DIFS: A Distributed Index for Features in Sensor Networks

Benjamin Greenstein*, Deborah Estrin*, Ramesh Govindan†, Sylvia Ratnasamy‡, and Scott Shenker§
*Department of Computer Science, University of California at Los Angeles, email: {ben,destrin}@cs.ucla.edu
†Department of Computer Science, University of Southern California, email: ramesh@usc.edu
‡Intel Research, Berkeley, email: sylvia@intel-research.net
§International Computer Science Institute, email: shenker@icsi.berkeley.edu

***Abstract*— Sensor networks pose new challenges in the collection and distribution of data. Recently, much attention has been focused on standing queries that use in-network aggregation of time series data to return data statistics in a communication-efficient manner. In this work, rather than consider searches over time series data, we consider searches over semantically rich high-level events, and present the design, analysis, and numerical simulations of a spatially distributed index that provides for efficient index construction and range searches. The scheme provides load balanced communication over index nodes by using the governing property that the wider the spatial extent known to an index node, the more constrained is the value range covered by that node.**

I. Introduction

There has been considerable interest recently in research in wireless sensor networks, a technology that promises analysis of and interaction with the environment at spatial and temporal densities not possible using conventional approaches. The nodes in such networks are equipped with sensors, local storage, CPUs and radio communication facilities, allowing them to both sense the local environment and communicate locally with other sensors in order to construct semantically rich conclusions about the environment that they are sensing, such as detecting the presence of animals, or of hotspots, or of other "events" [12], [15], [16], [23].

The primary resource constraint of nodes in such networks is energy. Nodes are expected to be long-lived (deployed not for hours, but for years), untethered (both in terms of communication and power), and unattended (and so must be self-configuring and self-adapting). Energy must be carefully budgeted and conserved, so all sensornet algorithms must minimize energy use. The primary energy consumer in such systems is radio transmission. For one scenario, Pottie and Kaiser explain that the cost of transmitting 1Kb a distance of 100 meters is approximately equal to the cost of executing three million CPU instructions [21]. Furthermore, the cost of reception in these systems is often almost as much as that of transmission.

Sensornets collect a tremendous amount of detailed time series data about the environment. As sensornet research and experience has accumulated, many different approaches for accessing this data have been proposed. The conventional approach to storing time series data is to have all sensing nodes feed their data to a central repository external to the sensing environment. While this approach provides for complete flexibility in processing the data, it incurs significant energy expenditure to send every sensor reading to an external site. Furthermore, links near a gateway to an external storage repository will become communication bottlenecks as the network size and amount of data produced increase. Consequently, in energy-constrained sensor networks it may be necessary to store data locally at or near the location of generation.

One approach to retrieve this stored data is to flood a query to all nodes that could potentially have suitable data, and have them send their response to the (perhaps external) querying node. In such an approach, data is sent when (*i.e.*, in response to an actual query) and where it is required. Some queries will originate within the sensornet itself, and in that case it makes little sense to send the data to an external site only to have it shipped back to the internal querying node. Furthermore, if far more data is collected than is actually required to answer queries, then this *local storage* approach results in significant energy savings.

There are two extensions of this approach that lead to further energy savings. First, the data can be processed, aggregated, and/or pruned as it propagates toward the query sink. The authors of Directed Diffusion, TAG, and others describe particular forms of in-network aggregation and pruning of data that can select relevant data and produce statistics such as *medians*, *averages* or *maximums* [16], [1]. This approach uses "data-centric" routing in that queries are not directed towards individual nodes, but rather are stated only in terms of the desired data. Second, the data can be processed locally to identify high-level "events" that are of interest. These events can refer directly to sensor readings, such as areas of relatively high temperatures, or to the conclusions of rather sophisticated identification algorithms, such as animal or vehicle sightings. In either case, the queries are directly for such events, and the responses contain summarized data about such events. Here, too, the routing is data-centric, but the queries (and responses) deal with higher-level abstractions.

These energy saving extensions reduce the energy required to respond to queries, but do not alter the basic "flood-then-respond" approach which incurs an inherent cost of flooding each query to all nodes (or at least to all nodes that could possibly have relevant data). If the rate of queries is relatively

high, this expense can be substantial.[1]

In contrast, the "data-centric storage" (DCS) approach, proposed in [10], avoids the flooding of queries. All events are named, and then stored at a network location based on the name. Queries for that particular kind of event are routed to the appropriate network node, where the relevant data (or pointers to that data) can be found. Storing the data by name provides a logical rendezvous mechanism between data and queries so that queries need not be flooded. GHT [11] describes a specific solution to achieving DCS in which event names are hashed to geographic locations and stored at the node closest to the hashed location. For improved efficiency and load balancing, GHT proposes *structured replication* in which a rendezvous point is replicated so that events can be stored at, and retrieved from, the rendezvous point closest to the detecting node.

While the basic idea of data-centric storage is quite general, the original instantiation was *binary* in the sense that it was limited to reporting whether a certain high-level event had occurred. If the events had additional attributes, such as temperature or humidity, one had no way of efficiently scoping the request based on the values of these attributes. For example, to discover only those event occurrences that recorded a temperature between 50 and 60 degrees, one had to query each of the replicated rendezvous points individually, instead of querying only the relevant rendezvous points.

In this paper, we extend the data-centric storage architecture to efficiently support range queries–that is, queries where only events with attributes in a certain range are desired. DIFS, our proposed distributed index, provides for low average search and storage communication requirements and seeks to balance these requirements over participating nodes.

Note that DIFS, like GHT, is well-suited to scenarios where the nature of some archetypal high-level events is well-defined. In such cases, efficient index structures (such as DIFS) may be applied that save on communication overhead since only data about high-level events is communicated rather than the lower-level time series data from which events are composed.[2] However, when the notion or type of events is not yet clearly defined, alternate search mechanisms may be required. For example, a protocol such as DIMENSIONS [6] also relies on the placement of data within the sensornet and the use of data-centric rendezvous points but, unlike DIFS, works with lower level sensor readings. DIMENSIONS takes time series data as input and compresses it while retaining significant features. This compressed data is then stored within the sensornet to produce a multiresolution map. Such maps allow users to drill down into areas that appear to contain significant phenomena without requiring a pre-defined notion of what constitutes such phenomena.

We describe events and provide a classification of their properties in Section II. Section III describes the rationale behind the design of DIFS. Section IV discusses one possible index on event data. Section V demonstrates the DIFS model. In Section VI we describe other ongoing projects. VII discusses the path ahead and Section VIII concludes our paper.

[1]However, if the queries are relatively few, and the desired data streams are long-lived (such as monitoring the temperature readings in a local hotspot), then this approach is reasonably efficient.

[2]Users can always retrieve low-level readings by having each event notification include the event's location, so that to gather detailed data one need only download the readings from the relevant sensors.

II. Events and Queries

The authors of TAG describe how time series queries can be categorized [1]. They classify the partial state of the query as distributive, algebraic, holistic, unique, or context sensitive. They separate those queries that are duplicate sensitive from those that are not; those that are exemplary from those that are summary; and those that are monotonic from those that are not. We seek to provide a similarly detailed categorization of range queries for high-level events, and provide four query categories.

A. High-Level Events

High-level events, such as a hot region or a target detection, a map, a histogram, or a contour, can be described in a number of ways. We propose adding new data structures to store high-level data abstractions to the simple attribute types provided by Diffusion. Like Diffusion, such abstractions would be defined system-wide at deployment time. Such abstractions would include vectors, maps, histograms, parametric equations, and n-degree functions. It is our intention that individual attributes describing a high-level event would be indexed using our system. Although the queries we address in this paper are for ranges and distributions, future work should do a more careful investigation of the tradeoffs between this and other approaches for range and binary queries.

B. Classification of Event Properties and Relationships

The classification proposed in this section has primarily been designed with attribute range and distribution queries in mind. It is for future work to investigate how it might be extended to apply to detections (as in GHT) and other domains.

The goals of a system directed at binary events like "elephant sightings" are different from our goals of providing range searches over events that are each comprised of attributes with values. The fundamental goal of a search over binary events is to determine the locations of such events. When such events are rare (*i.e.*, the ratio of events generated to nodes that are capable of generating events is low), it is much more energy-efficient to construct a rendezvous point where events could register and queries could search than to flood a search. Events defined by attributes with values that fall within a specified range are by definition less common. For example, there may be many hot regions in a network, but few with a heat gradient with a slope greater than s. For this reason we develop a new method to support range queries efficiently. This paper proposes mechanisms to run on top of GHT to address range queries, for which GHT alone was not intended.

We propose the following classification of the properties of and relationships among high-level events:

TABLE I
EVENT PROPERTY AND RELATIONSHIP CLASSIFICATION

Sensor Values	Timing Parameters	Spatial Dimensions	Event Interrelationships
Time series data and associated statistics	Sequence and duration of events	Physical shape and size of an event	Relationships over time and space between events

- **Sensor value(s).** This category includes raw sensor values that comprise high-level events, as well as composite measurements and summary statistics such as average, median, maximum, standard deviation, etc. Some examples of sensor values are the peak temperature of a hot region, the radiation flux density in the area of a sunfleck, the variance in direction of a gust of wind, and the speed at which an animal target is moving. Searches for sensor values may either be over a designated range or section of the distribution. Sensor values are typically represented as integers or floating point numbers.
- **Timing parameters.** Often it is not enough to know a particular value for a region, but it is integral to know how this value varies over time. For example, one might care about a hot region that has been hot for some period of time, or that has increased in temperature over time, or that has moved above a minimum speed for a period of time.
- **Spatial dimensions (including shape and size).** This refers to the physical shape and location of an event. Some examples of queries over spatial dimensions include hot regions larger than a given area, elongated beyond a certain ratio, with a primary hot ridge greater than a certain length, or with some defined curvature. Regions can be described as enclosing circles, ellipses, or polygons. Their centers or other points of interest can be represented in integer or floating point coordinates [7].
- **Relationships between events.** Finally, there are relationships between events. In the spatial domain, this translates to proximity or intersection. Is, for example, an area of high CO_2 concentration also an area of bright sunlight? Is the event of an animal detection near other events of animal detection? In the temporal domain, this translates to succession and temporal separation. Did an area of high CO_2 concentration come about immediately after bright sunlight? Is the hottest region in the sensing environment at the same time also the driest?

This classification serves as a logical base for potential queries of high-level events. Of course, there are often cases in which these classes are used in combination as in, "Did the hot region change shape over time?" and "Did the peak temperature of the region move location?" or composed, as in "return a region that has moved more than 10 feet in the last hour and has become elongated." Such queries can be evaluated in a straightforward manner using simple comparators. Alternatively, more complex evaluation functions can be disseminated to generate ranked results for top-k queries. A user might be interested, for example, in large high temperature regions, but heat might be more important than size.

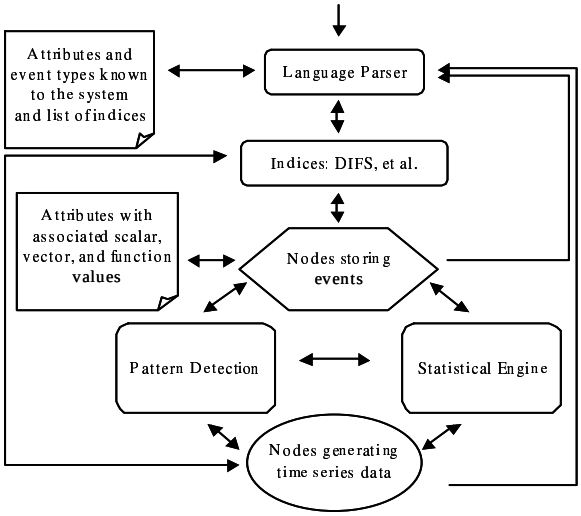


Fig. 1. A storage and search architecture.

C. The Big Picture

Before delving into the details of our index protocol, we describe where such an index fits into the bigger picture of a storage and search architecture for sensor networks.

Sensor nodes typically generate time series data. This data is locally processed by statistical and pattern recognition engines to generate high-level events. These events, in turn, are stored locally where they are created, and information about their various attributes is inserted into indices. A human user or interested automaton (a sensor node or actuator) poses queries to these indices. Query results then are found in the indices themselves, at the storage nodes, and possibly, when high-level event descriptions are not complete, even at the nodes that generate time series data.

Figure 1 describes the high-level relationships among the components of the storage and search architecture for sensor networks that is briefly described above.

In terms of event generation and search, nodes serve two functions. First, all nodes in the network may be used to store raw time series data and events. Second, a subset of nodes serve as index nodes to facilitate search.

III. APPROACH

Since DIFS builds on top of GHT, we begin with a brief description of GHT in Section III-A explaining its limitations when used for range queries. In Section III-B, we use a quad tree as an example of how range queries might be achieved using a traditional hierarchical search structure. The limitations of GHT and the quad tree motivate the design of DIFS. We

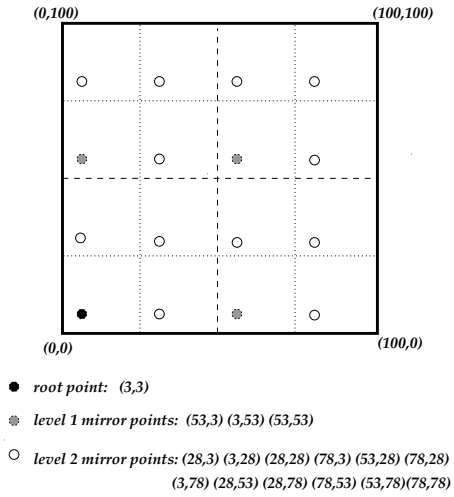


Fig. 2. Example of Structured Replication with a 2-level decomposition.

end this section with a quick overview of the DIFS search structure in Section III-C.

A. GHT: A Geographic Hash Table

GHT provides a key/value-based distributed index. Events are named with keys. Both the storage of an event and its retrieval are performed using these keys. In GHT, a key is hashed to a geographic position; geographic routing is used to locate the node closest to this position, which then stores the data associated with that key and, in general, acts as the rendezvous node for that key.

By hashing keys, GHT spreads the load due to different keys evenly throughout the sensornet. When many events with the same key are stored, GHT avoids creating a hotspot of communication and storage at their shared rendezvous node by employing *structured replication*, whereby data for a single event is divided among multiple mirrors. Structured replication uses a hierarchical decomposition of the geographic space similar to that used in GLS [20]. Let us say that an event name hashes to a geographic location r, which acts as the root for that name. Given r and a hierarchy of depth d, one can compute $4^d - 1$ mirror images of r. For example, Figure 2 shows a $d = 2$ decomposition, and the mirror images of the root point $(3, 3)$ at every level.

Now, an event is stored at the associated mirror that is closest to the detecting node. Queries, however, must now be routed to all mirror nodes[3], thus trading off lower communication costs for the storage of events versus more expensive querying.

In GHT, keys are event names and hence range queries over the *values* associated with an event are not efficiently supported; though only a small fraction of event occurrences might actually fall within the specified range, such queries would still have to visit every mirror node. While it might seem like an obvious fix would be to propagate summary information towards the root node while aggregating at mirror nodes along the way, this results in a hotspot at the root node since all range queries must traverse through the root. (This problem of overloading root nodes in tree-based hierarchies is discussed in greater detail in the following section.)

[3]Geographically scoped queries need only be disseminated to the subset of nodes located within the spatial extent specified in the search criteria.

B. Simple Quad Tree Approach

Perhaps the simplest approach to indexing values in a sensor network is to build a spatially distributed quad tree of histograms. A root index node maintains four histograms describing the distribution of data in each of four equally sized quadrants of the network. Each quadrant, in turn, maintains histograms for each of four subquadrants. In such a manner, an index tree is formed in which a parent node covers exactly four times the network area of each of its children. In this approach, data events are assumed to be stored locally at or near the node(s) that created them.

The simple quad tree approach differs from the hierarchical decomposition of structured replication in three aspects. First, a quad tree forms a search tree over events. Branches of the tree may be pruned during search if descending them would not add to the result set. Structured replication does not propagate information about event detections to the root. Therefore branches of the search tree can never be pruned and therefore all searches must descend to all leaf nodes. Second, the leaf and internal nodes in structured replication are derived from the event key and hence parent nodes need not explicitly point (*i.e.*, know the precise node address of) to its children. This makes a structured replication hierarchy more robust and easier to maintain. Finally, each parent in a quad tree has four children, one for each quadrant of the area the parent covers. Each parent in structured replications has three children. The parent itself serves to cover the fourth quadrant. Like structured replication, DIFS uses hashing to define its hierarchical structure; like a quad tree, DIFS maintains histograms at each internal node in the hierarchy, describing the values found for an event attribute in the region an index node covers.

Quad trees allow for efficient searching since the histograms can direct queries to only the relevant nodes. However, two problems arise when applying hierarchical search trees, such as a quad tree, to a distributed and energy-constrained setting. First, every time data to be indexed is generated anywhere in the network, that information must be propagated to the root. Since all data must be propagated to the root and all queries originate from it, the root handles significantly more traffic than any other node. This problem can be partially alleviated by caching data changes in intermediate level nodes and only propagating such information toward the root periodically. Note that this problem does not apply to structured replication since in that scheme, event detections are not propagated to the root. Second, every query over the entire spatial domain must originate with the root. If only a few queries are to be posed, this might not be a problem. Imagine, however, a network of thousands of actuators each capable of inserting multiple

queries. The one-root approach clearly is not scalable to such a situation.

As outlined in the following section, DIFS achieves efficiency by using histograms but avoids the load balance problem by avoiding the use of a single tree-based hierarchy.

C. DIFS overview

DIFS extends GHT to support efficient range queries while maintaining balanced load across nodes. DIFS achieves this by constructing a multiply rooted hierarchical index that differs from traditional binary and quaternary trees in that non-root nodes can have multiple parents. Nodes store event information for a particular range of values detected within a particular geographic region. Higher-level nodes cover smaller value ranges detected within large geographic regions while lower level nodes cover a wider range of values from within a smaller geographic region. The key idea behind the construction of this hierarchy is to incorporate (in addition to the event name) the value of an event, as well the location of the detecting node in determining the storage node for that event occurrence. Using this index, DIFS can efficiently support range queries, queries related to the distributions of values in space and so forth.

We include the quad tree in our performance evaluation to serve as an example of a conventional hierarchical approach. Our evaluation also covers structured replication.

IV. Distributed Indexing of High-Level Events

DIFS was created with the following design goals in mind:

- Even at the cost of a modest increase in overall traffic, a sensor network search solution should be load balanced both in terms of message traffic and storage. Hierarchical approaches that drill down through search trees are useful in general, but have limitations for sensor networks. The root of any such tree becomes a communication bottleneck, receiving the full brunt of search communication.
- Searches for data by name, value range, and location should be communication-efficient and fast.
- The solution should also allow for efficient answers to queries on the distribution of data, such as "Is my event's flux density in the top 10 percent of values seen?"
- Transactions should be reliable.

A. DIFS Design

DIFS was designed to provide the search efficiency of a quad tree in a manner that balances communication load across the index. Like GHT, DIFS uses a geographical hash within a hierarchically decomposed key space. Like a quad tree, it constructs a search hierarchy of histograms. Unlike the single-tree hierarchies of structured replication and quad trees, in DIFS each child has *bfact* parents, where *bfact* $= 2^i, i \geq 1$. Moreover, the range of values a child maintains in its histograms is *bfact* times the range of values maintained by its parents. That is, nodes in a DIFS hierarchy all have the following defining property: The wider the spatial extent an index node knows about, the more constrained the value range it covers.

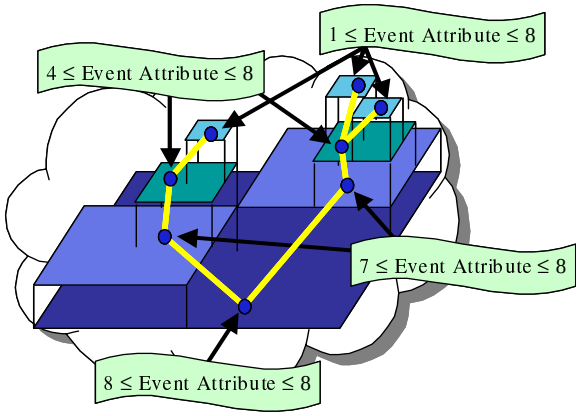


Fig. 3. An illustration of the DIFS hierarchy.

To make this idea more concrete, consider a high-level event with an attribute called *flux density*, which represents the radiation flux density of an instance of this high-level phenomenon. It is known perhaps that the flux density of such a phenomenon has never been recorded above 255 and is always at least zero. Suppose an event is detected and a flux density of 57 is to be recorded in the index. In a DIFS, this value would be stored in a local leaf node covering a value range of 0 to 255. It would also be stored in a parent of the leaf that covers a range of 0 to 63, in a grandparent covering 48 to 63, and in a great-grandparent with a value range of 56 to 59. The geographic area the parent covers is four times that of the leaf. The area covered by the grandparent is four times that of the parent and the area of the great-grandparent is four times that of the grandparent, and perhaps covers the entire spatial extent of the network.

To ensure a balance of communication load over the network, the range of values that an index node knows about is inversely related to the spatial extent the node covers. Rather than having one query entry point, as in the root of a quad tree (as will be shown in subsection IV-D), DIFS searches may originate at any nodes in the tree, including those below the root level. Query entry points are selected in accordance with both the spatial extent, as well as the range of values requested in the query.

To build this hierarchy, a node storing an event forwards information first to the local index node with the narrowest spatial coverage but covering the widest value range. This index node then forwards a histogram describing the values it has seen to a node with wider spatial coverage but narrower value range, and so on and so forth. The convention by which index nodes are selected is similar to that of GHT in that a geographical hash function is used. However, rather than hashing to any location in the network, the DIFS hash function limits its output to the area that a node in the hierarchy is to cover. Specifically, the DIFS hash takes a source location, a string of characters to hash, and a bounding box, and produces a location. The smaller the bounding box, the more high-order bits of the source location that are included in the

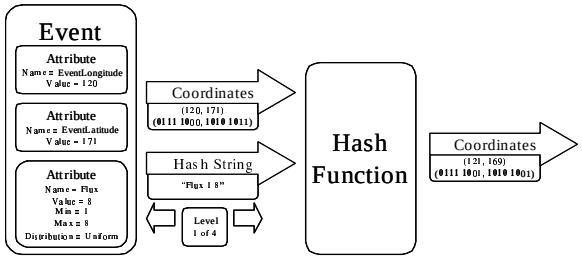


Fig. 4. A geographically bounded hash function.

output location. The expected distance of an index node from the event source node increases as one progresses up the hierarchy. In particular, the expected distance from a node to its parent is one half the expected distance from its parent to its grandparent.

B. Geographically Bounded Hash

The DIFS indexing method requires the use of a geographically bounded hash function that takes as input a location, a key, and a bounding box, and returns a location within that box. Our geographical hash has the same functional form as the geographical hash used in structured replication.

To simplify index construction and to provide a convention for index node selection, the partitioning of the network must be regular. The network is assumed to be rectangular and is divided into four quadrants of equal size. Each quadrant, in turn, is again divided into equal sized subquadrants. This process is repeated as many times as there are levels in the index hierarchy.

As a consequence, the bounding box given as input to the hash function must exactly describe a quadrant for some desired level in the hierarchy. Assuming an origin of (0, 0) at one corner of the network, the starting *x* and *y* coordinates of the bounding box must be integer multiples of the box's width and height, respectively.

The output of the hash function is a pair of coordinates somewhere within the bounding box that is supplied as input. To achieve this, first a hash location (x_h, y_h) is generated that falls somewhere within the network at large. Two separate hash functions are used to generate the output *x* and *y* coordinates respectively. The *m* high-order bits common to all points in the input bounding box and enough low-order bits of (x_{int}, y_{int}) to produce a valid pair of coordinates are concatenated. The value of *m* is dependent on the level in the DIFS for which an index node is sought.

For example, if the box is bounded by (0, 0) and (15, 15) and coordinates are one byte each, the four leading zeros common to all points in the bounding box are combined with the four low-order bits of the hash results to produce a hash location within the box.

C. Inserting a High-Level Event

Suppose potential query sinks are interested in the flux density attribute of various events. Below, we describe how an event containing this attribute will register itself with a DIFS index.

The first step in registering this event is to compute a geographically bounded hash on the concatenation of the attribute name and the range of values held by a local leaf index node. A character delimiter such as a colon is inserted between the name and each value to avoid ambiguity. Recall from Section IV-B that such a hash produces a location within a specified bounding box. This first bounding box has the dimensions of the system-specified minimum coverage region of an index node. For example, if the network has a length *l* and a width *w* and the DIFS is to have *h* levels, then the width and height of the minimum bounding box are $l/2^{h-1}$ and $w/2^{h-1}$ respectively.

As in the quad tree, the node storing the event sends a message containing the location where the event is stored and the value for flux density to the geographically closest leaf-level index node, which in turn stores the flux density and a pointer back to the storage node. In practice, this leaf-level node is the node geographically closest to the result of the bounded geographical hash function.

Recall quad tree nodes each have one parent. DIFS nodes have *bfact* parents. Periodically (the interval is system-defined), the leaf forwards information it maintains as a histogram to *bfact* parents that each are responsible for maintaining information for $1/bfact$ the values of the leaf. While these parents each hold information on $1/bfact$ of the values, they cover four times the spatial extent of a leaf. Consequently such parents each have four children.

Again periodically, parents forward information in the form of histograms to grandparents, and grandparents forward to great-grandparents until nodes covering the entire spatial extent of the network are reached. Every node except the leaves has exactly four children and every node except the roots has exactly *bfact* parents. A parent covers four times the spatial extent of each of its children. A child covers *bfact* times the value range of its *bfact* parents.

Suppose an event has been found in the vicinity of geographical coordinates (14, 37) and has a flux density of 57. Also, for the time being, suppose nothing is known a priori about the expected distribution of flux density other than that it always falls in the range [0-255]. Let the minimum bounding box have a width of 8 units. The hash will first be called with the key "flux density:0:255" and will return a location somewhere in the bounding box defined by the corners (8, 32) and (15, 39). A message containing the string "flux density", the coordinates (14, 37), and the value 57 will be sent to a leaf-level (level 0) index node.

Suppose for the moment that this is the only instance of an event in the system with a flux density attribute and that *bfact* = 2. The level 0 leaf index node will forward a histogram containing counts for values 0 to 127 to a level 1 index covering the region (0, 32) to (16, 47). The level 1 nodes will in turn forward a histogram containing counts for values 0 to 63 to a level 2 node covering the region (0, 32) to (32, 63). This process is continued until the value is forwarded to a

node covering the entire network. Each index node at levels 1 and above stores four histograms, one pointing to each of the composite lower-level index nodes. The level 0 indices point directly to storage nodes.

D. Querying for a set of events

Queries over a DIFS may be described by value range or by range in the distribution. Both types of queries may be constrained geographically.

By value. A user might be interested, for example, in flux densities ranging from 47 to 68. This query is decomposed and forwarded to those nodes that can service the query in the following manner.

As in the previous sections, let *bfact* be the factor by which the range of index values is decreased as one progresses up the index hierarchy. That is, if a level 0 node has a range of 256 values and *bfact* is 2, then a level 1 node will have a range of 128 values. If *bfact* is 4, then a level 1 node will have a range of 64 values.

The decomposition of the query selects the smallest number of nodes that exactly covers the query range. Suppose we have a *bfact* of 4 and a square network of width 128 with minimum index coverage of width 8 and a range of values between 0 and 255. Each level 0 leaf node covers 256 values over a width of 8, each level 1 node covers 64 values over a width of 16, each level 2 node covers 16 values over a width of 32, each level 3 node covers 4 values over a width of 64, and each level 4 node covers 1 value over a width of 128.

The minimum set of index nodes exactly covering this range is:

- One level 4 node for value 47
- One level 2 node for range [48-63]
- One level 3 node for range [64-67]
- One level 4 node for value 68

Using the same hash function as in storage, the query will start at these index nodes and propagate down to any events that satisfy this query.

Over space. In the previous subsection, the search was conducted for a range of values over the full spatial extent of the network. DIFS handles location criteria specified by a bounding box as well. The straightforward approach to geographically constraining a query is to parse the initial search range as was done above, but to send the query only to those index nodes that cover any part of the geographical range of interest. If any of these nodes covers no more of the region of interest than one of its children, we use the child instead. If we use the child, we make the same test recursively, to use the grandchild, great-grandchild, etc.

By distribution. One of the added benefits of percolating histograms up the index is that queries on the distribution of data can effortlessly be executed so long as each histogram passed contains a total count of values below the histogram range and a total count of values above the histogram range. An index node, for example, that maintains counts for values 5, 6, 7, and 8, would also keep a count for less than 5 and greater than 8. Given these two added bits of information, an index node knows exactly where in the distribution its values fall.

A question such as "What is the minimum value in the top ten percent of values?" may be answered using a binary search of root-level index nodes. A more intelligent strategy than a binary search may be applied if an estimated distribution (*e.g.*, normal) is provided. Alternatively, a special *distribution node* may be tasked with collecting the distribution from root-level nodes and with keeping that distribution current.

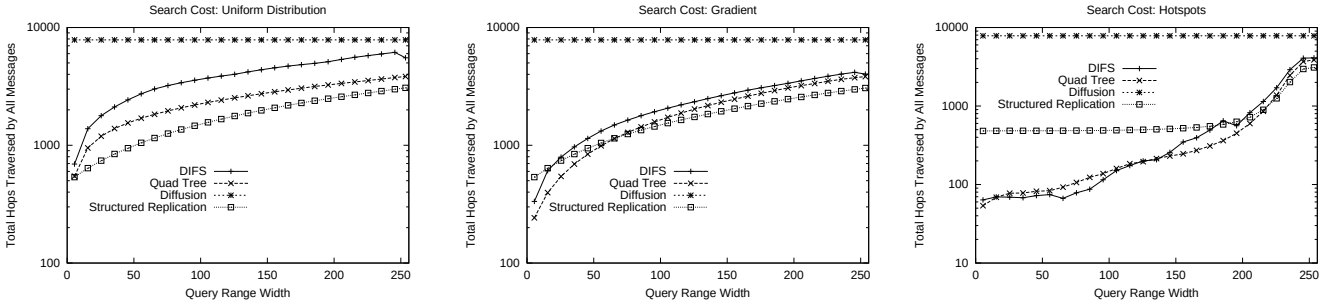
E. Removing a High-Level Event

Leaf-level index nodes control the insertion of data into the index, as they are the only nodes that directly communicate with storage elements. They also control storage-initiated deletion from the index using two mechanisms: *explicit deletion* and *timeout*. On the one hand, since leaf-level nodes are responsible for propagating histograms to their parents, an explicit deletion requires only removing a pointer to the storage location of the event to be deleted and decrementing the histogram bin associated with that event's attribute value. Timeout deletions, on the other hand, require that a time stamp be maintained with each source pointer in a leaf index. Periodically, these time stamps are checked and, if found to be older than a system-defined age, are deleted. A second method by which index values may be deleted is *search-initiated*. That is, a node may propagate a "delete" message for a range of values in the same way a search propagates. As in explicit delete and timeout, leaf nodes delete pointers associated with the value range specified. Search-initiated deletion can also be used to "un-task" the system from indexing mundane sections of the distribution. For example, if searches always tend to be for the top and bottom ten percent of values, there is little utility in expending the energy to index the middle 80 percent. A search-initiated deletion may contain a "never again" flag to prevent the system from ever again indexing a desired range.

V. Simulation and Analysis

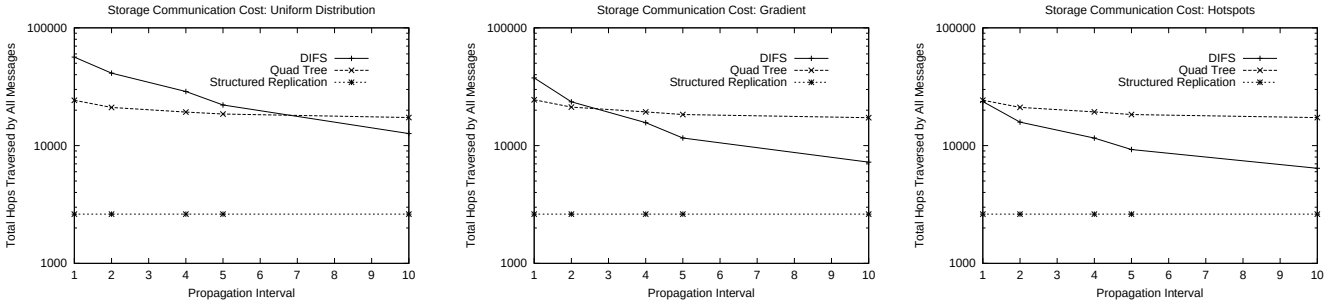
In this section we compare DIFS to the quad tree, Diffusion, and structured replication using numerical simulations. All simulations are performed over a 1024 meter wide square topology. The communication costs of search and storage as well as bottleneck analysis are presented for 2048 node scenarios with a communication radius of 25m. Three distributions are considered for the event values to be indexed: uniform, gradient, and hotspot. For all three distributions, 2048 events are generated over the time interval [0,10] at uniformly random locations. For the uniform case, the value generated for each event is also random. For the gradient case, the value generated is proportional to the x coordinate of the event's location. For the hotspot case, five peak locations are chosen at random. An event's value is inversely proportional to the minimum distance to any one of these peak locations.

Our simulations have been devised to evaluate the performance of our scheme for queries related to a possible range or distribution of values for a particular attribute of a high-level event. As our index was designed for range queries, we



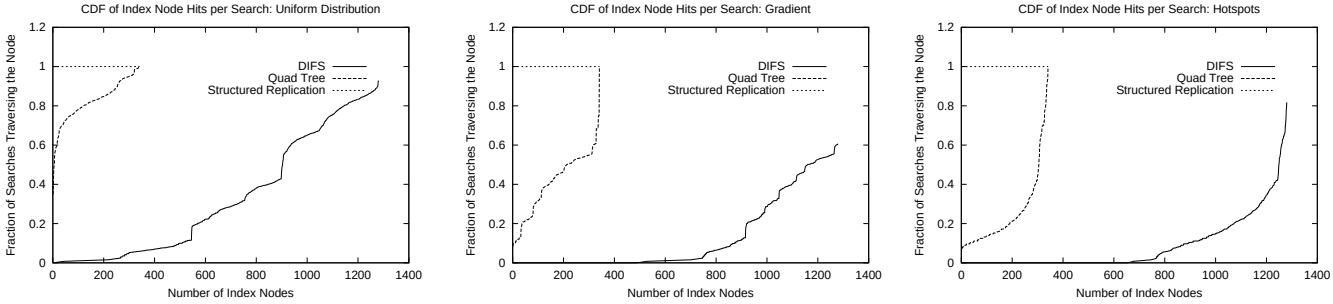
(a) Uniform Distribution (b) Gradient (c) Hotspots

Fig. 5. The cost of search in terms of the number of per hop receptions of the search message as a function of the width of the search range for 2048 nodes with a transmission radius of 25.



(a) Uniform Distribution (b) Gradient (c) Hotspots

Fig. 6. The communication cost of storing values in the indices in terms of the number of receptions of message as a function of the interval between propagations up the index hierarchy for 2048 nodes with a transmission radius of 25.



(a) Uniform Distribution (b) Gradient (c) Hotspots

Fig. 7. The CDF of index node involvement in a search. 2048 nodes with a transmission radius of 25.

do not investigate the performance of our index for binary queries, but would expect performance results comparable to GHT with structured replication.

We present the quad tree and structured replication as examples of techniques that rely on a hierarchical decomposition of the sensing environment. The key difference between these two techniques is that the hierarchy in the quad tree forms a search tree in which branches may be pruned during search, whereas the hierarchy in structured replication forms a dissemination tree whereby a query descends to all leaves. We do not make comparisons to GHT without structured replication because the single rendezvous point of GHT without structured replication is not suitable to range searches in networks with dense event generation. Finally, in our presentation of search costs, we also include Directed Diffusion, a technique that searches by flooding, as a reference.

Diffusion's performance is directly related to the deployment density and hence the degree of connectivity of participating nodes. When connectivity is high, it performs rather poorly when compared to the other techniques because of its underlying nature of flooding queries, in which every node broadcasts once, and every neighbor of every node receives once for each broadcast. In our comparisons, we use a network that is, on average, minimally connected. We set the transmission radius to $1/\sqrt{\rho}$ where ρ is the density in nodes per square meter.[4]

In our simulations, all queries were for a range of values. Queries were not constrained by geography. Search and storage communication cost results are given in terms of the aggregate number of communication hops traversed by all messages as a function of the query range width.

In dense deployments, such as the ones presented in this section, a uniform distribution of data values results in the worst-case search costs for DIFS and quad tree searches because most searches will result in all leaves being explored. Figure 5(a) demonstrates that in this situation, DIFS and the quad tree approaches only perform better than structured replication for extremely narrow search ranges. DIFS, the quad tree, and structured replication all perform roughly an order of magnitude better than Diffusion.

In the gradient case, every leaf is not necessarily explored by DIFS and the quad tree. Consequently, the search cost is less for these two techniques than structured replication for search ranges of less than about 20 values. In queries with larger search ranges, the greater dissemination costs of DIFS and the quad tree dominate the total cost of search, and hence both are outperformed by structured replication.

Finally, in the hotspot case we see that search tree branch pruning of DIFS and the quad tree lead to significant performance improvements over the non-pruning technique of structured replication.

There is an enhancement to DIFS that is not presented in these simulations that should improve the search cost somewhat. Suppose there are 256 leaf nodes each holding the full range of values. In the current version of DIFS, a search over the entire value range would result in 256 query messages being sent, each with an average path length of approximately $n/2$, where n is the width of the network, resulting in a total message distance of $128n$. If DIFS were to use the same dissemination technique as in structured replication for these cases, the cost could be reduced dramatically. For a tree of depth d, the path length for the same scenario would be $\frac{4^{d-1}n}{2^d}$. For $d = 5$ (as in the simulations), the message distance would be $16n$.

DIFS was designed for scenarios in which there are many queries relative to stores. It incurs a greater storage-related communication cost than non-indexing techniques because event values propagate to interior nodes of the tree. The communication cost of storing event data within indices is presented in Figure 6. These plots show the effect of varying the time between propagations of data up the hierarchy. In these simulations, events are generated in the network roughly every 0.004 seconds. These graphs show the communication cost of storage for intervals ranging from one to ten seconds. In general, both DIFS and the quad tree are outperformed by GHT with structured replication. The incorporation of the propagation delay into DIFS allows it to have storage costs that are better than those of the quad tree.

One of the design goals of DIFS was to provide a load balanced solution to range searching. Figure 7 describes the cumulative distribution function of the involvement of nodes in search. In the uniform case, some of the nodes of DIFS are involved in every query. This is because most of the leaf nodes are involved in every search. This is also the case for the quad tree. The quad tree's root is also involved in every search. In structured replication, every node in the tree is involved in every search.

The results for the gradient case show that the bottleneck nodes of DIFS are involved in a much smaller fraction of searches than in the quad tree or in structured replication. The hotspots case demonstrates this result in the extreme. The bottleneck nodes are involved in only about one-fifth of the searches.

In summary, structured replication performs best in terms of communication for storage. This is because beyond the initial registration of an event with its local leaf level mirror point, no additional work needs to be done to form a search tree. DIFS and the quad tree approach form search trees and pay for this tree formation up front. Queries with sufficiently constrained search criteria will lead to the pruning of branches in the search trees of both DIFS and the quad tree and will consequently lead to lower overall search costs. Queries requesting the most of the entire range of data will result in no pruning and will perform no better using DIFS or the quad tree than using structured replication. In DIFS, when the query range

[4]Unicast transmissions are assumed to require one transmission and one reception per hop. Broadcast transmissions, as are used in flooding, require one transmission and as many receptions as there are nodes in the sender's transmission radius per hop. Routing is assumed to be on the straight line path from sender to receiver and the number of hops from sender to receiver is assumed to be the total source to destination distance divided by the expected distance between a node and its neighbor closest to the destination. For an infinitely dense network, this is just the transmission radius.

is sufficiently constrained, the average bottleneck utilization is greatly reduced. In the quad tree, the bottleneck is always the root and its utilization is 1. In structured replication, every node in the tree is explored during every search.

VI. Related Work

In the short time since sensor networks have emerged as an academic area of widespread research, many query dissemination and response architectures have been proposed [1], [3], [4], [5], [6], [11], [16]. Directed Diffusion is one publish/subscribe architecture for sensor networks. Standing queries are disseminated into the network via controlled flooding. Gradients are constructed for efficient reverse path forwarding of data. This model works best when there are very few queries relative to the amount of data produced by the network that matches them.

Another approach is to leverage the strong spatial and temporal correlation typical of time series data. If, for example, a node senses warmth, then nearby nodes are likely to also sense warmth. Likewise, if the node is warm right now, it follows that it was likely warm just a moment ago. DIMENSIONS uses wavelet compression to reduce data redundancy due to spatial and temporal correlation [6]. It produces a multiresolution index (or view) of data. High-level events have attributes such as size, shape, and lifetime that do not necessarily correlate well in space or time. A more generic indexing mechanism is therefore needed.

The authors of TAG [1] propose a method to construct a search tree to collect statistical aggregates from raw time series data. An important contribution of this work is a list of canonical search types for time series data with details of their associated properties. Our work also proposes as classification, but is different in that the classification is for high-level events. Their queries are standing and continuous and do not attempt to provide a mechanism for retrieving data created before a query is instantiated. There is no storage system in their scheme other than a buffering mechanism to collect intermediate results at nodes while data is aggregating and propagating toward a sink.

In GADT, a continuous probability distribution function is proposed as a new object-relational abstract data type that models physical data as Gaussian pdfs. The focus is on a data representation, not on the indexing of such a representation [3].

Bonnet *et al.* describe the Cougar system, a distributed database for sensor networks [4]. They summarize characteristics they believe are common to all sensornet queries. The queries are long-running. The desired result is a series of notifications of system activity. Queries need to correlate data over space and aggregate data over time and often are constrained to a geographical region. In contrast, we address queries that run above the aggregators and correlators that construct high-level events. Consequently, their system operates on time series data.

VII. Future Work

DIFS is work in progress. As such, there are many issues identified for future work. Of highest priority is to implement DIFS on an experimental platform, to collect and aggregate real data into high-level events, and to evaluate DIFS's performance over such events. Below is a list of other areas for future and continued work.

The distribution of values stored in histograms is implicitly assumed to be uniform. If the distribution is known a priori, then the width of the histogram bins in the system should be adjusted to balance the storage and communication requirements of all index nodes. Dynamic repartitioning when the distribution changes over time is a subject of future study.

The underlying mechanism for robustness to node failure is that of GHT, in which data is cached along the minimum routing perimeter surrounding a hash location. A structured replication service caches data at remote locations in the network (in case of regional outages), and is a subject of future work.

All transactions are assumed to be reliable. This is not a formal requirement of the system, but is instead a starting point. Applications that can tolerate some degree of loss of data will be considered in future work.

We also plan to investigate the construction of an index keyed on more than one attribute. Such a scheme might create separate indices for each attribute or might collect the various attributes into one DIFS/kd-tree hybrid. Alternatively, n-dimensional histograms might be used for n attributes and would be propagated through one index. Finally, rather than treating attributes separately, an evaluation function may be disseminated to storage nodes and run over multiple attributes to produce a ranking that would then be indexed.

Furthermore, it is a topic of future work to investigate how indexing only part of the possible range of values affects system performance.

Hierarchy construction uses hashing. The resulting hash locations may serve directly as index storage nodes, or they may be index rendezvous nodes that know the current optimal locations of storage nodes. The latter, in effect, builds an overlay over an overlay and is a subject for future work.

The search strategy presented in this paper breaks down a search range into component ranges that each map directly to an index node in the tree. For example, a query range of 16 to 28 (with a *bfact* of 4) is broken down into components with ranges of 16-19, 20-23, 24-27, and 28-28. The query components are sent directly to the associated index nodes using independent unicasts, each with an average distance traveled of roughly half the network width. In cases in which the search range is small, and hence the covering nodes are higher up in the tree, each node covers a wider spatial extent, and consequently, fewer index nodes must be contacted to disseminate the search to the entire spatial extent of the query. In cases in which the range width is large, covering nodes may live lower down in the tree and may consequently cover smaller spatial extents; many more messages must be sent

to disseminate the search. At some range width, the cost of disseminating the query to the covering set is greater than the cost of dissemination using structured replication.

Two enhancements may be introduced to improve DIFS search cost. The first is to route the query using hierarchical dissemination, as in structured replication, rather than sending unicast messages to each of the covering nodes. The second is to route to nodes in the highest tree level that will cover the entire query range, rather than decomposing the query range into a minimal covering set.

The minimum cost of search (which is the cost of search when no data matches the query) should be used as a heuristic to determine which enhancement to use. It is a subject of future work to evaluate these enhancements, as well as to determine how the adjustment of *bfact* and the number of tree levels affect overall performance.

VIII. Conclusion

We presented an index structure for sensornets that provides for energy-efficient and load balanced range searches over previously generated high-level events. This index is designed to satisfy queries requesting data within a range of values or falling within some section of the distribution. Queries may be geographically constrained.

DIFS performs best when there are many queries posed relative to the number of events generated, when each result set is small, and when the result set is not likely to be distributed over the network in such a manner as to be discovered by most leaf level search nodes. Consequently, structured replication is preferred when many more events are stored in the system than searches are posed and when at least some of the result set would be found via most of the leaf level mirrors.

The quad tree described in the paper outperforms DIFS both in terms of the aggregate communication cost of storage and of search, but is not scalable to a large number of searches or stores. The root of a quad tree is involved in every search and unless a technique such as progressive resolution degradation or interval updating is employed, the root is involved in every store as well. DIFS scales well to large-scale networks by using a multiply rooted tree and a geography/value coverage tradeoff that balances communication overhead over many nodes.

The proposed event classification and the introduction and analysis of the DIFS scheme, as well as the comparison of DIFS to structured replication and the quad tree should serve as an initial framework for further work in this area.

References

[1] Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. *TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks.* OSDI, 2002.

[2] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman, Mehul Shah. *Adaptive Query Processing: Technology in Evolution.* IEEE Data Engineering Bulletin 23(2): 7-18, 2000.

[3] Anton Faradjian, J. E. Gehrke, and Philippe Bonnet. *GADT: A Probability Space ADT For Representing and Querying the Physical World.* To appear in Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), San Jose, California, February 2002.

[4] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. *Towards Sensor Database Systems.* In Proceedings of the Second International Conference on Mobile Data Management, Hong Kong, January 2001.

[5] Philippe Bonnet, J. E. Gehrke, and Praveen Seshadri. *Querying the Physical World.* IEEE Personal Communications, Special Issue on Smart Spaces and Environments, Vol. 7, No. 5, pp. 10-15, October 2000.

[6] Deepak Ganesan, Deborah Estrin, John Heidemann, *DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks?* To Appear in First Workshop on Hot Topics in Networks (Hotnets-I), October 2002.

[7] Badri Nath and Dragos Niculescu, *Routing on a Curve* To appear in First Workshop on Hot Topics in Networks (Hotnets-I), October 2002.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. *Oceanstore: An architecture for global-scale persistent storage.* In Proceedings of ACM ASPLOS. ACM, November 2000.

[9] A. Rowston and P. Druschel. *Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility.* In 18th ACM SOSP, volume 1, Lake Louise, Canada, October 2001.

[10] Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., and Estrin, D., *Data-Centric Storage in Sensornets.* To appear in the First ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets 2002), Princeton, NJ, October, 2002.

[11] Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., and Shenker, S., *GHT: A Geographic Hash Table for Data-Centric Storage.* First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA 2002), Atlanta, GA, September, 2002.

[12] J. Hill, R. Szewczyk, A Woo, S. Hollar, D. Culler, and K. Pister. *System architecture directions for networked sensors.* In Proceedings of ASPLOS-IX, pp. 93 104, Cambridge, MA, USA, November 2000.

[13] W. Adjie-Winoto, E. Schwartz, and H. Balakrishnan, *The Design and Implementation of an Intentional Naming System.* In Proceedings of the Symposium on Operating Systems Principles, pp. 186-201, Charleston, SC, USA, Dec. 1999.

[14] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, *Habitat monitoring: application driver for wireless communication technology.* In 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, Costa Rica, Apr. 2001.

[15] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, *Building efficient wireless sensor networks with low-level naming.* In Proceedings of the Symposium on Operating Systems Principles, pp. 146-159, Banff, Alberta, Canada, Oct. 2001.

[16] C. Intanagonwiwat, R. Govindan, and D. Estrin, *Directed diffusion: a scalable and robust communication paradigm for sensor networks.* In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000), Boston, MA, USA, Aug. 2000.

[17] J. M. Kahn, R. H. Katz and K. S. J. Pister. *Mobile networking for smart dust.* In ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, USA, Aug. 1999.

[18] B. Karp and H.T. Kung. *GPSR: greedy perimeter stateless routing for wireless networks.* In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000), Boston, MA, USA, Aug. 2000.

[19] S. Kumar, C. Alaettinoglu, and D. Estrin, *Scalable object-tracking through unattended techniques (SCOUT).* In Proceedings of the 8th International Conference on Network Protocols (ICNP), Osaka, Japan, Nov. 2000.

[20] J. Li, J. Jannotti, D. DeCouto, D. Karger, and R. Morris, *A scalable location service for geographic ad-hoc routing.* In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom 2000) Boston, MA, USA, Aug. 2000.

[21] G. Pottie and W. Kaiser, *Wireless integrated network sensors.* Communications of the ACM, 2000.

[22] T. Schoellhammer, *Distributed Pattern Matching in Sensor Networks.* Poster CENS Opening Ceremony, October 2002.

[23] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, *Next Century Challenges: Scalable Coordination in Sensor Networks.* In Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom 1999), Seattle, Washington, August 1999.