Networking Issues in Wireless Sensor Networks

Deepak Ganesan [*], Alberto Cerpa [*], Wei Ye [**], Yan Yu [*], Jerry Zhao [**], Deborah Estrin [*]

Abstract

The emergence of sensor networks as one of the dominant technology trends in the coming decades [1] has posed numerous unique challenges to researchers. These networks are likely to be composed of hundreds, and potentially thousands of tiny sensor nodes, functioning autonomously, and in many cases, without access to renewable energy resources. Cost constraints and the need for ubiquitous, invisible deployments will result in small sized, resource-constrained sensor nodes.

While the set of challenges in sensor networks are diverse, we focus on fundamental networking challenges in this paper. The key networking challenges in sensor networks that we discuss are: (a) supporting multi-hop communication while limiting radio operation to conserve power, (b) data management, including frameworks that support attribute-based data naming, routing and in-network aggregation, (c) geographic routing challenges in networks where nodes know their locations, and (d) monitoring and maintenance of such dynamic, resource-limited systems. For each of these research areas, we provide an overview of proposed solutions to the problem and discuss in detail one or few representative solutions. Finally, we illustrate how these networking components can be integrated into a complex data storage solution for sensor networks.

[*] Dept of Computer Science. UCLA, Los Angeles, CA 90095
[**] USC/ISI 4676, Admiralty Way, Marina Del Rey, CA 90292
Email address: `deepak@lecs.cs.ucla.edu` (Deepak Ganesan).

1 Introduction

The availability of micro-sensors and low-power wireless communications will enable the deployment of densely distributed sensor/actuator networks for a wide range of applications. Application domains are diverse and can encompass a variety of data types including acoustic, image, and various chemical and physical properties. These sensor nodes will perform significant signal processing, computation, and network self-configuration to achieve scalable, robust and long-lived networks [2]. More specifically, sensor nodes will do local processing to reduce communications, and consequently, energy costs. Akyildiz *et al.* [3] provide a comprehensive overview of different aspects of research in sensor networks. In this paper, we will take a more in-depth look at networking challenges, including more recent techniques in this area.

Sensor networks pose interesting challenges for networking research. Foremost among these is the development of long-lived sensor networks in spite of energy-constraints of individual nodes. Sensor nodes are expected to be battery equipped, and deployed in a variety of terrains. In some of these deployments, it may be feasible to harness energy from ambient sources, such as solar power, whereas in others such as climate monitoring in the canopies, sensor nodes may not be able to renew their energy resources. A major energy consumer is radio communication [4]. A comparison of the cost of computation to communication in future platforms by Pottie and Kaiser [4] reveals that 3000 instructions can be executed for the same cost as the transmission of one bit over 100m. Energy conservation in such networks involves two dominant approaches to minimize communication overhead. The first is at the MAC and networking layers, where nodes turn off their radios when they are not required for multihop communication (*adaptive duty cycling*). The second is data reduction through *in-network processing* (also called data aggregation), whereby correlations in data are exploited [5, 6] to reduce the size of data, and correspondingly communication cost.

The large number of nodes expected in sensor network deployments, and the unpredictable nature of deployment conditions introduce significant scalability and reliability concerns as well. Increased levels of system dynamics can be expected, including permanent or intermittent node failures. These dynamics are compounded by the vagaries of the wireless channel, introducing environmental dynamics. Designing long-lived and unattended systems under such conditions implies that the system itself must carry out the measurement and adaptive configuration in an energy constrained fashion.

Techniques to tackle these challenges fall into four broad categories:

Limiting radio operation

The dominant use of radios is for multi-hop communications, usually involving nodes sending data to one or more data sinks, or base-station. An ideal power conservation policy would switch radios off when a node is not required either as a data source or relay in such multi-hop routing. Powering down radios on sensor nodes, however, renders such nodes unavailable for multi-hop communication. A naive policy that shuts down radios on sensor nodes when they are not in use can cause regions of the network to be inaccessible by queries, due to routing partitions, especially in the face of environmental dynamics. How can sensor systems maintain communication backbones, and enable multi-hop routing while powering down radios for power conservation? Two techniques that have been explored to solve this problem are

- *Adaptive duty-cycling* (S-MAC [7, 8], ASCENT [9], SPAN [10]): Here, the set of nodes whose radios are powered down are carefully chosen such that a network backbone is continually maintained, while non-backbone nodes can put their radios to sleep. To load-balance, and thus prevent nodes from dying, the active subset of nodes are adaptively cycled, based on parameters such as available energy, radio coverage, etc.
- *Wakeup on demand* (STEM [11], Wake-on-Wireless [12]): This technique uses nodes with multiple radios, a low-power radio (such as a mote radio [13]) that is used exclusively to wake up the high power radio (such as 802.11) when the need arises, much like a paging channel in cellular networks. Such a technique is especially useful in ad-hoc networks or in sensor networks with image data, where bandwidths and data-rates are higher, and warrants the use of multiple radios.

Data Management

Sensor networks are intended to collect and actuate on data about the physical world, hence their use is expected to be highly data-centric. Unlike traditional end-to-end networking techniques, routing and data management need to be performed jointly in sensor networks in order to maximize energy savings. Therefore, a significant networking component is to provide a flexible platform to build data management frameworks that use various application-specific data aggregation schemes.

A key aspect of data management is data naming. While many attribute-based naming mechanisms have been proposed over an IP framework for the internet (eg: [14, 15, 16]), sensor networks pose unique challenges due to their data and resource constraints.

Foremost among these is the disparity between the amount of data generated by sensors and the amount of data that a network can communicate before depleting limited energy resources. For example, in an environmental monitoring application such as [17], a node has multiple weather sensors, each generating samples once every few seconds. If each mote sensor node transmitted all data, its lifetime would be limited to a few weeks, as opposed to a target lifetime of many years. Two factors can be exploited to reduce the amount of data communicated: (a) not all data is necessary for users, hence only interesting event detections or user-required data need to be selectively communicated, and (b) in a dense sensor network, significant correlation in data can be expected, and can be exploited to reduce the size of data. An important aspect of data management in sensor network is support for such *in-network* data reduction techniques.

The second challenge is resource constraints on embedded sensor nodes. The capabilities of these devices can range from Ipaq-class devices, to highly constrained mica motes [13]. Traditional database methods are resource-intensive, whereas data management schemes for sensor networks need to function on resource-limited nodes.

Geographic Routing in Sensor Networks

Routing in sensor networks differs from routing in both ad-hoc wireless networks and the Internet in two ways: The first is that it is attribute-based and often includes geographical location. The second is that energy constraints, network dynamics and deployment scale preclude proactive or global schemes for routing. Routing schemes that operate primarily on *local information* are more appropriate, since these can be reactive to local changes, while not requiring energy expensive global transfers of routing tables. Such reactive approaches are potentially more energy efficient as well, since sensor networks are expected to have bursty traffic.

We describe different approaches to such routing schemes in the context of sensor networks. Early Diffusion schemes [5], and TinyDB construct opportunistic routing trees to route data to the sink or base-station. Recent research in localization schemes have, however, made it likely that cheap and precise location information can be obtained even in some networks where all nodes cannot be GPS enabled. This development indicates that a dominant form of routing in sensor networks is likely to be geographically driven, since location attribute can reduce communication cost in the following ways:

- Sensor data is likely to be geographically correlated. Data reduction or aggregation schemes would need to route geographically to exploit such correlations (eg: Dimensions [6]).
- Queries that are geographically scoped are likely in many applications where users would prefer to query a small geographical region rather than the entire network. For instance, in a tracking application, the query is efficiently answered by querying only nodes on the trajectory of the target rather than all nodes in the network. Similarly, weather monitoring that is targeted at understanding local characteristics of data rather than global ones can be handled efficiently using geographically scoped queries.

A significant challenge results from non-uniform sensor node deployments that are likely due to deployment terrain, requiring that routing protocols be augumented with the capability to route around obstacles.

System Monitoring and Maintenance

System monitoring is a challenging problem in most distributed systems, where the ratio of number of devices to the number of personnel to maintain them is large. A suite of tools that ease the maintenance overhead is a key factor in the usefulness of such systems. Most typical systems, from operating systems on machines that we use everyday, to servers and routers, use some kind of logging facility to keep track of system state. In distributed systems, these logs are often transferred over the web to places where they can be maintained persistently. This log information can be remotely analyzed to monitor security problems, or system failures. Such monitoring often assumes that maintaining logs is cheap, and bandwidth inexpensive. While such an assumption is certainly true of today's Internet, energy constraints on sensor networks render it impossible to communicate extensive logs from nodes in the network to a central location. More discriminating methods of carefully selecting the data to be sent, and in-network aggregation techniques to reduce the debugging data need to be designed.

Such network monitoring and management schemes requires a suite of tools for networks of different scales, ranging from laboratory-scale networks comprised of a handful of nodes to operational networks comprising hundreds of nodes.

Case Study: Distributed Networked Storage and Feature Extraction

To see how these various networking primitives can be brought together to construct a flexible, yet efficient data-processing application, we consider the example of a networked storage infrastructure, Dimensions [18]. This system creates multi-resolution summaries of data, and moves these summaries around the network to facilitate both spatio-temporal feature extraction and long-term data storage. Such an infrastructure uses many of the above-mentioned networking primitives,

- Duty-cycling of the radio while supporting multi-hop communication can be done by using MAC-level techniques such as SMAC.
- Attribute-based naming is used to express queries on spatio-temporal attributes of data.
- Geographic Routing is used to route data to specific locations in the network for long-term storage.
- Distributed monitoring schemes can assist long-term maintenance under varying environmental conditions.

2 Adaptive Duty Cycling

This section describes techniques that enables low-duty-cycle operations of sensor nodes, which are critical for prolonging network lifetime.

2.1 Overview

Wireless sensor networks are designed to operate for long time. However, nodes are in idle state for most time when no sensing event happens. It would be a significant waste of energy if all nodes always keep their radios on, since the radio is a major energy consumer. Measurements have shown that a typical radio consumes the similar level of energy in idle mode as in receiving mode [19, 20]. It is important that nodes are able to operate in low duty cycles.

Current research on adaptive duty cycling can be broadly divided into three catagories: general schemes of sleep and wake-up, low duty cycle combined with MAC protocols, and duty-cycle control through topology management.

Research on general schemes of sleep and wake-up focuses on how to set up the communications between two nodes that are normally in sleep mode. These schemes are not tightly coupled with other protocols such as MAC and topology control. Piconet [21] is such an example. In Piconet, each node randomly goes into sleep mode, and periodically wakes up for a short period time. Every time a node wakes up, it broadcasts a beacon including its own ID. If other nodes want to talk to this node, they need to wake up and listen until receiving the beacon.

Another example is STEM [11]. STEM uses two radios operating in different channels, one for data transmission and the other for node wake-up. When there are no data to send, nodes turn off their data radio completely, and put the wake-up radio in low-duty-cycle mode as in Piconet. Unlike Piconet, in STEM a sender is responsible for waking up the receiver. It does so by sending a wake-up tone (STEM-T) or beacon (STEM-B). Since nodes do not synchronize on their wake-up time, the tone or beacon must be long enough for the intended receiver to receive it.

The second catagory is MAC protocols with low duty cycles. This is a broad research area, and can be further divided as TDMA protocols and contention protocols.

TDMA-based MACs naturally enable low-duty-cycle operations on nodes, since they only need to turn on their radio during their own time slots for sending and receiving. Typical examples include Bluetooth [22] and LEACH [23]. In both protocols, nodes form clusters, and TDMA is used for intra-cluster communications. The major disadvantage of TDMA protocols is scalability, *i.e.*,, it is difficult to dynamically change the frame size or the number of slots when the number of nodes changes. For example, Bluetooth may have at most 8 active nodes in a cluster.

Sohrabi and Pottie [24] proposed a self-organization protocol for wireless sensor networks. Each node maintains a TDMA-like frame, in which the node schedules different time slots to communicate with its known neighbors. Nodes go to sleep during the time that is not scheduled.

Among contention-based MACs, the IEEE 802.11 distributed coordination function (DCF) [25] is widely used in ad hoc networks. It has a power save (PS) mode, which enables low-duty-cycle operations. In PS mode, nodes periodically sleep and wake up, and sychronize on their wake-up time. 802.11 assumes all nodes are within one hop. In multi-hop operation, the PS mode may have problems in clock synchronization, neighbor discovery and network partitioning, as pointed out in [26]. Tseng *et al.* [26] proposed three sleep schemes to improve the 802.11 PS mode. None of them requires node synchronizations. The cost is more frequent beaconing and more wake-up packets before broadcasts.

S-MAC [7, 8] is a contention MAC with integrated low-duty-cycle operation that supports multi-hop operation. More details of S-MAC are described in Section 2.2.

The last catagory of adaptive duty cycling is achieved through topology control. Protocols in this catagory explore benefits that a dense network provides for energy savings. The basic idea is to only power on a small number of nodes that are sufficient to maintain network connectivity. Examples include GAF [27], SPAN [10] and ASCENT [9].

GAF utilizes geographic location information, and divides the network into fixed square grids. Within each grid, nodes are equivalent from the routing point of view, so only one node needs to be active at any given time. In SPAN, each node decides whether to sleep or join the backbone based on connectivity information supplied by a routing protocol. In ASCENT, each node makes the decision only based on locally measured packet loss and connectivity information. More details of ASCENT is provided in Section 2.3

2.2 S-MAC

S-MAC [7, 8] explores design trade-offs for energy-conservation in the MAC layer. It reduces energy consumption on radio from the following sources: collision, control overhead, overhearing unnecessary traffic, and idle listening. Idle listening is the idle state that the radio keeps listening for possible traffic.

The basic scheme of S-MAC is to put all nodes into low-duty-cycle mode — periodic listen and sleep. When nodes are listening they follow a contention rule to

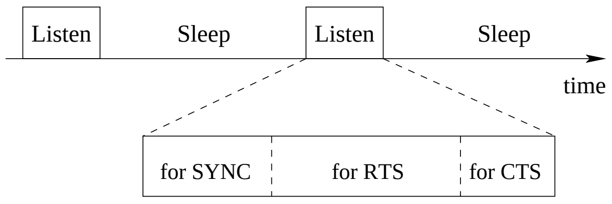


Fig. 1. Low-duty-cycle operation of each node in S-MAC.

access the medium, which is similar to the IEEE 802.11 DCF [25]. The following discussion focuses on the adaptive duty cycling in S-MAC.

Coordinated Sleeping

In S-MAC, nodes exchange and coordinate on their sleep schedules rather than randomly sleep on their own. Before each node starts the periodic sleep, it needs to choose a schedule and broadcast it to its neighbors. To prevent long-term clock drift, each node periodically broadcasts its schedule as the SYNC packet. To reduce control overhead and simplify broadcasting S-MAC encourages neighboring nodes to choose the same schedule, but it is not a requirement. A node first listens for a fixed amount of time, which is at least the period for sending a SYNC packet. If it receives a SYNC packet from any neighbor it will follow that schedule by setting its own schedule to be the same. Otherwise, the node will choose an independent schedule after the initial listen period.

It is possible that two neighboring nodes have two different schedules. If they are aware of each other's schedules, they have two options: 1) following two schedules by listening at both scheduled listen time; 2) only following its own schedule, but sending twice to both schedules when broadcasting a packet. In some cases the two nodes may not be aware of the existence of each other, if their listen intervals do not overlap at all. To solve the problem, S-MAC let each node periodically perform neighbor discovery, *i.e.,*, listening for the entire SYNC period, to find unknown neighbors on a different schedule.

Figure 1 depicts the low-duty-cycle operation of each node. The listen interval is divided into two parts for both SYNC and data packets. There is a contention window for randomized carrier sense time before sending each SYNC or data (RTS or broadcast) packet. For example, it node A wants to send a unicast packet to node B, it first perform carrier sense during B's listen time for data. If carrier sense indicates an idle channel, node A will send RTS to node B, and B will reply with a CTS if it is ready to receive data. After that, they will use the normal sleep time to transmit and receive actual data packets. Broadcast does not use RTS/CTS due to the potential collisions on multiple CTS replies.

Low-duty-cycle operation reduces energy consumption at the cost of increased latency, since a node can only start sending when the intended receiver is listening. S-MAC developed an adaptive listen scheme to reduce the latency in a multi-hop

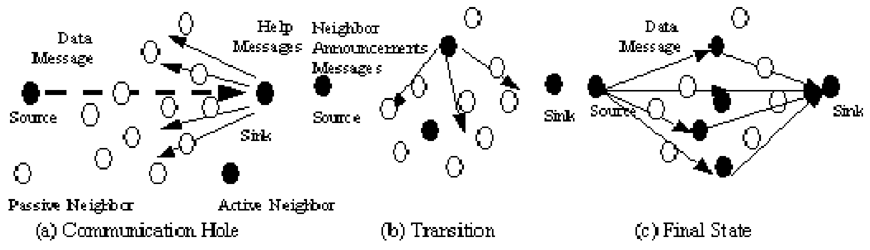


Fig. 2. Self-confi gation of a sensor network using ASCENT.

transmission. The basic idea is to let the node who overhears its neighbor's transmissions (ideally only RTS or CTS) wake up for a short period of time at the end of the transmission. In this way, if the node is the next-hop node, its neighbor is able to immediately pass the data to it instead of waiting for its scheduled listen time. If the node does not receive anything during the adaptive listening, it will go back to sleep.

Trade-offs on Energy, Latency and Throughput

With low-duty-cycle operation S-MAC effectively reduces the energy waste due to idle listening. Experimental results show that an 802.11-like protocol without sleeping consumes 2–6 times more energy than S-MAC for traffic load with messages sent every 1–10s in a 10-hop network. On the other hand, S-MAC with adaptive listen has about twice the latency as the MAC without sleeping. Periodic sleeping increases latency and reduces throughput. However, adaptive listening largely reduces such cost. It enables each node to adaptively switch mode according to the traffic in the network. The overall gain on energy savings is much larger than the performance loss on latency and throughput [8].

2.3 ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies

To motivate the need for ASCENT, consider a habitat monitoring sensor network that is to be deployed in a remote forest. Deployment of this network can be done, for example, by dropping a large number of sensor nodes from a plane, or placing them by hand. Besides the fundamental theme of energy conservation, such deployments introduce scaling challenges:

- Ad-hoc deployment: The sensor field cannot be expected to be deployed in a regular fashion (e.g. a linear array, 2-dimensional lattice). More importantly, uniform deployment need not correspond to uniform connectivity owing to unpredictable propagation effects when nodes, and therefore antennae, are close to the ground and other surfaces.
- Unattended operation under dynamics: the anticipated number of elements in these systems will preclude manual configuration, and the environmental dy-

namics will preclude design-time pre-configuration.

In many such contexts it will be far easier to deploy larger numbers of nodes initially than to deploy additional nodes or additional energy reserves at a later date (similar to the economics of stringing cable for wired networks). ASCENT describes one approach towards exploiting the resulting redundancy in order to extend system lifetime. If too few of the deployed nodes are used, the distance between neighboring nodes will be too great and the packet loss rate will increase; or the energy required to transmit the data over the longer distances will be prohibitive. If all deployed nodes are used simultaneously, the system will be expending unnecessary energy, at best, and at worst the nodes may interfere with one another by congesting the channel. ASCENT describes a localized, energy efficient, and adaptive procedure for addressing this problem.

The two primary contributions of the design of ASCENT are:

- The use of adaptive techniques that permit applications to configure the underlying topology based on their needs while trying to save energy to extend network lifetime. Results presented in [9] show that ASCENT always improves network lifetime (defined as time till 90% of network runs out of power), and for large densities is a factor of 3 better than a network not running the protocol.
- The use of self-configuring techniques that react to operating conditions *measured locally*. ASCENT is designed to react when links experience high packet loss. Results show that a stable packet delivery ratio ($\approx$100%) is maintained, that is between 10-20% better than a setup without ASCENT. It does not detect or repair network partitions and complementary techniques may be required to address such problems.

ASCENT Design

ASCENT adaptively elects ”active” nodes from all nodes in the network. Active nodes stay awake all the time and perform multi-hop packet routing, while the rest of the nodes remain ”passive” and periodically check if they should become active. Consider a simple sensor network for data gathering. Such a sensor field cannot be expected to have uniform connectivity due to unpredictable propagation effects in the environment. Therefore, regions with low and high density can be expected. ASCENT does not deal with complete network partitions; it assumes that there is a high enough node density to connect the entire region. Figure 2 shows a simplified schematic for ASCENT during initialization in a high-density region.

Initially, only some nodes are active. The other nodes remain passively listening to packets but not transmitting as shown in Figure 2(a). The source starts transmitting data packets toward the sink. Because the sink is at the limit of radio range, it gets very high message loss from the source. This situation is called a communication hole; the receiver gets high packet loss due to poor connectivity with the sender. The sink then starts sending help messages to signal neighbors that are in listen-only mode -also called passive neighbors- to join the network. When a neighbor receives a help message, it may decide to join the network. This situation is illustrated in Figure 2(b). When a node joins the network it starts transmitting and receiving packets, *i.e.*,it becomes an active neighbor. As soon as a node decides to join the network, it signals the existence of a new active neighbor to other passive neighbors by sending a neighbor announcement message. This situation continues until the number of active nodes stabilizes on a certain value and the cycle stops. When the process completes, the group of newly active neighbors that have joined the network make the delivery of data from source to sink more reliable. The process re-starts when some future network event (e.g. node failure) or environmental effect (e.g. new obstacle) causes message loss again.

3 Data Management

This section describes schemes that address these challenges to manage data. We will focus on architectural frameworks for data management, rather than specific aggregation schemes that fit into one or more of these frameworks.

3.1 Overview

Data management approaches to sensor networks have broadly followed similar themes, although their specific approaches have varied widely. Table 3 shows three popular techniques, Directed Diffusion, Cougar and TinyDB. These schemes are based on two primary architectural principles:

- Use a declarative language to specify queries on data: A declarative language can be especially useful to describe sensor network interaction since it hides details of sensor node interaction, routing and placement of in-network processing from users.
- Support in-network processing to reduce data within the network: Since local computation is much cheaper than radio communication [4], shifting computation into the network can provide significant energy savings.

While all three schemes are based on the above principles, there are significant differences in their approach. Directed Diffusion is a library-based lower-level approach, and provides Filters [28] to ease application development. The Filter API exposes the underlying routing structure to an application designer and can be used to write algorithms that optimize communication and in-network processing to application requirements. We describe this approach in greater detail in Section 3.2.

Both Cougar and TinyDB use a database approach towards sensor data management.

The Cougar device database system proposes distributing database queries across a sensor network as opposed to moving all data to a central site [29]. Sensor data is represented as an Abstract Data Type attribute, the public interface to which corresponds to specific signal processing functions supported by a sensor type. Joins or aggregation in the network are performed as specified by a centrally computed query plan. As shown in Table 3, queries are declarative, and users can issue queries without knowing how the data is generated in the sensor network and how the data is processed to compute the query answer. An underlying query optimizer decides on the placement of in-network processing, and is performed by a gateway node that has an idea of the status of nodes and links in the network. Such a centralized scheme can be inefficient if the status of nodes is rapidly changing (eg: environmental dynamics), but can enable a potentially more efficient global optimization of in-network query processing.

TinyDB is a query processing system with small footprint intended for highly resource-constrained mote sensor nodes [13]. TinyDB provides a declarative interface for data collection and aggregation inspired by selection and aggregation facilites in database query languages. TinyDB's aggregation service, Tag (Tiny Aggregation [30]), operates on a routing tree structure, where the root is typically a base-station or other egress point to users, and leaves of the tree consititute all nodes in the network.

An instance of an aggregation query specified using Tag is shown below.

```
SELECT AVG(volume), room FROM sensors
WHERE floor = 6
GROUP BY room
HAVING AVG(volume) > threshold
EPOCH DURATION 20s
```

In this example, SELECT specifies arithmetic operations over aggregation attributes. The WHERE clause specifies which sensor readings should be transmitted from each node, and which should be discarded. The GROUP BY operator can be used to partition data into groups such that the aggregation operators can be applied separately to each group. In the above example, sensor readings are grouped by the room, and the average over each room computed. Periodicity of this query is specified by the EPOCH clause.

Two other internet-centric approaches bear mention due to their relation to naming sensor data.

The Intentional Naming System is an attribute-based name system operating in an overlay network over the Internet [15]. Its use of attributes as a structuring mecha-

Scheme	Type	Platform	Query Specifi cation Language	Placement of In-network Processing	Paradigm	Supported Routing Strategies
Directed Diffu-sion	Non-database	Ipaq-class (TinyDif-fusion for mote-class nodes)	Application-specifi c language can be defi ned using Filters [28]	Distributed - Routes are independently constructed from each source of data to data sink. Aggregation is performed at junctions where data combines. Called Opportunistic Aggregation	Publish-subscribe - can support multiple data sources and sinks	Two-phase commit when location of nodes is not known and GEAR when location is known
Cougar	Database	Ipaq-class	SQL-like	Centralized - A gateway node (query optimizer) maintains a catalog of status of sensor nodes. For each new query, optimizer selects a plan that describes both datafbw in the network and computation fbw in the network.	Cluster-based	Not explicitly speci-fi ed, any ad-hoc routing protocol
TinyDB	Database	Mote-class	SQL-like	Opportunistic Aggregation - Routes are based on reverse shortest path from the base-station. Conceptually similar to route construction in Directed Diffusion	reverse multicast tree - all leaves are data sources	parent-child routing on tree

Fig. 3. Comparison of Data Management strategies for sensor networks

nism and a method to cope with dynamically locating devices is similar in spirit to Directed Diffusion.

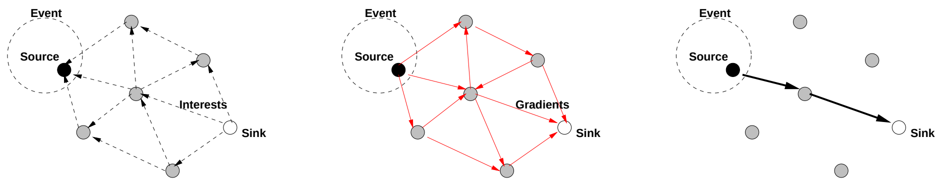
DataSpace describes an attribute based naming mechanism for querying physical objects that produce and store local data [16]. The DataSpace is divided into smaller administrative and logical *datacubes*, which are logically grouped into *dataflocks*. Query results may involve aggregation of more specific queries addressed to *sub-datacubes*. This approach is internet-centric and does not explore in-network processing.

3.2 Directed Diffusion

Directed Diffusion has three key characteristics: localized algorithms, named data, and support for in-network processing. Diffusion adopts a declarative, publish/subscribe API that isolates data producers and consumers from the details of the underlying data dissemination algorithms [31]. The key abstraction of this API is that data is identified by a set of attributes, data producers (or sources) generate data it by publishing, data consumers (or sinks) subscribe to data, and it is the business of the diffusion implementation to ensure that data travels from publisher to subscriber efficiently. Diffusion encourages applications to influence data flow through the use of filters [28] and in-network processing, but many applications require only attribute-selected data and allow diffusion to completely control routing. Many different algorithms can match publishers and subscribers without change to the high-level API or semantics.

Routing in Directed Diffusion

Diffusion uses a two-phase algorithm [31] where data consumers seek out data sources, and then sources search to find the best possible path back to subscribers. A subscriber, or data sink, identifies data by a set of attributes. This information propagates through the network in an interest message. In principle, information cached from prior runs, other constraints (such as geographic information), or application-specific filters can can be used to optimize the distribution of interests. Without



(a) Interest propagation

(b) Initial gradients set up

(c) Data delivery along reinforced path

Fig. 4. A simplified schematic for directed diffusion.

such information, however, interests must be flooded throughout the network to find any data sources (as shown in Figure 4(a)). As they are distributed, nodes establish gradients, state indicating the next-hop direction of other nodes interested in the data (Figure 4(b)). When an interest arrives at a data producer, that source begins producing data. (To conserve power, nodes may avoid producing data before being triggered, or they may produce and store such data locally.) The first data message sent from the source is marked as exploratory and is sent to all neighbors that have matching gradients. As with interest messages, this transfer could be limited using additional information or application involvement, but by default it is sent to all nodes. When exploratory data reaches the sink, the sink reinforces its preferred neighbor (Figure 4(c)), establishing a reinforced gradient towards the sink. (Preference being given to the lowest latency neighbor, possibly modified by other concerns such as link quality or energy.) The reinforced neighbor reinforces its neighbor in turn, all the way back to the data source or sources, resulting in a chain of reinforced gradients from all sources to all sinks. Subsequent data messages are not marked exploratory, and are sent only on reinforced gradients rather than to all neighbors. Nodes can also generate negative reinforcements if they receive data that is not relevant to them. Typically this occurs when topology changes and multiple gradients accidentally point to the same node. A negative reinforcement corrects this situation.

Gradients are managed as soft-state, thus both interests and exploratory data occur periodically to refresh this state. Interests are sent every interest interval, exploratory data every exploratory interval.

Variants of this basic model have been proposed to optimize Diffusion for different application requirements. For instance, a two-phase diffusion model may not be efficient when applications have many sources and sinks cross-subscribed to each other, a case that results in a large amount of control traffic, even with geographic scoping. Push Diffusion, introduced in [32], reverses the role of data publishers and subscribers, causing data sources to actively search for consumers. An advantage of push is that it requires only one phase where control traffic needs to be widely disseminated to find sinks, unlike the two phases needed in two-phase pull. One-phase pull [32] is a third diffusion algorithm that simplifies two-phase pull by eliminating one side of the search.

3.2.1 Aggregation using Filters

An example of filter-driven data aggregation using Directed Diffusion can be used to illustrate how in-network processing can reduce data traffic to conserve energy.

An anticipated sensor application is to query a field of sensors and then take some action when one or more of the sensors is activated. For example, a surveillance system could notify a biologist if an animal enters a region. Coverage of deployed sensors will overlap to ensure robust coverage, so one event will likely trigger multiple sensors. All sensors will report detection to the user, but communication and energy costs can be reduced if this data is aggregated as it returns to the user. Data can be aggregated to a binary value (there was a detection), an area (there was a detection in quadrant 2), or with some application-specific aggregation (seismic and infrared sensors indicate 80% chance of detection).

Although details of aggregation can be application-specific, the common systems problem is the design of mechanisms for establishing data dissemination paths to the sensors within the region, and for aggregating responses. Consider how this kind of data fusion may be implemented in a traditional network with topologically-assigned low-level node names. First, in order to determine which sensors are present in a given region, a binding service must exist which, given a geographical region, lists the node identifiers of sensors within that region. Once these sensors are tasked, an election algorithm must dynamically elect one or more network nodes to aggregate the data and return the result to the querier.

Instead, Directed Diffusion ([31]) addresses this problem using *opportunistic* data aggregation. Sensor selection and tasking is achieved by naming nodes using geographic attributes. As data is sent from the sensors to the querier, intermediate sensors in the return path identify and cache relevant data. This is achieved by running application-specific filters. These intermediate nodes can then suppress duplicate data by simply not propagating it, or they may slightly delay and aggregate data from multiple sources.

Filters benefit opportunistic aggregation strategies in many ways. It provides a natural approach to inject application-specific code into the network. Attribute naming and matching allow these filters to remain inactive until triggered by relevant data. A common attribute set means that filters incur no network costs to interact with directory or mapping services.

More complex examples of in-network aggregation using filters are also discussed in [31]. Nested queries, where one sensor cues another, can be used for triggering, and used to reduce overall energy consumption significantly. For example, a person entering a room is often correlated with changes in light or motion, Multi-modal sensor networks can use these correlations by triggering a secondary sensor based on the status of another, in effect nesting one query inside another. Reducing the duty cycle of some sensors can reduce overall energy consumption (if the secondary sensor consumes more energy than the initial sensor, for example as an accelerometer triggering a GPS receiver) and network traffic (for example, a triggered imager generates much less traffic than a constant video stream). Alternatively, in-network processing might choose the best application of a sparse resource (for example, a motion sensor triggering a steerable camera).

4 Geographic Routing

The physical nature of a sensor network's deployment makes geographically scoped queries natural. If nodes know their locations, then geographic queries can be leveraged to constrict the data dissemination to the relevant region and to reduce routing control overhead. Geographic Routing protocols such as GPSR (Karp et al [33]) or GEAR (Yu et al [34]) can be used to optimize the process of finding sources by using geographic information to constrain the search process. Since geographical routing does not have to be tied to any particular data dissemination scheme, we describe these protocols separate from our discussion of data management schemes. However, these routing protocols can be integrated quite easily into the attribute-naming frameworks where location is considered an attribute.

4.1 Overview

Most geographical routing protocols use greedy forwarding to route packets to the destination. They differ from each other in how they handle communication holes. Among early work in geographical routing, Finn [35] used a restricted flooding search to navigate around holes [1] . One drawback of this mechanism is the difficulty in determining an appropriate scope for the search.

Greedy Perimeter Stateless Routing (GPSR [33]), elegantly avoids this problem by deriving a planar graph out of the original network graph. When a packet reaches a region where greedy forwarding is impossible, GPSR recovers by forwarding the packet along the perimeter of the planar graph to circumvent communication holes. We discuss this scheme in more detail in Section 4.2.

While GPSR addresses point-to-point routing, GEAR (Geographic and Energy-Aware Routing) [34] studies the problem of forwarding a packet to all the nodes

[1] a **communication hole** is when greedy forwarding reaches a local maximum where the current node is closer to the destination than any of its neighbors.

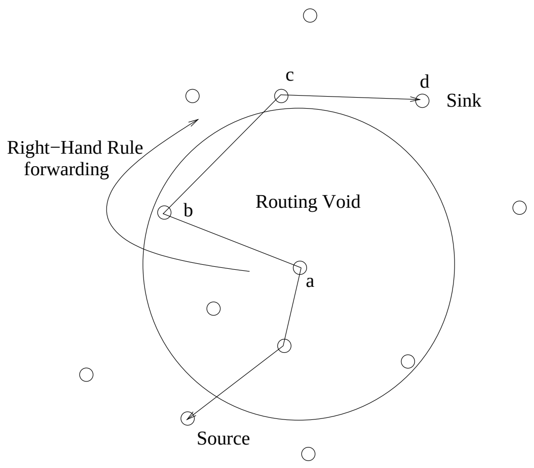


Fig. 5. Greedy and Perimeter Forwarding in GPSR: Greedy mode is used till packet reaches a routing void. Perimeter forwarding uses graph planarization techniques to route around voids.The well-known right hand rule traverses the interior of a closed polygonal region in clock-wise edge order, in this case, $a- > b- > c- > d$

inside a target region, which is a common primitive in data-centric sensor network applications [5]. The protocol has two distinctive features:

- It uses an energy-aware neighbor selection heuristic to forward packets. Nodes with more remaining energy are preferred over depleted nodes to avoid nodes that
- It proposes a Recursive Geographic Forwarding algorithm to dissemminte packets within a target region. Systematic unicast is used to deliver packets to the region as opposed to broadcast, thus limiting the receive power expended by nodes.

GEAR (Geographic and Energy-Aware Routing) has been used to extend Directed Diffusion when node locations and geographic queries are present [34].

4.2 Greedy Perimeter Stateless Routing (GPSR)

GPSR offers two benefits. First, it is stateless, and requires propagation of topology information only within a single hop neighborhood. Such a localized algorithm scales better in dynamic networks, where a global algorithm would need to track each topology change or risk having inaccurate information. Second, when a packet reaches communication holes, GPSR uses a distributed perimeter forwarding algorithm that routes around such obstacles.

There are two main components to GPSR:

Greedy Forwarding

A forwarding node makes a locally optimal, greedy choice for the next hop by choosing the neighbor that is geographically closest to the packets destination (as shown in Figure 5). Such an approach has negligible state overhead that depends on the density of deployment, and not on the total network size.

However, greedy forwarding will not work in topologies such as Figure 5, where to reach the destination, the packet would need to be routed farther in geometric distance from the destination.

Perimeter Forwarding

GPSR explores two distributed algorithms to construct planar graphs, the Relative Neighborhood Graph(RNG) and Gabriel Graph (GG), to enable perimeter forwarding. These procedures involve selecting a subset of edges in the network such that there are no crossing edges in the resulting graph. Given such a sub-graph, perimeter forwarding can be done in a straghtforward manner using the Right Hand traversal rule. A detailed treatment of the planarization procedure can be obtained from [33].

The complete GPSR algorithm combines greedy forwarding on the full network topology, with the perimeter forwarding on the subgraph when greedy forwarding becomes impossible.

By keeping state only about local topology and using it in the packet forwarding, GPSR scales better than other ad-hoc routing protocols, However, the derived planar graph in GPSR is much sparser than the original one, and the traffic concentrates on the perimeter of the planar graph in perimeter mode. Thus, the nodes on the planar graph tend to be depleted quickly.

5 Sensor Network Monitoring and Maintenance

As in most distributed systems, it is important to have an infrastructure for wireless sensor networks to provide indication of node failures, resource depletion, and overall system performance. More than for wired networks and ad-hoc wireless networks, sensor networks need a comprehensive system for understanding the execution of the network as a whole. There are several reasons for this: sensor networks implement intricate distributed and collaborative applications; sensor nodes use wireless communication the vagaries of which are well documented; nodes detect physical events in the environment, and the variability in the physical phenomena they sense at least rivals that of wireless communication. However, existing distributed monitoring and management protocols are designed for multi-computer distributed systems connected with wired networks. For example, SNMP [36] provides per-device information databases for network monitoring and management, where node state and logs can be collected and processed. However, the energy and bandwidth constraints of wireless sensor networks preclude such centralized solution. In addition, collaboration between redundant sensor nodes imposes challenges in the definition of system failure and status. One or few node crashes do not necessarily undermine system functionality. A regional or global view of the system is helpful to identify possibility of failures.

5.1 Overview

Several protocols are proposed recently to address different important aspects of system monitoring and maintenance for sensor networks. To meet the unique design challenges in wireless sensor networks, explicitly or implicitly, those proposals share two common design principles: localized network state exchange and in-network aggregation.

For example, the coverage problem in wireless sensor networks is studied in [37, 38]. By exchanging sensor coverage information within a local neighborhood, these techniques detect maximal breach path and maximal support path, along which there is poorest and best coverage of sensors, respectively. In [39, 40], node failure information is collected using similar techniques. By limiting state exchange of nodes to only their local vicinity, these protocols successfully monitor collective network states without communicating individual node state over long distance.

eScan [41] takes advantage of in-network aggregation to construct an aggregated map of the remaining energy levels for different regions in a sensor field. Instead of extracting individual node states, the protocol combines residual energy level information into a more compact form if those nodes are *nearby* and have *similar* energy levels. To accomplish a similar task, [42] applies a network state model to represent node behavior. By probabilistically predicting the energy consumption, it reduces the communication overhead of reporting energy levels to the monitoring node. Dimensions [18] proposes the use of multi-resolution wavelet processing to compress debugging data such as network-wide packet-loss statistics. This scheme has two properties: (a) correlations in data over time and spatial dimensions can be exploited, and (b) the scheme provides approximate, but sufficiently accurate responses to many queries with low communication overhead.

Some protocols are designed to self-regulate monitoring activity when network workload is heavy or at least provide a “knob” for adaptive aggregation. In *eScan* [41], resolution (how close nodes are geographically nearby) and tolerance (how similar their energy levels are) can control the extent of aggregation. In STREAM

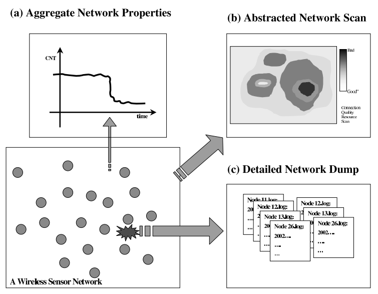

(a) Aggregate Network Properties
CNT
time
(b) Abstracted Network Scan
(c) Detailed Network Dump
A Wireless Sensor Network

Fig. 6. Monitoring wireless sensor networks

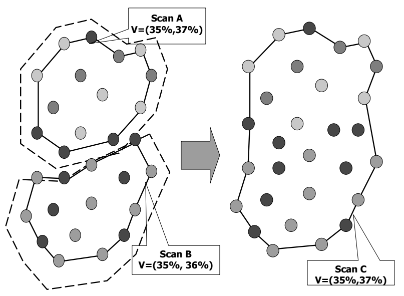


Fig. 7. Representation and Aggregation of eScans.

[43], topology mapping of sensor networks can be collected at different levels of detail, which is decided by the parameters of “virtual range” and “resolution factor”.

Note that none of techniques above are specific to sensor network monitoring and management. They apply in general to most scenarios of sensor network application design and implementation. However, monitoring protocols have unique requirements: given that monitoring tends to be a long-term activity on large networks, the energy constraints are more strict on those protocols. Furthermore, those protocols are supposed to be even more robust than generic applications because they might be the last resort when a massive failure happens.

5.2 A Monitoring Architecture for Wireless Sensor Networks

In this section, *eScan*[41] and *digest*[44] are illustrated as two examples of sensor network monitoring solutions. These tools can be combined into a coherent architecture for monitoring sensor networks [44]. This architecture is distinguished by three levels of monitoring, where each level consists of a class of tools. Each level is distinguished from the next in the spatial or temporal scale at which the corresponding tools are invoked. This is illustrated in Figure 6.

The first component consists of tools such as *dump*, which can seen as more classical SNMP like component in this architecture. Upon user request, *dump* collects detailed node state or logs over the network for diagnosis. For example, the raw temperature readings from some sensors could be dumped to debug the collaborative event detection algorithm between nearby nodes. *Dump* can be implemented as an generic application over Directed Diffusion[5] or other data management frameworks . Because the amount of data per node may be large, *dump* should be invoked only at small spatial scales (*i.e.*, from a few nodes), and only when there is a reasonable certainty of a problem at those nodes.

A second class of tools, *scans*, is envisoned to guide system administrators to the location of problems. Scans represent abstracted views of resource consumption throughout the entire network, or throughout a significant section of the network. Thus, this class of tools has a significantly greater spatial extent than dumps. One example of a scan is the *eScan* [41]. To compute an *eScan*, a special user-gateway node initiates collection of node state, for instance residual energy supply level, from every node in the system. Instead of delivering the raw data to user node, *eScan* computation takes advantage of in-network aggregation. Residual energy level data from individual nodes are combined into more compact forms, if and only if those nodes are nearby and have similar energy level. By pushing the data processing into the network, *eScan* constructs an approximate system-wide view of energy supply levels with much less communication cost as compared to centralized collection. From such a global view, users are able to isolate those nodes upon which they can invoke tools such as *dump*. Simulation studies in [41] reveal that good aggregation benefit ($\approx$15% data reduction) is obtained with low distortion ($\approx$5%) in the energy scans collected from a 400 node randomly placed network

Clearly, the energy cost of collecting an eScan can be significant, and a third class of tools, *digests*[44], can help alert users to error conditions (partitions, node deaths) within the network. A digest is an aggregate of some network property in small size (say a few bytes). For example, the size of network i.e. the number of nodes, can indicate several system health conditions: Sudden drop in the network size can be taken as hint for massive node failure or network partitioning. Digests, like eScans, also span the entire network, or a large spatial extent. However, unlike eScans, they are continuously computed. Digests are not intended to isolate network problems, merely to tell users when to invoke network-wide scans.

6 Case Study: Building a Distributed Storage Framework

We consider the application of the networking schemes described above in the context of a distributed data storage and querying framework, Dimensions [6, 18]. This system is targeted at long-term scientific deployments, such as micro-climate monitoring [17, 45], to obtain data about previously unobservable phenomena for detailed analysis by experts in the field.

Consider the following sample deployment: *A medium scale wireless sensor network (many hundreds of nodes) is deployed in parts of a reserve park such as the James Reserve, for monitoring climatic variations at finer granularities, and larger scales than previously possible. These nodes are equipped with weather sensors: temperature, pressure, humidity and rainfall. These sensor networks can be queried through a gateway (or base-station) connected to the internet. User queries are typically spatio-temporal, including simple min-max queries, more complex queries searching for edge patterns, long-term data trends, anomalous behavior, frequency of events, etc. For the small subset of nodes that satisfy these queries, the user may decide to obtain more detailed datasets for further off-line analysis. This could*

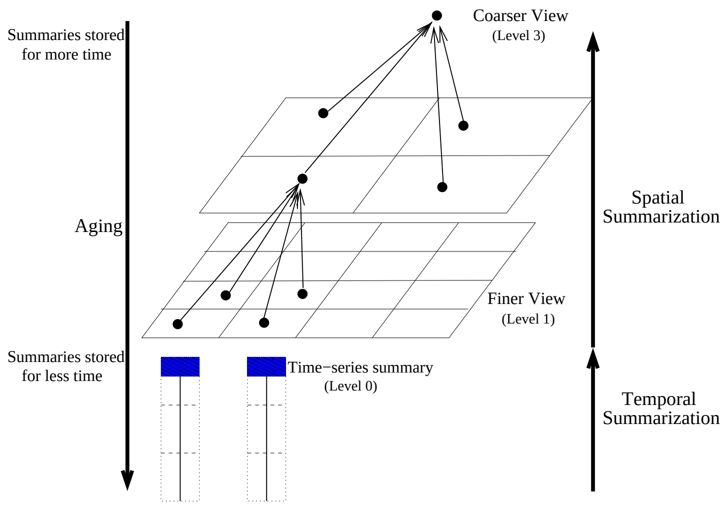


Fig. 8. Constructing a Dimensions Hierarchy: Temporal and Spatial Summarization

be requested in a multi-resolution manner, first obtaining a low-resolution dataset which can be used to "decide" that a higher resolution, possibly lossless, dataset needs to be extracted.

Data analysis in such applications often involves complex signal manipulation, including modeling, searching for new patterns or trends, looking for correlation structures, etc. Conventional approaches to such monitoring has involved wired and sparsely deployed networks that transfer all data from sensors to a central data repository for persistent storage. The consequence of limited energy and storage resources of sensor nodes introduce two problems: (a) it severely limits deployment lifetime if all raw data must be transmitted to a central location, [46]), and (b) the storage resources on individual sensor nodes are often not sufficient to store data over long durations losslessly. We provide a high-level description of the methodology used in Dimensions to tackle these problems, and then discuss different ways in which networking techniques fit into such a system.

6.1 Dimensions Architectural Overview

Dimensions [6] constructs progressively lossy hierarchy of wavelet summaries of sensor data and distributes them around the network. These summaries are generated in a multi-resolution manner, i.e. there are equal-sized summaries corresponding to different spatial and temporal scales. Figure 8 illustrates their construction: at each higher level of the hierarchy, summaries encompass larger spatial scales, but are compressed more, and therefore more lossy. At the highest level, one or a few nodes have a very lossy summary for all data in the network. Such a multi-resolution approach results in a per-node worst case communication cost of $O(\log n)$ as opposed to $O(\sqrt{n})$ for centralized data collection, where n is the number of nodes in the network.

Queries into the network are routed in the reverse order of building the summaries. A query is injected into the network at the highest level of the hierarchy, and is processed on a coarse highly compressed summary corresponding to a large spatio-temporal volume. The result of this processing is an approximate result that indicates which regions in the network are most likely to provide a more accurate response to the query. The query is forwarded to summaries for these regions of the network, and processed on more detailed views of these sub-regions. This procedure continues until the query is routed to a few nodes at the lowest level of the hierarchy. In this way, query processing cost can be dramatically reduced when the target data is sparsely distributed. The authors show that for the studied set of query types (Max, Min, Avg, Edge), Dimensions can satisfy queries to within $\approx$10% error by querying less than 5% of a 160 node network.

Long-term storage in Dimensions exploits the progressively lossy storage model. Summaries are aged such that those corresponding to larger spatial areas and longer time-scales are retained for longer periods of time. Thus, even though raw sensor data and high resolution summaries may be aged early, coarse summaries are retained about past data and can be used for long-term querying, at the cost of slightly reduced accuracy.

6.2 Networking Schemes in Dimensions

A system such as Dimensions has numerous networking components. Significant amount of data is transferred around the network in the form of summaries at different spatio-temporal resolutions. Drill-down queries need to be expressed in a query language, and involve in-network computation at different levels of the hierarchy. In this section, we discuss how these components can be constructed using schemes that we have discussed.

Adaptive Duty Cycling

As described in Section 2, multihop routing requires that radios be turned on, whereas nodes would need to switch them off for power conservation. How would an adaptive duty cycling strategy fit with such a scheme, which requires significant data transfer? To understand the coupling between the two schemes, we look at the routing requirements in more detail. Routing is required for two purposes in Dimensions: (a) to transfer data between clusterheads at different levels, and (b) to route queries in a drill-down manner. In the first case, latency is not a concern, since this is a background, slow-running process. Thus, a scheme like STEM [11] would not be necessary. Further, the periodicity of communication is fixed, since summaries are generated every epoch. These requirements lend the problem to be particularly well-suited for a scheduling-based protocol such as S-MAC [7]. The advantages of using SMAC for long messages can be exploited to send the summaries efficiently.

Routing queries does not require large data transfers, but have two features: (a) they can be latency-limited, in the case of an interactive user session, and (b) queries may arrive at random times, and the user could potentially query the network from any location within it. For the first case, an on-demand wakeup scheme such as STEM can be used if multiple radios are available at each node. Such a protocol would ensure that queries are answered with low latency, albeit at higher cost. For the second case, a network backbone would need to be continually maintained, since user queries can be injected into the network anytime and from anywhere. A protocol like ASCENT [9], that maintains a network routing topology continually, would enable such querying, while being robust to environmental dynamics.

Debugging the network deployment

The following example can illustrate why debugging a deployed network such as the one for environmental monitoring can be difficult.

An unexpected situation has arisen and performance of the deployed sensor network has degraded significantly: queries are lost, throughput is low, individual nodes cannot be contacted, reliable transmission is slow and involves many retransmissions. A maintenance engineer is called to the site to look into the situation. There are a number of possible causes: node failures causing a ill connected network, interference from new sources, general battery degradation over time requiring adjustment of operating parameters, etc. A significant amount of debugging information is required to clearly understand the problem. Debugging each individual node by physically accessing it is impractical given the scale, and infeasible due to the lack of sufficient personnel.

How can the engineer design a networked debugging scheme to tackle such problems? A suite of debugging tools such as eScan can be appropriately used to solve the problems. For the above instance, periodic *digests*[44] can provide clues to the engineer to help debug the problem.

Geographic Data Forwarding

The first step in constructing the distributed storage structure is the selection of cluster nodes that are responsible for data storage at different resolutions. Dimensions builds on prior work in Data Centric Storage (DCS [47]) that constructs a Distributed Hash Table (DHT [48]) to hash the name of a certain event key (dataName) to a location within the network. Each node in the network uses the same hash function, and thus global consensus on the mapping between locations and keys is achieved.

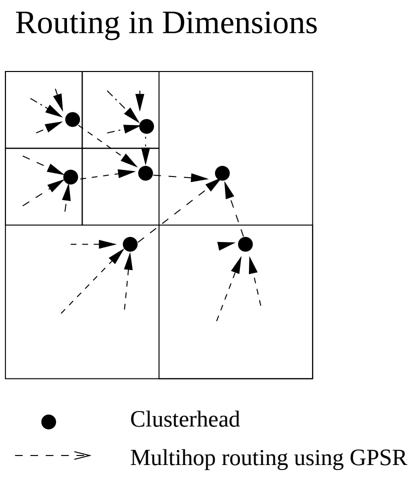


Fig. 9. Routing between clusterheads at different levels of the hierarchy involves using the GPSR geographic routing protocol.

This procedure is slightly modified in Dimensions to elect a clusterhead to represent each square grid using a globally well-known hash function. A geographic hash function is used that accepts the location of the source data and current time, and returns the location corresponding to the clusterhead is used.

Two routing questions are posed by such an approach. First, how is routing done to the chosen location? Second, how can one ensure that a node is present at the location, since the hash function operates without a knowledge of the global topology?

GPSR, described in Section 4.2, can be used to address the first concern, i.e. to forward data to the destination location obtained from the hash function. To address the second problem, the perimeter forwarding strategy in GPSR is cleverly used in DCS ([47]) to consistently deliver data addressed to a location to the node *closest* to the target location. Thus, the storage application does not need to deal with this translation.

Drill-down Queries

Sensor data for the application is identified by node location, time, and sensor type. Queries on the data can be expressed using these attributes and operators to manipulate them.

Consider a simple MAX query that operates on temperature data from the network: *Find the Max temperature between Jan and July, 2003*. The query is first routed to the root, which has a coarse summary corresponding to the entire network. The root runs the query over its local summary and identifies which quadrant is likely to have the maximum temperature for the specified period. Since the root has only a coarse summary, the result returned by this processing is likely to be approximate. However, further drill-down is likely to improve the accuracy of the result. The query is geographically routed to the clusterhead corresponding to the quadrant, and the procedure repeats.

How can drill-down query processing be implemented using Directed Diffusion? Nested queries, discussed in [31], provide a natural way of expressing drill-down querying. Nested queries can be implemented by enabling code at each triggered sensor that watches for a nested query. This code then generates a sub-query that is routed to the relevant quadrant for further refining the approximate query answer. Each further level of drilldown can be expressed as an additional level of nesting, where the result of the processing determines the next level query.

7 Conclusions

Wireless sensor networks enable dense sensing of the environment, offering unprecedented opportunities for observing the physical world. These systems offer unique challenges to researchers: scale, unpredictable wireless communication conditions, severely energy and resource constraints. In this paper, we have summarized research in networking techniques for sensor networks, focusing on representative approaches.

As the sensor network community moves out of its infancy, and embark on real deployments [17, 45], many of these networking schemes that have been developed are being put to test in real environments. Each application domain will, no doubt, introduce novel challenges, and involve application-specific optimization to the networking schemes that we have discussed. However, the schemes that we have discussed provide a broad library of techniques from which to select appropriate strategies.

References

[1] Ten emerging technologies that will change the world. *Technology Review*, February 2003.

[2] Deborah Estrin, Ramesh Govindan, John Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, pages 263–270, 1999.

[3] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[4] G.J. Pottie and W.J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, May 2000.

[5] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 56–67, Boston, MA, USA, August 2000. ACM.

[6] Deepak Ganesan, Ben Greenstein, Denis Perelyubskiy, Deborah Estrin, and John Heidemann. An evaluation of multi-resolution search and storage in resource-constrained sensor networks. In *Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys).*, 2003. to appear.

[7] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the IEEE Infocom*, pages 1567–1576, New York, NY, June 2002. IEEE.

[8] Wei Ye, John Heidemann, and Deborah Estrin. Medium access control with coordinated, adaptive sleeping for wireless sensor networks. Technical Report ISI-TR-567, USC Information Sciences Institute, January 2003. To appear in the IEEE/ACM Transactions on Networking.

[9] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sEnsor networks topologies. In *Proceedings of the IEEE Infocom*, New York, NY, June 2002. IEEE.

[10] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 85–96, Rome, Italy, July 2001. ACM.

[11] Curt Schurgers, Vlasios Tsiatsis, Saurabh Ganeriwal, and Mani Srivastava. Optimizing sensor networks in the energy-latency-density space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, 2002.

[12] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, Atlanta, GA, September 2002. ACM.

[13] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-IX)*, pages 93–104, Cambridge, MA, USA, November 2000. ACM.

[14] Larry L. Peterson. A yellow-pages service for a local-area network. *Proceedings of ACM SIGCOMM '87*, pages 235–242, August 1987.

[15] William Adjie-Winoto, Elliot Schwartz, Hari Balakrishnan, and Jeremy Lilley. The design and implementation of an intentional naming system. In *Proceedings of the 17th Symposium on Operating Systems Principles*, pages 186–201, Kiawah Island, SC, USA, December 1999.

[16] Tomasz Imielinski and Samir Goel. DataSpace: Querying and Monitoring Deeply Networked Collections in Physical Space. *IEEE Personal Communications. Special Issue on Smart Spaces and Environments*, 7(5):4–9, October

2000.

[17] Michael Hamilton. James San Jacinto Mountains Reserve.

[18] Deepak Ganesan, Deborah Estrin, and John Heidemann. Dimensions: Why do we need a new data handling architecture for sensor networks? In *First Workshop on Hot Topics in Networks (Hotnets-I)*, volume 1, October 2002.

[19] Mark Stemm and Randy H Katz. Measuring and reducing energy consumption of network interfaces in hand-held devices. *IEICE Transactions on Communications*, E80-B(8):1125–1131, August 1997.

[20] Oliver Kasten. Energy consumption. `http://www.inf.ethz.ch/~kasten/research/bathtub/energy_consumption.html`. Eldgenossische Technische Hochschule Zurich.

[21] Frazer Bennett, David Clarke, Joseph B. Evans, Andy Hopper, Alan Jones, and David Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications Magazine*, 4(5):8–15, October 1997.

[22] Jaap C. Haartsen. The Bluetooth radio system. *IEEE Personal Communications Magazine*, pages 28–36, February 2000.

[23] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocols for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on Systems Sciences*, January 2000.

[24] Katayoun Sohrabi and Gregory J. Pottie. Performance of a novel self-organization protocol for wireless ad hoc sensor networks. In *Proceedings of the IEEE 50th Vehicular Technology Conference*, pages 1222–1226, 1999.

[25] LAN MAN Standards Committee of the IEEE Computer Society. *Wireless LAN medium access control (MAC) and physical layer (PHY) specification*. IEEE, New York, NY, USA, IEEE Std 802.11-1999 edition, 1999.

[26] Yu-Chee Tseng, Chih-Shun Hsu, and Ten-Yueng Hsieh. Power-saving protocols for IEEE 802.11-based multi-hop ad hoc networks. In *Proceedings of the IEEE Infocom*, pages 200–209, New York, NY, June 2002. IEEE.

[27] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 70–84, Rome, Italy, July 2001. ACM.

[28] Fabio Silva, John Heidemann, and Ramesh Govindan. *Network Routing Application Programmer's Interface (API) and Walk Through 9.0.1*. USC/Information Sciences Institute, December 2002.

[29] Philippe Bonnet and Praveen Seshadri. Device database systems. In *16th International Conference on Data Engineering*, San Diego, California, February 2000.

[30] Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI*, volume 1, Boston, MA, 2002.

[31] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the Symposium on Operat-*

ing Systems Principles, pages 146–159, Chateau Lake Louise, Banff, Alberta, Canada, October 2001. ACM.

[32] John Heidemann, Fabio Silva, and Deborah Estrin. Matching data dissemination algorithms to application requirements. Technical Report ISI-TR-571, USC/Information Sciences Institute, April 2003.

[33] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proc. ACM Mobicom*, Boston, MA, 2000.

[34] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report, May 2001.

[35] Gregory G. Finn. Routing and addressing problems in large metropolitan-scale internetworks. Technical Report ISI/RR-87-180, USC/ISI, March 1987.

[36] J. Case, M. Fedor, M. L. Schoffstall, and C. Davin. Simple Network Management Protocol. Technical report, internet-rfc, November 1996.

[37] S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. Coverage Problems in Wireless Ad-hoc Sensor Networks. In *Proceedings of the IEEE Infocom*, 2001.

[38] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. Exposure In Wireless Ad Hoc Sensor Networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom)*, pages 139–150, July 2001.

[39] S. Chessa and P. Santi. Comparison based system-level fault diagnosis in ad-hoc networks. In *Proceedings of the 20th*, October 2001.

[40] J. Staddon, D. Balfanz, and G. Durfee. Efficient tracing of failed nodes in sensor networks. In *Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, pages 122–130, Atlanta, USA, September 2002.

[41] Y. Zhao, R. Govindan, and D. Estrin. Residual energy scans for monitoring wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference*, March 2002.

[42] Raquel A. F. Mini, Badri Nath, and Antonio A. F. Loureiro. Prediction-based Approaches to Construct the Energy Map for Wireless Sensor Networks. In *21 Simpsio Brasileiro de Redes de Computadores*, page to appear, Natal, RN, Brasil, May 2003.

[43] B. Deb, S. Bhatnagar, and B.Nath. Multi-resolution State Retrieval in Sensor Networks. In *Proceedings of the IEEE ICC Workshop on Sensor Network Protocols and Applications*, page to appear, Anchorage, AK, May 2003.

[44] J. Zhao, R. Govindan, and D. Estrin. Computing Aggregates for Monitoring Wireless Sensor Networks. In *Proceedings of the IEEE ICC Workshop on Sensor Network Protocols and Applications*, page to appear, Anchorage, AK, May 2003.

[45] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[46] A.A. Abidi, G.J. Pottie, and W.J. Kaiser. Power-conscious design of wireless circuits and systems. *Proceedings of the IEEE*, 88(10):1528–45, October 2000.

[47] S. Ratnasamy, D. Estrin, R. Govindan, B. Karp, L. Yin S. Shenker, and F. Yu. Data-centric storage in sensornets. In *ACM First Workshop on Hot Topics in Networks*, 2001.

[48] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght - a geographic hash-table for data-centric storage. In *First ACM International Workshop on Wireless Sensor Networks and their Applications*, 2002.