Data Dissemination with Ring-Based Index for Sensor Networks

Wensheng Zhang, Guohong Cao and Tom La Porta
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
E-mail: {wezhang, gcao, tlp}@cse.psu.edu

Abstract

In current sensor networks, sensor nodes are capable of not only measuring real world phenomena, but also storing, processing and transferring these measurements. Many data dissemination techniques have been proposed for sensor networks. However, these techniques may not work well in a large scale sensor network where a huge amount of sensing data are generated, but only a small portion of them are queried. In this paper, we propose an index-based data dissemination scheme to address the problem. This scheme is based on the idea that sensing data are collected, processed and stored at the nodes close to the detecting nodes, and the location information of these storing nodes is pushed to some index nodes, which act as the rendezvous points for sinks and sources. We further extend the scheme with an adaptive ring-based index (ARI) technique, in which the index nodes for one event type form a ring surrounding the location which is determined by the event type, and the ring can be dynamically reconfigured for fault tolerance and load balance. Analysis and simulations are conducted to evaluate the performance of the proposed index-based scheme. The results show that the index-based scheme outperforms the external storage-based scheme, the DCS scheme, and the local storage-based schemes with flood-response style. The results also show that using ARI can tolerate clustering failures and achieve load balance.

1 Introduction

Recent advances in electronic and communication technology have enabled the production of battery powered wireless sensor nodes. These nodes are capable of not only measuring real world phenomena, but also storing, processing and transferring these measurements. They can be deployed in the physical world to form a sensor network [2], which enables many applications such as monitoring the habitats of animals, gathering information in a disaster area, and providing traffic information. These applications involve collecting, processing, storing and retrieving a large volume of sensing data generated by the sensor nodes. Since bandwidth and power are scarce resources in sensor networks, how to design scalable and energy efficient data dissemination schemes for sensor networks is a big challenge.

In the past several years, many data dissemination schemes [6, 7, 13, 11, 4, 5] have been proposed for sensor networks. One widely adopted scheme is the *external storage-based (ES)* data dissemination. It relies on a centralized base station, which is external to the sensor network, for collecting and storing sensing data. If many queries are issued from nodes within the network [13], this scheme is very inefficient, since data have to be sent back and forth. Ratnasamy *et al.* [11] proposed a *data-centric storage-based (DCS)* data

dissemination scheme, in which the sensing data of an event (e.g., elephant sightings) are stored at certain nodes within the network. In this scheme, however, data are still pushed in a predefined manner regardless of queries. Hence, it lacks flexibility and may introduce lots of unnecessary data transfers when the querying rate is low.

To avoid unnecessarily transferring the sensing data, *local storage-based (LS)* data dissemination schemes, e.g., directed diffusion [7] and TTDD [13], have been proposed. In these schemes, a source sends data to a sink only when the sink has sent a query for the data. These schemes need certain sink-source matching mechanism to facilitate a sink to find the source holding the data of interest. The matching mechanisms adopted by most LS schemes follow a *flood-response* pattern [5], which inherently needs the flooding of certain control messages. For example, in directed diffusion, a sink floods its query over the whole network, and then the source with the requested data knows where to send the data. In TTDD, the source detecting certain event floods the advertisements of the event to the network, and the sinks interested in the event can send their queries directly to the source. Considering the large number of nodes in a sensor network, the network-wide flooding may introduce significant traffic, especially when the network is queried frequently or the number of detected events is large.

Due to the drawbacks mentioned above, the existing data dissemination schemes may not work well in large scale sensor networks, especially in scenarios where a large amount of sensing data are generated, but only a small portion of them are queried. To address the problem, we propose an index-based data dissemination scheme. In this scheme, the sensing data of an event are stored at the detecting nodes themselves or some nodes close to them (these nodes are called *storing nodes*). A storing node sends data to a sink, only when it receives a query from the sink. Also, the location information (called *index*) of the storing nodes are pushed to and maintained at some nodes (called *index nodes*) based on the event type related to the stored data. Hence, queries for a particular event are routed to the appropriate index nodes. The index-based scheme is more attractive than the existing data dissemination schemes since it avoids both unnecessarily transferring the sensing data and flooding control messages to the whole network.

One major challenge of implementing the index-based data dissemination is how to maintain the indices in the network, such that the indices are highly accessible to sinks and the index nodes are not overloaded. To achieve these goals, we propose an implementation based on the *ring* structure, and call it the *Adaptive Ring-based Index (ARI)* scheme. In ARI, the index nodes for a particular event type are a set of nodes surrounding one or more particular locations (called *index centers* of the event type). The index centers are determined by applying a predefined hash function, e.g., GHT [11], on the event type. Note that the hash function maps an event type to one or more locations within the detecting region of the sensor network. The index nodes for the same event type are connected via some other nodes to form a ring (called *index ring*). The number and locations of the index nodes on an index ring, as well as the shape of the ring, can be adaptively changed to achieve load balance and optimize the system performance.

Extensive analysis and simulations are conducted to analyze and evaluate the performance of the pro-

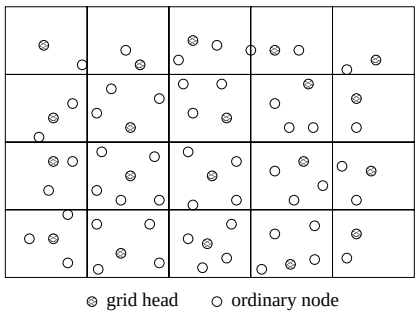


Figure 1: Dividing a sensor network into grids

posed index-based data dissemination scheme, and compare its performance to the existing data dissemination schemes. The results show that, the index-based scheme outperforms the external storage-based scheme, the DCS scheme, and the local storage-based scheme in certain scenarios. The results also show that using the ARI scheme can tolerate clustering failures and achieve load balance.

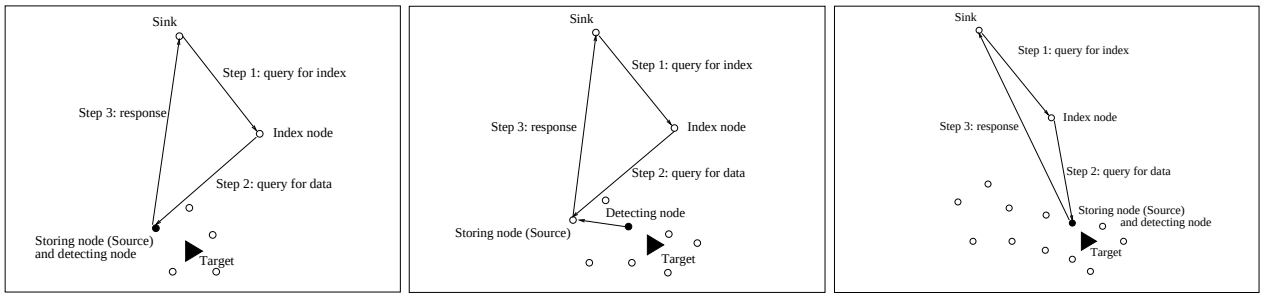
The remainder of the paper is organized as follows. Section 2 presents the system model. Section 3 describes the index-based data dissemination scheme, analyzes the overhead of several data dissemination schemes, and identifies the scenarios in which the index-based scheme performs the best. Section 4 describes the proposed ARI scheme in detail. Performance evaluations are presented in Section 5, and Section 6 concludes the paper.

2 System Model

We consider an application scenario as follows: A sensor network is deployed for monitoring many targets moving within a vast region. The sensor nodes in the network detect the status of each target, and periodically generate sensing data. Many users are also moving within the region. From time to time, a user may issue a query via a sensor node (sink) for the data about the current status of a target and/or a summary of the recent activities of the target.

We assume that the sensor nodes are stationary, and are aware of their own locations using GPS [1] or other techniques such as triangulation [3]. To save power, the nodes stay in the sleep mode most of the time based on the GAF protocol [12]. Using this protocol, as shown in Figure 1, the sensor network is divided into grids, where each pair of nodes in neighboring grids can communicate directly with each other. Grid heads are responsible for forwarding the messages, and other nodes only need to wake up periodically.

Our proposed data dissemination scheme is built atop of GPSR [8], a well-known geographic routing system for multi-hop wireless networks. GPSR uses two distinct algorithms for routing. One is a greedy forwarding algorithm which forwards packets progressively closer to the destination at each hop. The other is a perimeter forwarding algorithm that forwards packets where greedy forwarding is impossible. In this system, which uses both the GAF and the GPSR protocols, packets are forwarded by grid heads. A grid head



(a) (b) (c)

Figure 2: Index-based data dissemination

first tries to forward a packet to its neighboring grid head that is closest to the destination. On failing to find such a node, it forwards the packet to one of the neighboring grid heads based on the perimeter forwarding algorithm.

In our proposed data dissemination scheme, the targets to detect are classified into several types, each of which has a unique key. With certain hash function $H(.)$, a key is mapped to one or more locations within the detecting region of the sensor network. Formally, $(\forall e \in E)H(e) = L_e \in \mathcal{R}$, where E is the set of types and $\mathcal{R}$ is the detecting region.

3 The Index-Based Data Dissemination

In this section, we first describe the basic idea of the index-based data dissemination. Then, we analytically compare the overhead of different data dissemination schemes, and identify the scenarios in which the index-based scheme performs the best.

3.1 Basic Idea of the Index-based Data Dissemination

The basic idea of the index-based data dissemination is illustrated in Figure 2 and is described as follows: A node detecting a target periodically generates sensing data about the target, and stores the data in a storing node, which can be itself or some node nearby. When the target moves, as shown in Figure 2 (b) and (c), the detecting node is changed accordingly. However, the new detecting node still stores the sensing data at the same storing node, until the storing node is far away from the detecting node, and then a new storing node is selected. When a new storing node of a target has been selected, the old storing node processes the data it stores to generate a summary of small size, and sends the summary to the new storing node. Also, the new storing node should register its location at the index nodes for the target. When a sink wants to query the sensing data of a target, it sends a query message to an index node for the target. On receiving the message, the index node forwards the request to the storing node which sends a response directly to the querying sink.

Table 1: Estimating the overhead of the data dissemination schemes

	Overall message complexity	Hotspot message complexity
ES	(1.1) $(\sqrt{N} * r_d + 2 * \sqrt{N} * r_q) * T * n$	(1.2) $(r_d + 2 * r_q) * T * n$
DCS	(2.1) $(\sqrt{N} * r_d + 2 * \sqrt{N} * r_q) * T * n$	(2.2) $(r_d + 2 * r_q) * n$
LS	(3.1) $(N * r_c + 2 * \sqrt{N} * r_q) * T * n$	(3.2) $r_c * T * n + 2 * r_q$
Index-based	(4.1) $(\sqrt{N} * r_s + 3 * \sqrt{N} * r_q) * T * n$	(4.2) $(r_s + 2 * r_q) * n$
	Overall traffic	Hotspot traffic
ES	(1.3) $\{\sqrt{N} * r_d * s_d + \sqrt{N} * r_q * (s_q + s_d)\} * T * n$	(1.4) $\{r_d * s_d + r_q * (s_q + s_d)\} * T * n$
DCS	(2.3) $\{\sqrt{N} * r_d * s_d + \sqrt{N} * r_q * (s_q + s_d)\} * T * n$	(2.4) $\{r_d * s_d + r_q * (s_q + s_d)\} * n$
LS	(3.3) $\{N * r_c * s_i + \sqrt{N} * r_q * (s_q + s_d)\} * T * n$	(3.4) $r_c * s_i * T * n + r_q * (s_q + s_d)$
Index-based	(4.3) $\{\sqrt{N} * r_s * s_i + \sqrt{N} * r_q * (2 * s_q + s_d)\} * T * n$	(4.4) $\{r_s * s_i + r_q * (s_q + s_d)\} * n$

3.2 Analytical Comparison of the Data Dissemination Methods

To compare the overhead of different data dissemination schemes, we introduce the following notations:

- N: the total number of sensor nodes in the network. Similar to [11, 4], we use N to represent the message complexity of flooding a message to the whole network, and use $\sqrt{N}$ to represent the message complexity of sending a message between two nodes in the network.
- T, n: T is the number of types, and n is the number of targets of each type.
- r_d, r_q, r_c, r_s: r_d is the rate of updating the sensing data of a target. r_q is the rate of querying a target. r_c is the rate that the detecting node of a target is changed. r_s is the rate that the storing node of a target is changed. Obviously, $r_d > r_c > r_s$.
- s_d, s_q, s_i: s_d is the size of a data report, s_q is the size of a query message, and s_i is the size of an index update message. Obviously, $s_d \gg s_q$ and $s_d \gg s_i$.

We consider the following metrics when comparing the overhead of the data dissemination schemes:

- Overall message complexity: the number of messages generated in the whole network.
- Hotspot message complexity: the maximum number of messages sent/received/forwarded by one single node in the network.
- Overall traffic: the amount of data transferred in the whole network.
- Hotspot traffic: the maximum amount of data sent, received, and forwarded by one single node in the network.

According to the basic operations of the index-based scheme and the other data dissemination schemes [11], we can estimate the overhead of the data dissemination schemes in terms of the overall message complexity, hotspot message complexity, overall traffic and hotspot traffic. The estimated results are shown in Table 1.

Based on the estimations, we can obtain the following observations:

Observation 1: The index-based scheme has smaller overall message complexity than the other schemes, if:

(1) N is large enough.

(2) $r_s + r_q < r_d$, which can be derived by comparing (4.1) to (1.1) and (2.1).

Observation 2: The index-based scheme has smaller hotspot message complexity than the other schemes, if

(1) $\frac{r_q}{r_c} < T/2$. This relationship is derived as follows:

Let $(r_s + 2 * r_q) * n < r_c * T * n + 2 * r_q$ and $r_s = c * r_c$, where $c < 1$.

Thus, $\frac{r_q}{r_c} < \frac{n*(T-c)}{2*(n-1)} \approx T/2$.

(2) $r_s < r_d$, which can also be derived by comparing (4.2) to (1.2) and (2.2).

Observation 3: The index-based scheme has smaller overall traffic than the other schemes, if:

(1) N is large enough.

(2) $r_s * s_i + r_q * s_q < r_d * s_d$, which can be derived by comparing (4.3) to (1.3) and (2.3).

Observation 4: The index-based scheme has smaller hotspot traffic than the other schemes, if:

(1) $\frac{r_q}{r_c} < \frac{s_i * T}{s_q + s_d}$. This relationship is derived as follows:

Let $\{r_s * s_i + r_q * (s_q + s_d)\} * n < r_c * s_i * T * n + r_q * (s_q + s_d)$ and $r_s = c * r_c$, where $c < 1$.

Thus, $\frac{r_q}{r_c} < \frac{s_i * n * (T-c)}{(s_q + s_d) * (n-1)} \approx \frac{s_i * T}{s_q + s_d}$.

(2) $r_s * s_i < r_d * s_d$, which can be derived by comparing (4.4) to (1.4) and (2.4).

4 An Adaptive Ring-based Index (ARI) Scheme

We now propose an adaptive ring-based index (ARI) scheme. We first present the motivations of the scheme, and then describe the operations including index querying/updating, failure management, and adaptive ring reconfiguration in detail.

4.1 Motivations

One challenge of implementing the index-based data dissemination is how to distribute, update, maintain and query the indices in the network. The scheme should satisfy the following requirements:

- Fault tolerance: Since sensor nodes are prone to failures [2], we should not rely on a single node to maintain an index. Techniques such as replication should be employed to achieve certain level of fault tolerance.
- Load balance: Some index nodes may become overloaded. The scheme should remove the overloaded index nodes and add some light-loaded index nodes.
- Efficiency: Since bandwidth and energy are scarce resources in a sensor network, the scheme should not introduce too much overhead.

One simple scheme can be derived directly from the basic DCS scheme. In this scheme, the index nodes for a target are the nodes which form the smallest perimeter surrounding the location calculated. However, the index nodes can not be changed except when they are failed, and all queries and index updates for the target are processed by these nodes. Hence, the index nodes may become overloaded soon, especially when the query rate or the index update rate for the target is very high. Also, since the index nodes for an event type are close to each other, the scheme can not tolerate clustering failures, which may destroy all the index nodes for an event type.

A hierarchical scheme similar to the structured replication DCS (SR-DCS) [11] or the resilient DCS (R-DCS) [4] can be used to tolerate clustering failures. In these schemes, the whole detecting region is divided into several subregions, and there is one index node in each subregion. Using these schemes, the index nodes in some subregion can still be overloaded if the number of queries issued from that subregion is very high. In addition, a query may be routed through several levels of index nodes before reaching the valid index node.

Tree-based replication of indices [10], which is already applied in peer-to-peer networks, may also be used in sensor networks. However, this scheme may introduce significant maintenance overhead. For example, the failure of the root or some intermediate nodes in the tree may cause the tree to be reconfigured. Furthermore, the nodes should be aware of the addition, removal and migration of index nodes.

To achieve the goals of fault tolerance, load balance and efficiency, We propose the *Adaptive Ring-based Index (ARI)* scheme, which is described in detail in the remainder of this section.

4.2 The ARI Scheme

4.2.1 Initializing an Index Ring

We first describe how to initialize an index ring for the targets of certain type. As mentioned in Section 2, the index center for the targets is calculated using a hash function, which maps a target to a location within the

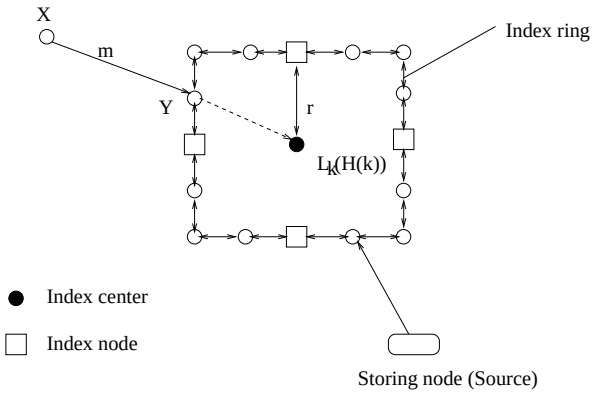


Figure 3: Initializing an Index Ring

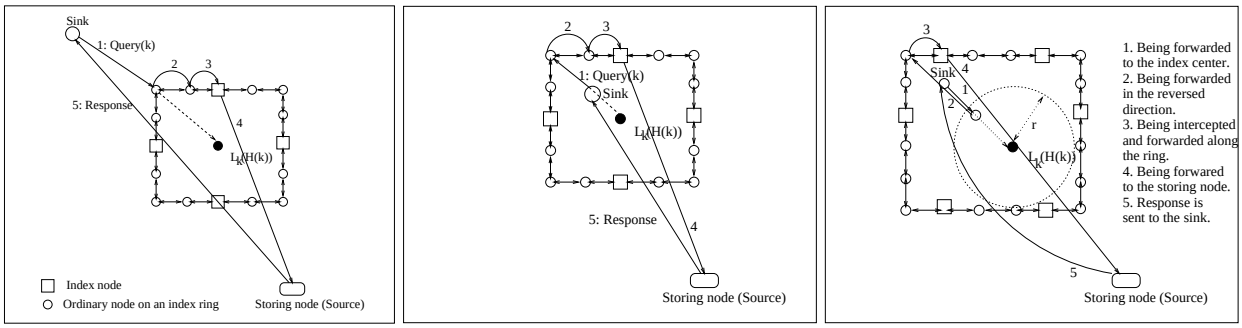
detecting region. Initially, the index nodes for the targets are m selected nodes whose distance to the index center is equal to or larger than r, where m and r are system parameters. Also, these nodes are connected via some other nodes to form an index ring surrounding the index center. The index ring must satisfies the following **conditions**:

(*C1*) The distance between the index center and any node on the ring should be larger than or equal to r.

(*C2*) Any message which is sent by a node outside of the ring-encircled region and is destined at the index center can be intercepted by some node on the ring.

Figure 3 shows an example of an initial index ring, where $m = 4$. The ring satisfies (*C1*), since the distance between each node on the ring and the index center is at least r. Due to the assumption that each node can only forward messages to the nodes in its neighboring grids (refer to Section 2), a message that is sent by a node outside of the ring-encircled region and is destined at the index center, must pass some nodes on the ring. For example, as shown in Figure 3, message m sent by node X can be intercepted by node Y. Thus, the ring also satisfies (*C2*). Note that the index ring may be changed later. But the changed ring must also satisfy the above two conditions.

Parameters m and r affect the system performance. As m becomes large, the number of initial index nodes increases. Hence, the overhead for querying is decreased, at the cost that the overhead for storing and updating index is increased. When r becomes large, the overhead for queries issued by nodes far away from the index center is reduced. Also, more nodes are included in the ring, which improves the fault tolerance level. However, the overhead for queries issued by nodes within the region encircled by the ring is increased, so is the overhead for index update.



(a) (b) (c)

Figure 4: Querying an index

4.2.2 Querying an Index

We now describe the process that a sink queries the index of a target as follows: When a sink S_i wants to query the index of a target, it first calculates its distance to the index center (denoted as L_k) of the target. If the distance is larger than r, the query message is forwarded along the direction of $S_i\vec{L}_k$. Otherwise, the message is forwarded along the direction of $L_k\vec{S}_i$. When a node receives a message, it has different behaviors based on the following cases:

(1) If the node is an index node on the index ring for the queried target: the query is served, and is further forwarded to the storing node of the target if there exist such a storing node.

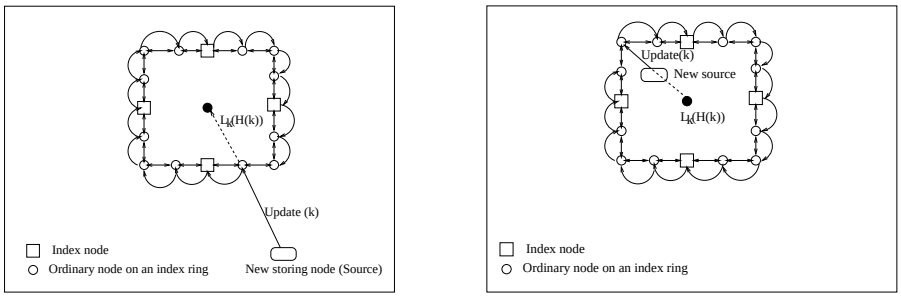
(2) If the node is an ordinary node on the index ring for the queried target: the query is forwarded to its clockwise neighboring node on the ring.

(3) If the forwarding direction of the message is $S_i\vec{L}_k$, and the distance between the node and the index center of the target is smaller than r: the forwarding direction of the query message is changed to be $L_k\vec{S}_i$, and the message is forwarded in the new direction.

(4) Otherwise: the message is forwarded in the specified direction.

Next, we use Figure 4 to further illustrate the querying process. Figure 4 (a) shows the case that sink S_i is outside of the region encircled by the index ring of the queried target. In this case, the query issued by the sink is forwarded along the direction of $S_i\vec{L}_k$, until it is intercepted by some node on the index ring. The node intercepting the query passes the message on the index ring in the clockwise direction, and the query is finally received and served by an index node, which further forwards the message to the storing node of the queried target.

Figure 4 (b) shows another case that S_i is within the ring-encircled region and its distance to L_k is smaller than r. In this case, the query issued by the sink is forwarded along the direction of $L_k\vec{S}_i$ before it is intercepted by some node on the ring. The subsequent process is the same as the previous case.



(a) (b)

Figure 5: Updating an index

Figure 4 (c) shows a complicated case, where S_i is inside the ring-encircled region and its distance to L_k is larger than r. In this case, the query is first forwarded along the direction $\vec{S_iL_k}$. When the message arrives at a node whose distance to L_k is less than r, the forwarding direction of the message is changed to $\vec{L_kS_i}$. The subsequent process is the same as the previous case.

4.2.3 Updating an Index

When the storing node of a target changes, the location of the new storing node should be updated at the index nodes for the target. Similar to querying an index, the new storing node S_o sends an update message along the direction of $\vec{S_oL_k}$ or $\vec{L_kS_o}$, based on whether the node is outside or inside the region encircled by the ring. When the message arrives at some node on the index ring, the node updates the index it stores if it is an index node, and forwards the message along the circle in the clockwise direction. The message is dropped when it is forwarded back to a node that has already received it. Examples are shown in Figure 5 to illustrate the index updating process.

4.2.4 Dealing with Node Failures

To detect the failures of index nodes, neighboring nodes on an index ring should monitor each other by exchanging beacon messages periodically. In this paper, we consider two kinds of failures that sensor nodes may have.

Individual Node Failures:

Due to energy depletion or hardware faults, an individual node may fail. This kind of failures can be tolerated by the GAF protocol, in which a failed grid head is detected and replaced by other nodes in the same grid. When a new grid head is elected after the old grid head (which is on an index ring) fails, the information (e.g., some indices and pointers) held by the old grid head is lost. However, the new head can receive beacon messages from its neighboring nodes on the ring. From these messages, it knows that it is a node on the index ring, and get the lost information from the neighbors.

Clustering Failures:

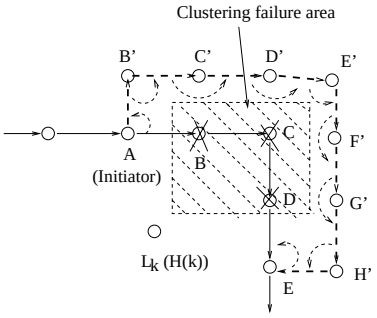


Figure 6: Dealing with clustering failure

Due to some environmental reasons, the nodes within a certain region may all fail or be isolated. This kind of failures is called *clustering failures*. Clustering failures may cause an index ring to be broken. Figure 6 shows a scenario where an index ring is broken after an clustering failure occurs. In this scenario, nodes B, C and D are failed and can not be replaced by other nodes in their grids, since all the nodes in these grids are dead. The failure can be detected by the the neighbors of failed nodes (i.e., nodes A and E), who do not receive beacon messages from the failed nodes for certain time interval. The counterclockwise neighbor (i.e., node A) of the failed nodes initiates a process to repair the ring.

In the repairing process, the initiator A sends out a $recovery$ message, and the message is forwarded around the failed region based on the *right hand rule* [8], until the message arrives at a functioning node on the ring. The process is depicted in Figure 6, and is further explained as follows: The initiator A forwards the $recovery$ message to B', which is the next neighbor of A and is sequentially counterclockwise about A from edge $\langle A, B\rangle$. On receiving the message, C' forwards the message to its functioning neighbor D', which is the next one sequentially counterclockwise about C' from edge $\langle B'C'\rangle$. The process continues, until the message arrives at E, which is a functioning node on the ring. When E receives the $recovery$ message, it consumes the message, and sends an ack message back to A along the reverse path $\langle E, H', G', F', E', D', C', B', A\rangle$. When A receives the ack message, the repairing process finishes, and the broken fragment of the ring, i.e., $\langle A, B, C, D, E\rangle$, is replaced by a new fragment $\langle A, B', C', D', E', F', G', G', E\rangle$.

4.2.5 Ring Reconfiguration for Load Balance

It is possible that some nodes on the index ring become overloaded compared to other nodes. For the purpose of load balance, these nodes should be replaced by some other light-loaded nodes. In the ARI scheme, we propose a simple algorithm to support load balance-oriented ring reconfiguration: Each node maintains a *willing flag* to indicate whether it is willing to be a node on an index ring. Initially, the flag is turned on. Each node i periodically exchanges its residual energy level (denoted as e_i) with its neighbors, and calculates the averaged residual energy level of all its neighbors (denoted as $\overline{e(i)}$). Node i turns off its willing flag when: $e_i < \alpha * \overline{e(i)}$, where α is a system parameter. When a node on the ring turns off its willing flag, it sends a $quit$ message to its counterclockwise neighbor. On receiving the message, the neighbor initiates a process

similar to the ring repairing process (refer to 4.2.4) to remove the quitting node and reconfigure the ring.

5 Performance Evaluations

5.1 Simulation Model

We develop a simulator based on ns2 (version 2.1b8a) [9], to evaluate and compare the performance of the index-based data dissemination scheme (with the ARI scheme) and other data dissemination schemes proposed by previous works, which include the ES scheme, the DCS scheme [11] and the LS scheme (with source-initiated sink-source matching mechanism) [13].

In this simulator, the MAC protocol is based on IEEE 802.11, and the transmission range of each node is $40m$ [7]. 2500 sensor nodes are distributed over a $850 \times 850m^2$ flat field, which is divided into $17 \times 17m^2$ GAF grids, such that there is one sensor node in each grid. For simplicity, we do not simulate the GAF protocol in the simulations.

Several targets (the number of target is denoted as N_t) are deployed in the detecting region. Each target may move in any direction and its average velocity is denoted as v. When a target enters a grid, the sensor node located in that grid can detect it. If the index-based data dissemination scheme is simulated, the sensor node that first detects a target becomes the initial storing node of the target. When the distance between the current detecting node of a target and the storing node of the target is higher than a threshold θ, the storing node of the target is changed to be the current detecting node. Each target has a name, which is an integer between 0 and $N_t - 1$. We use a simple hash function to map the name of a target to one of N_i index centers, which are uniform-ally distributed in the detecting field. The mapping rule is as follows: $H(i) = L_{iMODN_i}$, where i is the name of a target and L_j is the location of the j^{th} index center. Any sensor node can be a sink which issues a query for the data of a certain target. Table 2 lists most of the simulation parameters.

5.2 Simulation Results

The simulations include two parts. In the first part, the overhead of four data dissemination schemes are evaluated and compared to verify the effectiveness of the analytical results presented in Section 3.2. In the second part, the ARI scheme is evaluated and some preliminary results are presented to show that this scheme is resilient to cluster failures and adaptive to balance workload.

5.2.1 Comparing the performance of data dissemination schemes

We first evaluate and compares the overall message complexity of the data dissemination schemes, and the results are shown in Figure 7. The LS scheme is shown to have the largest message complexity, since it needs to flood control messages to the whole network. the ES scheme and the DCS scheme have the same order of message complexity in theory. However, the ES scheme pushes data to a base station outside of the network, and hence has larger overhead for pushing and retrieving data than the DCS scheme, which pushes data to

Table 2: Simulation Parameters

Parameter	Value or range
field size (m^2)	850×850
number of nodes	2500
communication range (m)	40.0
grid side (m)	17.0
number of targets: N_t	10
data update rate: r_d (per target per second)	0.25
number of index centers: N_i	4
the migration threshold for a storing node: θ_s (m)	34.0
initial radius of an index ring: r (m)	34.0
initial number of index nodes on a ring: m	4
simulation time for each experiment	300.0
average velocity of a mobile target: v (m/s)	1.0
size of an update message ($byte$)	10
size of a query message ($byte$)	10
size of a data message ($byte$)	50 or 100

nodes within the network. Figure 7 also shows that the index-based scheme and the DCS scheme has similar overall message complexity. When the query interval is small (i.e., the query rate is high), the index-based scheme has a little bit higher message complexity than the DCS scheme. And the trend is reversed when the query interval is large (i.e., the query rate is low). This phenomenon is consistent with *Observation 1* presented in Section 3.2.

Figure 8 shows the hotspot message complexity of the data dissemination schemes. the ES scheme is shown to have the highest hotspot message complexity, since the hotspot nodes (i.e., the edge nodes close to the base station) need to forward all the messages related to storing and retrieving data, which include the data pushed from the detecting nodes, the queries from the sinks, and the responses sent from the base station. The DCS scheme has much smaller hotspot message complexity, because a hotspot node (i.e. a nodes surrounding a index center) only processes messages related to one type of targets whose data are stored there. The LS scheme is shown to have smaller hotspot message complexity than the DCS scheme. This is due to the following facts. In our simulations, the target does not move quickly and the number of index centers is small (i.e., 4). According to the theoretic estimation results (4.2) and (3.2) presented in Section 3.2, the index-based scheme has small number of messages at hotspots than the DCS scheme. From Figure 8, we can also see that, the index-based scheme has lower hotspot message complexity than the LS scheme only when the query interval is large (i.e., the query rate is small), which is also consistent with *Observation 2* presented in Section 3.2.

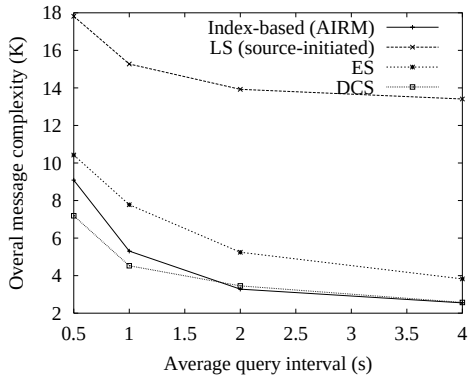


Figure 7: Comparing the overall message complexity

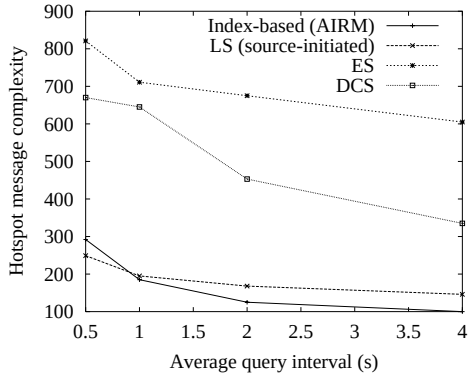


Figure 8: Comparing the hotspot message complexity

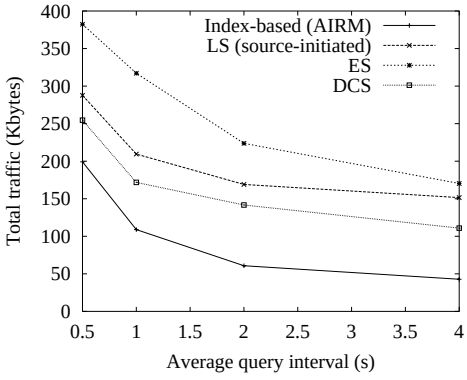


Figure 9: Comparing the overall traffic

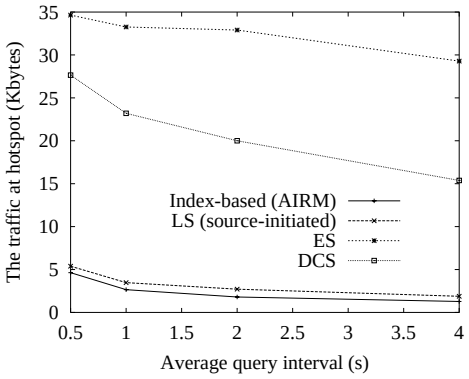


Figure 10: Comparing the hotspot traffic

The overall traffic of the data dissemination schemes are shown in Figure 9. From the figure, we can see that the ES and the DCS schemes cause very large overall traffic, since they both push data from detecting nodes to storing nodes regardless of queries. The ES scheme has larger traffic than the DCS scheme, because it pushes data to nodes outside of the network, and hence brings more communication overhead. The LS scheme also causes large traffic due to the fact that it floods control messages over the whole network. The index-based scheme has the smallest traffic among all the schemes. This is due to the fact that, in this scheme, data are sent from a source to a sink only when the data is queried, and the control messages (i.e., queries and index updates) are never flooded.

Figure 10 shows the hotspot traffic of the data dissemination schemes. From this figure, we can see that the ES scheme has the highest hotspot traffic, which is followed by the DCS scheme. The index-based scheme has the smallest hotspot traffic when the query interval is large (i.e., the query rate is high). The phenomenon is similar to that shown in Figure 8 due to the same reasons as explained before.

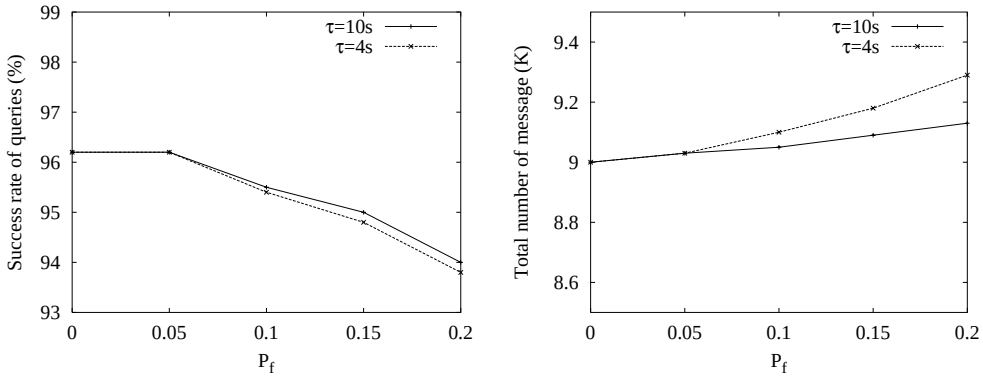
5.2.2 Preliminary Evaluations of ARI

Tolerating to Clustering Failures

We first show how the ARI scheme reacts to clustering failures. In this simulation, every certain time interval (τ), a square region that has a side of $51m$ centered at a node on an index ring has a probability of p_f to fail. The average query interval is $0.5s$, and other simulation parameters do not change.

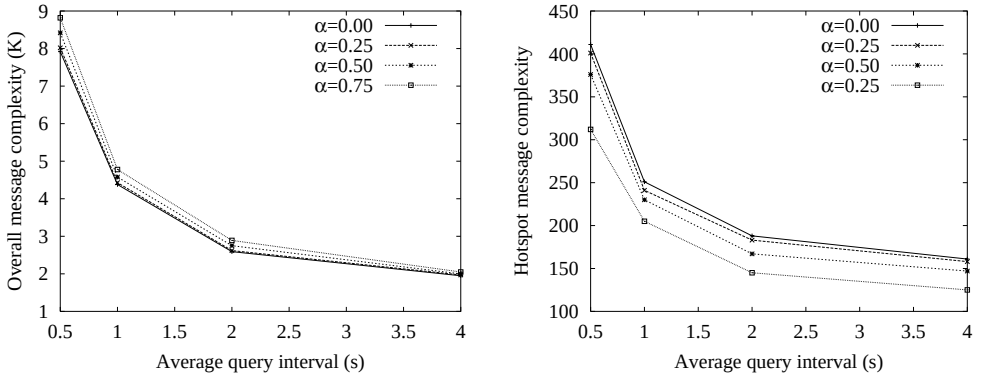
The success rate of the queries and the overall message complexity are measured as τ and p_f are being changed, and the results are shown in Figure 11. From Figure 11 (a), we can see that the query success rate decreases slightly when the frequency of clustering failures increases. This is due to the fact that, when the index ring is broken due to clustering failure, the ARI scheme guarantees that a query can still be routed to a normal node on the index ring, and routed to one functioning index node. Only when all index nodes are failed, which occurs very infrequently, the query is dropped.

Figure 11 (b) shows that the overall message complexity is slightly increased as the frequency of clustering



(a) Query success rate (b) Overall message complexity

Figure 11: Evaluating the fault tolerance feature of ARI



(a) Overall message complexity (b) Hotspot message complexity

Figure 12: Evaluating the load balance module of ARI

failures increases. This can be explained as follows. When an index ring is broken due to clustering failures, the ring should be repaired, and hence some communication overhead is introduced. Also, the repaired ring becomes longer, which increase the overhead for index updates.

Load Balance

Next, we evaluate the ARI scheme's ability to achieve load balance. In this simulation, each node is initially equipped with an energy of $0.2J$, and other simulation parameters are the same as the previous simulations. We measure the overall message complexity and the hotspot message complexity, when the load balance module is turned on (i.e., the load balance parameter is larger than 0) or off (i.e., the load balance parameter is set to 0).

Figure 12 (a) shows the overall message complexity. We can see that, the overall message complexity is increased by about 10% as α changes from 0 to 0.75. This is due to the reason that, the index ring has to be dynamically reconfigured when the load balance module works. With the dynamic reconfigurations, as shown in Figure 12 (b), the hotspot message complexity is decreased by around 25%.

6 Conclusions

In this paper, we proposed an index-based data dissemination scheme with adaptive ring-based index (ARI). This scheme is based on the idea that sensing data are collected, processed and stored at the nodes close to the detecting nodes, and the location information of these storing nodes is pushed to some index nodes. The index nodes for the targets of one type forms a ring surrounding the location which is determined based on the type, and the ring can be dynamically reconfigured for load balance. Analysis and simulations were conducted to evaluate the performance of the proposed index-based scheme. The results show that the index-based scheme outperforms the external storage-based scheme, the DCS scheme, and the local storage-based scheme. The results also show that using the ARI scheme can tolerate clustering failures and achieve load balance.

References

[1] "US Naval Observatory (USNO) GPS Operations," *http://tycho.usno.navy.mil/gps.html*, April 2001.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, March 2002.

[3] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Location For Very Small Devices," *IEEE Personal Communication, Special Issue on "Smart Space and Environments"*, October 2000.

[4] A. Ghose, J. Grobklags and J. Chuang, "Resilient data-centric storage in wireless ad-hoc sensor networks," *Proceedings the 4th International Conference on Mobile Data Management (MDM'03)*, pp. 45–62, 2003.

[5] B. Greenstein, D. Estrin, R. Govindan, S. Ratnasamy and S. Shenker, "DIFS: A Distributed Index for Features in Sensor Networks," *First IEEE Ineternational Workshop on Sensor Network Protocols and Applications*, May 2003.

[6] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocal for Wireless Microsensor network," *Proc. of the Hawaii International Conference on System Sciences*, January 2000.

[7] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication," *MobiCOM '00*, August 2000.

[8] B. Karp and H. Kung, "Gpsr: Greedy perimeter stateless routing for wireless networks," *The Sixth Aunual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000)*, Aug. 2000.

[9] The CMU Monarch Project, "The CMU Monarch Projects Wireless and Mobility Extensions to ns," *http://www.monarch.cs.cmu.edu/cmu-ns.html*, October 1999.

[10] M. Roussopoulos and M. Baker, "Cup: Controlled update propagation in peer-to-peer networks," *Proceedings of the 2003 USENIX Annual Technical Conference*, June 2003.

[11] et al. S. RatNasamy, "GHT: A Geographic Hash Table for Data-Centric Storage," *ACM International Workshop on Wireless Sensor Networks and Applications*, September 2002.

[12] Y. Xu, J. Heidemann and D. Estrin, "Geography Informed Energy Conservation for Ad Hoc Routing," *ACM MOBICOM'01*, July 2001.

[13] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang, "A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks," *ACM International Conference on Mobile Computing and Networking (MOBICOM'02)*, pp. 148–159, September 2002.