2. The structure of Scope Decay Bloom Filter (SDBF).

to the best neighbor that has the maximum amount of hints among all neighbors. If multiple neighbors tie, we can select one randomly. We can also break the tie based on some SDBF component. To control scope decay, each SDBF bit segment is equipped with a TTL counter. A counter for a segment is set to k at an event source if an event hashes to a bit in that segment. A TTL counter decreases by 1 at an advertisement. Ties can be broken by choosing the neighbor with the maximum TTL counter value associated with the desired event (i.e. selecting the one closest to an event source). A sink can control which neighbors are qualified for receiving queries by stipulating the minimum amount of hints that a neighbor must have. This design choice is for finding at least one desired event efficiently.

The second approach is demonstrated in Figure 1(b). A query is redirected to all neighbors whose SDBF sets all bits of an event. A neighbor does not receive the query if it does not have the full amount of hints. This scheme is targeted at efficiently locating all events within the no-decay k-neighborhood.

The hint update is accomplished as follows. A sensor first creates a local SDBF, encoding the set of events detected by itself. This SDBF is broadcast to all its neighbors. A neighbor combines this SDBF with the SDBFs from its own neighbors and propagates the aggregated SDBF. To reduce the routing traffic further, incremental updates to SDBFs are actually disseminated.

We consider two options for decreasing the information about an event, exponential decay and linear decay. In the exponential decay model, each bit that is currently 1 in an SDBF remains 1 at a constant probability p. The amount of information about an event at a node outside the k-hop neighborhood is an exponential function of the distance from the boundary of the k-hop neighborhood of the event source. In the linear decay model, the information about an event is approximately a linear function of the distance from the boundary of the k-hop neighborhood. Neither decay model couples the amount of decay with the particular event source and the particular distance; therefore they do not need to memorize those states.

IV. HR-SDBF Protocol- Detailed Design

This section presents the design details of the HR-SDBF routing protocol. We first introduce our SDBF bloom filter. Then we discuss how to use the SDBF to build up and maintain the probabilistic routing hints. At the end, we describe how to route queries intelligently with the help of these probabilistic routing hints. We make the following assumptions about the wireless sensor network we are concerned about. The event source model is random source model where event sources are selected uniformly at random from all sensors. Each sensor may potentially initiate event queries. An event query is considered successful if at least one desired event is found.

A. Scope decay bloom filter (SDBF)

A SDBF is designed as a lossy channel coding scheme to reduce the amount of network traffic. An SDBF can represent the set membership information and the different amount of information about an element in the set. Similar to a BF, an SDBF also has a bit string of width m and d hash functions, h_1, h_2, ..., and h_d. An SDBF encodes the information about an element similarly to the way a BF inserts an element. Given an element e, the SDBF sets all bits $h_1(e)$, $h_2(e)$, ..., and $h_d(e)$ in the bit string. An SDBF differs from the basic BF in the decoding procedure. A BF obtains the membership information by checking whether all mapped d bits are set or not. An SDBF decodes the information about an element e by computing the number of 1s among the d mapped bits, denoted by $I(e)$. This number ranges from 0 to d. The more bits are set to 1, the larger $I(e)$ is, and there is more information about e.

Fig 2 shows an example of an SDBF where $m = 16$ and $d = 3$. The SDBF is composed of a 16-bit string $bstr$ and 3 hash functions, h_1, h_2, and h_3. When the SDBF initially encodes the information about element e_1, h_1, h_2, and h_3 hash e_1 to bits 3, 8, and 12 respectively and set these bits to 1s. When we decode e_1 from this initial SDBF, we apply these three hash functions to e_1, and compute the number of 1s, $I(e_1)$, in bit positions 3, 8, and 12. Clearly $I(e_1)$

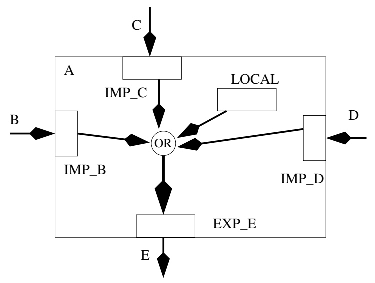


Fig. 3. Event hint update from A to its neighbor E. B, C, D and E are A's neighbors. $LOCAL$: A's local SDBF. IMP_i: the SDBF hint imported (received) from neighbor i.

has the maximum value 3 that corresponds to the maximum information about e_1.

During the decay process, some bits in the initial SDBF are probabilistically reset to 0s. In the decayed SDBF in Fig 2, bit 8 is reset to 0, bits 3 and 12 remain 1s. When we decode e_1 from this decayed SDBF, $I(e_1) = 2$, which means that this decayed SDBF probably has less information about e_1 than the initial SDBF.

There are many design choices for decreasing the information about an element in an SDBF. To simplify the decay process, we choose two stateless decay schemes that do not need to remember the specific event contributing to a particular bit. These two models are *the exponential decay* and *the linear decay*.

The exponential decay model. The information about an element (e.g. an event) decreases exponentially as the distance from the boundary of the k-hop neighborhood of the element source (e.g. the sensor detecting an event) increases assuming that there is no hash collision. Specifically, if any bit in an SDBF is currently 1, it remains 1 at a constant probability p during each decay. The number of 1s corresponding to an element (an event e) at i-hop from the element source is

$$I(e) = d * (p^{i-k}).$$

The linear decay model. The information about an element decreases in a linear fashion as the distance from the k-hop boundary of the element source increases. Let $sumI$ denote the current total number of 1s in the SDBF and r be a random number in the range $[c_1, c_2]$, where c_1 and c_2 are system parameters. Randomly select r bits among the $sumI$ bits and reset these bits to 0. The total number of 1s in the SDBF after the decay, denoted by $sumI'$, is

$$sumI' = sumI - r.$$

Assuming no hash collision, a bit currently being 1 is reset to 0 with the probability p_l.

$$p_l = r/sumI.$$

Obviously, a bit is more likely to be reset to 0 (i.e. decay faster) when the SDBF has a smaller number of 1s.

The SDBF can also be used to represent probabilistic sets. The number of 1s corresponding to an element can be considered as the probability of the element being in the set. The more 1s in the corresponding d bits, the more likely the element is in the set.

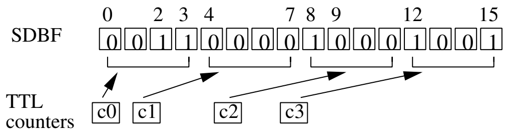


Fig. 4. Control no decay within k-hop neighborhood of an event source: 1 TTL counter per SDBF segment (segment size = 4 bits).

B. Probabilistic routing hints creation and maintenance

Probabilistic routing hints are represented by SDBFs. Each sensor maintains an SDBF for each neighbor. An SDBF encodes hints about events that may be found through a neighbor. To create these hints, each sensor first creates a local SDBF that encodes all local events detected by itself. Then these local SDBFs are propagated according to the decay model. At each sensor, the SDBF hints from different neighbors are first decayed if they contain information outside the k-hop neighborhood of event sources, then aggregated (including the non-decayed local SDBF), and propagated further to other neighbors. Fig 3 shows how a node A propagates updates to its neighbor E. A ORs its own SDBF and the SDBFs it receives from neighbor B, C, and D, and sends the combined SDBF as hints to neighbor E. If a sensor notices some change to its local SDBF, the changes are incrementally spread out to nearby nodes.

The control of no-decay within k-hop is illustrated in Fig 4. An SDBF is partitioned into n segments and one TTL counter is equipped for each SDBF segment. The counter for a segment takes the maximum TTL value from an advertising source that contributes to the segment. In the example, the SDBF is 16-bit and the segment size is 4 bits. Only 4 TTL counters are required.

Fig 5 shows how to create a local SDBF and hint update at a sensor s. To create a local SDBF LR_s for the local event set LE_s, s first checks whether LE_s is empty. If not, s hashes each event e using d hash functions and sets the corresponding d bits in LR_s to 1s. s also initializes the TTL counter for each segment to the maximum value k. If LE_s is empty, LR_s has all 0s in its bit array and counter array. To create a hint update UR_{n_i} to a neighbor n_i, s first includes LR_s into UR_{n_i} if LE_s is not empty. Then for each segment in the SDBF of each neighbor n_j other than n_i, s calls the procedure $Decay()$ to process the SDBF segment according to the pre-defined decay model.

C. Query routing based on SDBF hints

In the 1-thread best search, the sink first finds all neighbors whose SDBFs have at least $minBits$ set among the bits for event e. These qualified neighbors are ranked according to the number of bits set for e. The query is routed to the top-ranked

```
Create a local SDBF $LR_s$ for the local event set $LE_s$:
    // Assume: one TTL counter per SDBF segment.
    // $g$: the segment size
    // $LR_s.barr$: the bit array for encoded elements
    // $LR_s.carr$: the TTL counter array
1.  $if\ !empty(LE_s)$
2.      Reset $LR_s$;
3.      $\forall e \in LE_s$
4.          Set bits $LR_s.barr[h_1(e)]$, ..., $LR[h_d(e)]$ to 1;
5.      $\forall i \in \{1, \cdots, m/g\}$;
6.          $if\ Count\ Ones\ (LR_s.barr, i) > 0$;
7.              $LR_s.carr[i] = k$;
8.          $else$
9.              $LR_s.carr[i] = 0$;

Create a hint update $UR_{n_i}$ to neighbor $n_i$:
1.  $if\ !empty(LE_s)$
    // Include the non-decayed local SDBF $LR_s$
2.      $UR_{n_i} = LR_s$;
3.  $else$
4.      Reset $UR_{n_i}.barr$ and $UR_{n_i}.carr$;
    // Process each segment in each SDBF from a neighbor.
5.  $\forall n_j \in my\_neighbors$ such that $n_i \neq n_j$
6.      $\forall i \in \{1, \cdots, m/g\}$;
7.        $if\ Count\ Ones\ (UR_{n_j}.barr, i) > 0$;
8.          $if\ UR_{n_j}.carr[i]$ - 1 $> UR_{n_i}.carr[i]$
9.              $UR_{n_i}.carr[i] = UR_{n_j}.carr[i]$ - 1;
10.         $if\ UR_{n_j}.carr[i] == 0$
11.         $seg = extractSeg(UR_{n_j}.barr, i)$;
12.             $seg$ = Decay($seg$);
13.         $UR_{n_i}.barr$ = $OR(UR_{n_i}.barr$, i, $seg)$;
14. Return $UR_{n_i}$;
```

Fig. 5. Algorithms for creating local hints and hint updates.

neighbor. On receiving Q_e, a neighbor checks if the query is a duplicate. If so, Q_e is dispatched to a random neighbor. If e is a local event at this neighbor, the query forwarding terminates. If not, Q_e is redirected similarly if the query TTL does not expire.

During query forwarding, there may be top one ties. If tie breaking is random, a neighbor among all ties is randomly selected as the best neighbor. If the tie breaking is TTL counter, a max-min strategy is used to select the best neighbor. First, for each neighbor, the minimum TTL counter value among those of all segments in association with e is chosen as the TTL counter value for that neighbor. Then the neighbor with the maximum TTL counter value with respect to e is considered the best.

The rationale for the max-min TTL counter strategy is as follows. When an SDBF segment is only used by one event, the TTL counter for that segment is set to k at the event source and decreased by one (until 0) at each advertisement. When two events share a segment, if the TTL counter for the segment is set according to one event, the other event can only cause the shared TTL counter to stay the same or increase but not to decrease. Therefore, in each neighbor's SDBF, among all segments of an event, if at least one segment's TTL counter value is not changed by other events, the minimum TTL value is the correct value for that event. If TTL counters of all segments that are related to an event are false, the minimum TTL counter value has the smallest error.

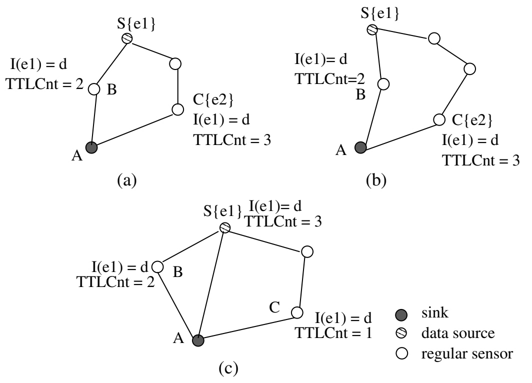


Fig. 6. Scenarios for Theorem 1 ($k = 3$). (a) Error due to shared TTL counters ($g > 1$) alone. (b) Error due to hash collision ($g = 1$) alone (c) Best scenario in 1-thread forwarding.

If all neighbors' TTL counter values are correct with respect to an event, the neighbor with the maximum TTL counter value is closest to the event source. Otherwise, there is no way to distinguish between true and false TTL counter values. Choosing the neighbor with the maximum TTL counter value is like a random selection.

In the N-thread search, the $minBits$ must be the same as d and the maximum query TTL must equal to k. The sink forwards a query Q_e to all neighbors with all d bits set for event e. When a neighbor receives a non-duplicate query Q_e for a local event e, the detailed information about e is returned to the sink. The neighbor continues to send Q_e similarly to the sink until the query TTL expires. Duplicate queries are discarded at the receiving neighbor.

V. Analysis

In this section, 1-thread HR-SDBF with the exponential decay model is analyzed.

*Theorem 1 (**Best query performance within k-hop**):* In the best scenario, the 1-thread query forwarding within k-hop of an event source follows the shortest path if the max-min TTL counter strategy is used to break ties.

Proof: Within k-hop of an event source, a query may be forwarded to a neighbor that is not actually the best due to two factors, sharing one TTL counter ($g > 1$) and hash collision. When $g > 1$, as shown in Fig 6(a), A is the query source, S has the desired event e_1. At node C, a local event e_2 hashes to a different bit in each segment of event e_1. This causes the TTL counter value for C with respect to e_1 to be 3. The max-min TTL counter strategy incorrectly chooses C as the best neighbor. When $g = 1$, hash collision may still cause a false selection of the best neighbor. For example in Fig 6(b), a local event e_2 at node C hashes to the same set of bits as the desired event e_1, which causes $I(e_1)$ at C to be d and the TTL counter value for C with respect to e_1 to be 3. C is incorrectly selected by the max-min TTL counter strategy to be the best neighbor.

Notation	Definition	Value
m	The width of SDBF filter	$12kbits$
d	The number of hash functions	16
g	The SDBF segment size	8
k	The no-decay scope	3
p	The exponential decay rate	$1/8$
$(c1, c2)$	The linear decay control range	$(3, 6)$

TABLE I
MAJOR SYSTEM PARAMETERS

There are four scenarios in which the two factors together cause a false selection of the best neighbor.

- $g = 1$ and no hash collision.
- $g = 1$ and hash collision.
- $g > 1$ and no hash collision.
- $g > 1$ and hash collision.

The best scenario is the first one, where there is one TTL counter per bit and no two events ever hash to the same bit. As illustrated in Fig 6(c), the TTL counter values for all neighbors are true values for the desired event. And they represent the shortest distance between neighbors and the event source. Therefore, the best 1-thread query forwarding within k-hop follows the shortest path. ■

VI. EVALUATIONS

A. Experimental setup

The sensor network is generated by randomly deploying 500 sensors in the field of size $200m \times 200m$. It is assumed that each node can reliably send packets to any node within $R = 15$ meters. The event model is random source. 250/1000/2500 events are randomly distributed among all sensors to simulate small/medium/large event scenarios. Each unique event has 5 replicas. Queries are generated by randomly selecting a sensor as a sink and an existing event as the desired one. In each simulation run for HR-SDBF, hints about events are first propagated according to the scope decay model, then queries are processed. The performance metrics are routing energy efficiency and routing quality. We assume that it costs more to set up a connection than transmitting a single message. The routing energy efficiency is computed in terms of the average number of query messages and the average number of amortized messages. The latter is defined as the sum of the total number of query messages and the routing update messages divided by the number of queries. The routing quality is evaluated based on the query success rate. A query is considered successful if at least one desired event is found. Table 1 lists the major system parameters.

B. Decay models: exponential vs linear

One tradeoff in HR-SDBF design is exponential decay or linear decay. Fig 7 shows the performance contrast between these two choices. The number of events are 2500. The search type is 1-thread HR-SDBF with tie breaking by max-min TTL counter policy. In the simulation, max-min TTL tie-breaking performs significantly better than random tie-breaking. The results are not included here due to space limitation. Fig 7(a) indicates that the two decay models generate approximately the same amount of query traffic. Exponential decay incurs less routing overhead than linear decay, as shown in Fig 7(b). In addition, slightly more queries are successfully resolved with exponential decay than with linear decay, as illustrated in Fig 7(c). Therefore, exponential decay is slightly better than linear decay.

C. 1-thread HR-SDBF vs SQR

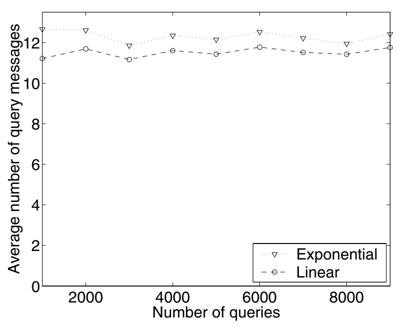
The 1-thread HR-SDBF with tie-breaking using max-min TTL counter values is compared to SQR routing. We first compare their performance by varying the number of queries (i.e. varying the frequency of an event being queried) and the number of events. This is shown in Fig 8.

Each line in Fig 8(a) plots the average number of query messages per query when the number of queries increases and the maximum TTL for a query is fixed at 50. Different lines correspond to the performances of different algorithms under three different event scenarios, 2500 events, 1000 events, and 250 events. Each unique event has 5 replicas. The figure indicates that in the same event scenario, the average query traffic in neither scheme increases as events are queried more frequently. 1-thread HR-SDBF incurs about 33% less query traffic than SQR in all three event scenarios. This is because HR-SDBF does not decay event hints within k-hop neighborhood of event sources and therefore can propagate hints further.

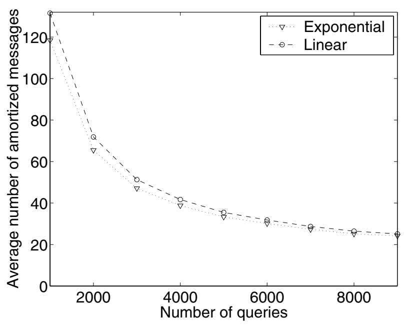
The amortized messages including the update overhead is illustrated in Fig 8(b). In the 250-event scenario, 1-thread HR-SDBF always has a lower amortized traffic than SQR. This is because HR-SDBF generates about the same amount of hint propagation traffic as SQR but significantly less query traffic than SQR. In the other event scenarios, both schemes generate decreasing amortized traffic with increasing queries. 1-thread HR-SDBF has a higher amortized traffic than SQR when the number of queries is small. However, the difference decreases dramatically as the number of queries increases. In the 1000-event/2500-event scenario, HR-SDBF generates less amortized traffic than SQR when the number of queries is greater than 3000/6000. This means that the extra hint propagation traffic caused by no-decay within k-hop in HR-SDBF is effectively amortized with the increase in query frequency.

Fig 8(c) shows the query success rates of both schemes when the number of queries increases. It is observed that in the same event scenario, the number of queries does not impact the query success rate in both schemes. 1-thread HR-SDBF achieves a dramatically higher query success rate than SQR in all three event scenarios. The query success rate of HR-SDBF is about twice as much as that of SQR in the same event scenario. This significant increase is due to the fact that HR-SDBF can push event hints further than SQR.

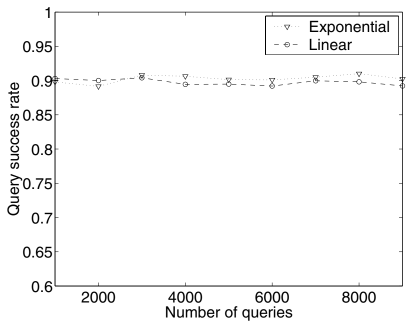
To compare the performance of 1-thread HR-SDBF and SQR when they incur the same per-query traffic. We gathered data with varying maximum query TTLs and plot the query success rate in terms of the amortized per-query traffic, as



(a) Average number of query messages.

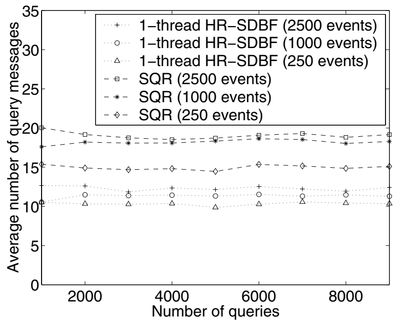


(b) Average number of amortized messages.

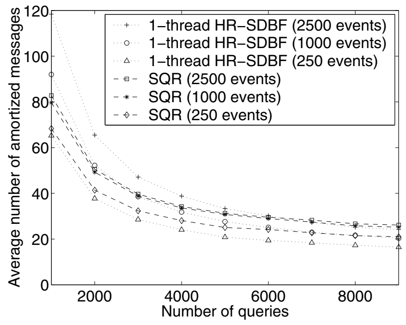


(c) Query success rate.

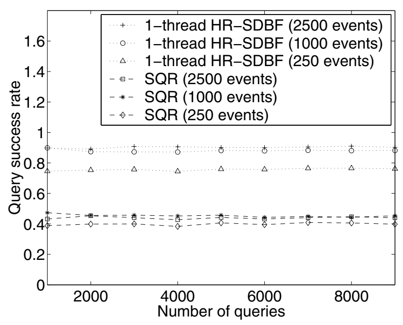
Fig. 7. 1 thread HR-SDBF: exponential decay vs linear decay.



(a) Average number of query messages.



(b) Average number of amortized messages.



(c) Query success rate.

Fig. 8. 1-thread HR-SDBF compared to SQR, varying number of queries ($R = 15$).

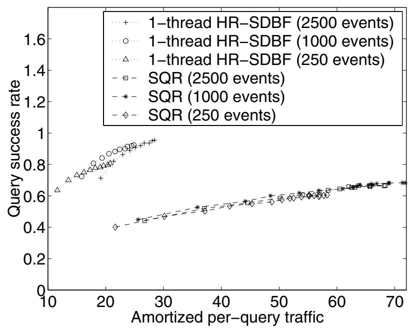


Fig. 9. 1-thread HR-SDBF vs SQR, query success rate in terms of amortized per-query traffic ($R = 15$)

shown in Fig 9. The number of queries is 8000. Clearly, 1-thread HR-SDBF is superior to SQR. It can achieve a higher query success rate than SQR with the same amount of amortized traffic in all three event scenarios.

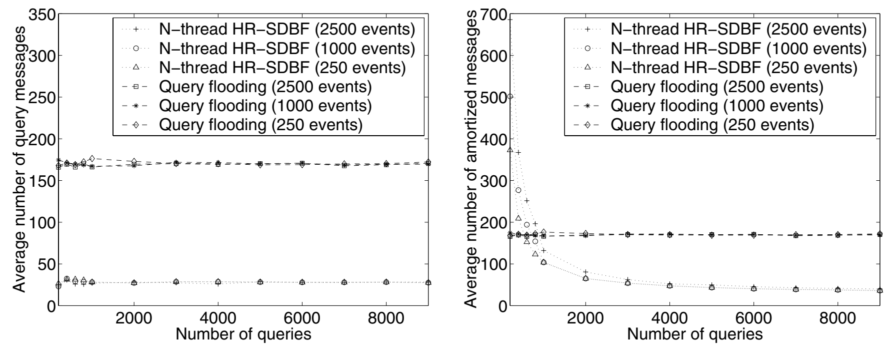
In summary, HR-SDBF generates less query traffic and achieves a significantly higher query success rate than SQR. This is because HR-SDBF does not decay event hints within k-hop neighborhood of event sources. Therefore, hints can propagate further. When there are many events, this no-decay within k-hop also causes more traffic in spreading hints. But the decrease in the query traffic outweighs the increase in the hint maintenance traffic at high query frequencies.

D. *N-thread HR-SDBF vs query flooding*

The N-thread HR-SDBF is evaluated against query flooding. Both schemes have the same query TTL, which is k, the no-loss scope in HR-SDBF. Each query is flooded within k hops in query flooding. In N-thread HR-SDBF, the minimum hint percentage for forwarding a query is 100. A sensor forwards a query to a neighbor only if all bits of the desired event are set in that neighbor's SDBF. Queries are not forwarded outside k-hop neighborhoods of sinks. We are interested in the number of events that both approaches find and the query traffic and the routing traffic that both approaches generate as the number of queries changes (i.e. the event query frequency changes).

The simulation result shows that N-thread HR-SDBF and query flooding locate all events within the same k-hop neighborhood in three event scenarios. Therefore we only plot the query traffic and the amortized traffic incurred by both schemes in Fig 10. The average number of query messages forwarded by N-thread HR-SDBF is always much smaller than query flooding, as shown in Fig 10(a) because N-thread HR-SDBF utilizes hints.

Fig 10(b) shows that query flooding has almost the same amortized traffic when the number of queries changes. N-thread HR-SDBF generates a larger amortized traffic than



(a) Average number of query messages.

(b) Average number of amortized messages.

Fig. 10. N-thread HR-SDBF vs query flooding, varying number of queries ($R = 15$).

query flooding when the number of queries is small ($<$ 1000). However, the amortized traffic in N-thread HR-SDBF drops dramatically as the number of queries increases. When the number of queries is greater than 1000, N-thread HR-SDBF incurs less amortized traffic than query flooding. The number of amortized messages delivered by N-thread HR-SDBF decreases slowly when the number of queries is more than 2000. At 9000 queries, HR-SDBF finds the same number of events with almost three times less traffic. Fig 10 also shows that the number of events does not make an impact on the performance of N-thread HR-SDBF and query flooding.

In summary, when designing HR-SDBF, breaking ties according to the Max-Min TTL counter strategy is significantly better than random selection. The exponential decay model is slightly better than the linear decay model. 1-thread HR-SDBF accomplishes a higher query success rate with the same amortized traffic than SQR. The N-thread HR-SDBF locates all desired events within k-hop neighborhoods but incurs much less amortized traffic than query flooding when events are queried frequently.

VII. Conclusions

In this paper, we proposed a routing protocol called HR-SDBF and a data structure SDBF. The HR-SDBF protocol uses SDBFs to advertise event hints such that the information about an event does not attenuate within the k-hop neighborhood of an event source but decreases outside the k-hop neighborhood with increasing distance. In HR-SDBF, sinks can conduct two types of searches: 1-thread best search or N-thread search. In the 1-thread best search, a node always forwards a query to the best neighbor that has the most hints about the desired event. In the N-thread search, a node directs a query to all neighbors with the full amount of information. Compared to existing query-based routing protocols in WSNs, HR-SDBF increases the query success rate with low amortized routing overhead and reduces energy consumption by keeping probabilistic hints instead of precise hints.

In the future, we plan to explore other decay models in HR-SDBF, such as decaying based on node degrees. We also intend to do analytical and simulation study in extending HR-SDBF to clustered WSNs or actor-sensor model WSNs [17].

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, pp. 102–114, Aug. 2002.

[2] D. P. Agrawal and Q. Zeng, *Wireless and mobile systems.* Thomson-Brooks/Cole, Inc., 2003.

[3] J. N. Al-Karaki and A. E. Kamal, "Routing Techniques in Wireless Sensor Networks: a Survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec. 2004.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramania, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks (Elsevier) Journal*, vol. 38, no. 4, pp. 393–422, 2002.

[5] B. H. Bloom, "Space/time tradeoffs in hash codingwith allowable errors," *Communications of ACM*, vol. 13, no. 7, pp. 422–426, July 1970.

[6] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," in *Fortieth Annual Allerton Conference on Communication, Control, and Computing*, 2002.

[7] L. Fan, P.Cao, J. Almeida, and A. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," in *Proc. of ACM SIGCOMM'98*, Sept. 1998, pp. 254–265.

[8] A. Kumar, J. Xu, and E. W. Zegura, "Efficient and scalable query routing for unstructured peer-to-peer networks," in *Proc. of IEEE INFOCOM'05*, 2005.

[9] D. Guo, H. Chen, X. Luo, and J. Wu, "Theory and network application of dynamic bloom filters," in *Proc. of IEEE INFOCOM'06*, 2006.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks," in *Proc. of ACM Mobi-Com*, 2000.

[11] C. Schurgers and M. B. Srivastava, "Engergy efficient routing in wireless sensor networks," in *Proc. of MILCOM communications for network-centric ops: creating the information force*, 2001.

[12] R. C. Shah and J. Rabaey, "Energy aware routing for low energy ad hoc sensor networks," in *Proc. of IEEE WCNC*, 2002.

[13] L. Li, J. Halpern, and Z. Haas, "Gossip-based ad hoc routing," in *Proc. of the 21st Conference of the IEEE Communications Society (INFOCOM'02)*, 2002.

[14] F. Ye, A. Chen, S. Lu, and L. Zhang, "A scalable solution to minimum cost forwarding in large sensor networks," in *Proc. of 10th international conference in computer communication networks*, 2001.

[15] D. Braginsky and D. Estrin, "Rumor Routing Algorithm for Sensor Networks," in *Proc. of Intl. Conf. Distrib. Comp. Sys.*, Nov. 2001.

[16] A. Ghose, J. Groklags, and J. Chuang, "Resilient data-centric storage in wireless ad-hoc sensor networks," in *Proc. of MDM'03: 4th International Conference on Mobile Data Management (MDM 2003)*, 2003.

[17] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Networks Journal (Elsevier)*, vol. 2, pp. 351–367, Oct. 2004.