The Cougar Approach to In-Network Query Processing in Sensor Networks

Yong Yao Johannes Gehrke
Department of Computer Science
Cornell University

Abstract

The widespread distribution and availability of small-scale sensors, actuators, and embedded processors is transforming the physical world into a computing platform. One such example is a sensor network consisting of a large number of sensor nodes that combine physical sensing capabilities such as temperature, light, or seismic sensors with networking and computation capabilities. Applications range from environmental control, warehouse inventory, and health care to military environments. Existing sensor networks assume that the sensors are preprogrammed and send data to a central frontend where the data is aggregated and stored for offline querying and analysis. This approach has two major drawbacks. First, the user cannot change the behavior of the system on the fly. Second, conservation of battery power is a major design factor, but a central system cannot make use of in-network programming, which trades costly communication for cheap local computation.

In this paper, we introduce the Cougar approach to tasking sensor networks through declarative queries. Given a user query, a query optimizer generates an efficient query plan for in-network query processing, which can vastly reduce resource usage and thus extend the lifetime of a sensor network. In addition, since queries are asked in a declarative language, the user is shielded from the physical characteristics of the network. We give a short overview of sensor networks, propose a natural architecture for a data management system for sensor networks, and describe open research problems in this area.

1 Introduction

One of the characteristics of the post-PC era is to push computation from desktops and data centers out into the physical world. This is an exciting time for systems research, as platforms emerge with characteristics quite different from traditional environments. Already today networked sensor nodes can be constructed using commercial components using only a fraction of a watt in power on the scale of a few centimeters. Figure 1 shows a Berkeley MICA Mote, one of the platforms available today, and Figure 2 shows the hardware characteristics of a mote. These prototypes measure about 5cm^3, and application of Moore's law tells us that we will soon see components that measure 1 cm^3 running Unix [45, 47] or an embedded microkernel [19], and there is a plethora of research to scale components down to about 1 mm^3 (about the size of a large piece of dust) integrated into the physical environment potentially powered by ambient energy [25, 27].

Sensor networks have the following physical resource constraints:

- **Communication.** The bandwidth of wireless links connecting sensor nodes is usually limited, on the order of a few hundred Kbps. In addition, the wireless network connecting the sensor nodes provides usually only very limited quality of service, has latency of high variance, and drops packets frequently.

- **Power consumption.** Sensor nodes have limited supply of energy, and energy conservation is one of the main system design considerations. Current small batteries provide about 3000mAh of capacity, powering the MICA Mote for approximately one year in the idle state and for one week under full load. Note that future sensor nodes will have sophisticated power management features; current nodes already have three different sleep modes with several orders of magnitude different power usages [20].

- **Computation.** Sensor nodes have limited computing power and memory sizes that restrict the types of data processing algorithms that can be

Figure 1: A Berkeley MICA Mote

deployed and intermediate results that can be stored on the sensor nodes.

- **Uncertainty in sensor readings.** Signals detected at physical sensors have uncertainty due to limitations of the sensor, and they may contain environmental noise. Sensor malfunctions might generate inaccurate data, and unfortunate sensor placement (such as a temperature sensor directly next to the air conditioner) might bias individual readings.

Potential applications for sensor networks abound. One possible example could be intelligent building management. Sensor nodes are deployed in offices and halls to measure noise, temperature, and the level of light, and interact with the control system to adjust the environment automatically. People may pose queries of interest to the sensor network, such as "Is Johannes in his office", or "Is there an empty seat in the meeting room"? Sensor networks can also be used in physical environment to benefit scientific research. A biologist may want to know of the existence of a specific bird, and when this particular bird is detected, the bird should be followed as closely as possible. In this case, the sensor network is used for automatic target recognition and tracking. More specific applications in different fields will arise, and instead of deploying preprogrammed sensor networks only for specific applications, future networks will have sensor nodes with different physical sensors for a wide variety of application scenarios and different user groups. (The MICA motes already support temperature sensors, a magnetometer, an accelerometer, a microphone, and also several actuators.)

In this paper, we advocate a database approach to sensor networks. Our approach is motivated by two main reasons. First, declarative queries are especially suited for sensor network interaction: Users and application programs issue queries without knowing how the data is generated in the sensor network and how the data is processed to compute the query answer. Sophisticated catalog management, query optimization, and query processing techniques will abstract the user from the physical details of contacting the relevant sensor nodes, processing the sensor data, and sending the results to the user. To enable declarative querying of sensor networks, we propose a *query layer* consisting of a *query proxy* on every sensor node. Architecturally, on the sensor node, the query proxy lies between the network layer and the application layer, and the query proxy provides higher-level services through queries that can be injected into the network from a specified gateway node.[1]

Processor	4Mhz, 8bit MCU (ATMEL)
Memory	128KB Flash, 4KB RAM, 4KB ROM
External Flash	512KB
Radio	916Mhz Radio (RF Monolithics)
Communication Range	200 ft
Data Rate	50 Kbits/sec
Transmit Current	12 mA
Receive Current	1.8 mA
Sleep Current	5 uA

Figure 2: Hardware Characteristics of the MICA Motes

Our database approach is motivated by a second reason. Since nodes are usually powered by batteries, increasing network lifetime is a major design goal of any sensor network system. Data transmission back to a central node for offline storage, querying, and data analysis is very expensive for sensor networks of non-trivial size since communication using the wireless medium consumes a lot of energy [11, 38]. Since sensor nodes have local computation abilities, part of the computation can be moved from a location outside the network and pushed into the sensor network, aggregating records, or eliminating irrelevant records. In-network processing can reduce energy consumption and improve sensor network lifetime significantly compared to traditional centralized data extraction and analysis. Thus one of the main roles of the query proxy when processing user queries is to perform in-network processing.

Given the view of the sensor network as a huge distributed database system where each sensor node corresponds to a database site that holds part of the data, we would like to adapt existing techniques from distributed and heterogeneous database systems for the sensor network environment. But as we outline

[1] In some applications it might be desirable to inject queries into the system from arbitrary nodes.

in Section 2, there are major differences between sensor networks and traditional distributed and heterogeneous database systems that require us to rethink current approaches to distributed data management.

The remainder of this paper is structured as follows. In Section 2, we will give an introduction to sensor networks as a processing platform, describe the properties of sensor networks and the associated data, and introduce a natural architecture for a sensor data management system. In Section 3, we outline open research problems in in-network query processing for sensor networks, and in Section 4, we briefly review related work. For technical papers related to the Cornell Cougar Project, visit `http://www.cs.cornell.edu/database/cougar`.

2 Sensor Networks

2.1 Properties of Sensor Networks

A sensor network consists of a large number (currently up to several thousand) of sensor nodes.[2] Individual sensor nodes (or short, nodes) are connected to other nodes in their vicinity through a wireless communication interface, and they use a multihop routing protocol to communicate with sensor nodes that are spatially distant. Nodes also have limited computation and storage capabilities: a node has a general-purpose CPU to perform computation and a small amount of storage space to save program code and data.

Since nodes are usually not connected to a fixed infrastructure, they use batteries as their main power supply, and preservation of power is one of the main design goals in a sensor network. Since communication consumes much more energy than computations, it is attractive to reduce the amount of traffic flow between nodes by local computation.

Sensor nodes might reside in a hostile environment, such as battlefields or in regions of recent disasters to support rescue missions, thus a sensor node might fail at any time. In addition, the communication links between adjacent nodes might break frequently due to environmental interferences and noise. Thus it is critical to design a robust system that recovers fast after failures.

2.2 Sensor Data

A sensor node has one or more physical sensors attached that are connected to the physical world. Example sensors are temperature sensors, light sensors, or PIR sensors that can measure the occurrence of events (such as object detections) in their vicinity. Sensor readings are usually timestamped. If an application cares about the current state of the network, readings from the network have to be updated relatively frequently since sensor data becomes outdated fast if new events are happening in the network. Long-running queries that recompute query results periodically are one possibility to keep query results up-to-date.

Sensor networks are distributed to measure and monitor a physical environment, such as tracking objects throughout an area or measuring environmental conditions in a large area. Due to the multitude of sensor nodes deployed, there is usually a huge number of data records generated. For example, in environmental monitoring applications, sensor readings are generated every few seconds (or even faster), thus the total volume of data generated is quite large. However, not all sensor readings are of interest to users. For some sensor types, their data might change rapidly, and thus be outdated rather quickly, whereas for other sensors, their value changes only slowly over time. Example sensors of the first type are PIR sensors that sense the presence of objects; example sensors of the second type are temperature sensors that in steady state have a small bounded derivative. For applications that require only approximate results, we can cache previous results for the second type of sensors and lower the query update rate to save energy.

Inherent to data from a physical measurement is *uncertainty* regarding the true value of the measured quantity. This uncertainty is often most properly described by a *continuous probability distribution function* over the possible measurement values [12]. For example, consider a temperature sensor in your office that reports an estimate $\widehat{T}$ of the current temperature T; let this estimate be $\widehat{T} = 68°$ Fahrenheit (F). Given this measurement, do we believe that the temperature in your office is exactly 68° F? Assuming that the error introduced by the sensor has a Gaussian distribution with a known standard deviation of σ°F, we can compute the probability that the true temperature T lies in the range $[T_1, T_2]$. In the context of a sensor network, a user should be able to submit a query that retrieves all temperatures whose true values lie in the range $[T1, T2]$ with a given probability. Moreover, as long as several sensors nodes measure the same physical phenomenon, their readings can be aggregated to construct a "super-node" whose temperature readings have a much lower variance than

[2] Hardware trends let us believe that the number of nodes in future sensor networks will drastically increase.

the readings from individual sensor nodes.

Another source of uncertainty in sensor readings is noise. For many applications individual sensor readings are of minor importance, and users are usually interested in aggregates that *fuse* a set of sensor data readings (often from multiple different sensors) into a single, more robust statistic [16].

2.3 An Architecture for a Sensor Database System

Existing sensor networks work mainly as data collectors, and transfer data from sensor nodes to a central frontend where the data is aggregated and stored for offline querying and analysis. Fjords improve the centralized architecture by sharing scan operators at the sensor nodes and switching sensors on and off according to query specifications. They help to reduce energy consumption, as nodes are aware of user queries, but they lack support for more advanced query processing techniques [29].

Since local computation is much cheaper than communication, pushing partial computation out into the network could improve energy consumption significantly. We propose a loosely-coupled distributed architecture to support both aggregation and more complicated in-network computation. In our architecture, there is a new *query proxy layer* on each sensor node, interacting with both routing layer and application layer. A query optimizer is located on the gateway node to generate distributed query processing plans after receiving queries from the outside. The query plan is created according to catalog information and the query specification. Such a query plan specifies both the data flow (between sensors) and an exact computation plan (at each sensor). The plan is then disseminated to all relevant sensor nodes. Control structures are created to synchronize sensor behavior, and the query is started. At run-time, data records flow back to the gateway node as in-network computation happens on-the-fly.

2.4 An Example

Let us illustrate the individual components of the architecture with a simple example. Suppose we have a long-running query Q to monitor the average temperature of an office every t seconds. The query Q notifies (i.e., Q generates an output record) an administrator if the average temperature in the office is greater than a user-defined threshold.

As a first step in evaluating this query, the query optimizer will optimize the query, taking the existing query workload into account by trying to *merge* the new query with existing, similar queries. Assuming that the query Q is the only query that is running in the network, the query optimizer will generate a new query plan QP. The query plan QP specifies how to determine the *leader* of this query, a designated node where the computation of the average temperature will take place. The leader could be a fixed sensor with more remaining power and energy, or a randomly selected node by some distributed leader election algorithm. Two computation plans are produced, one for the leader node, and a second plan for the remaining nodes in the query region.

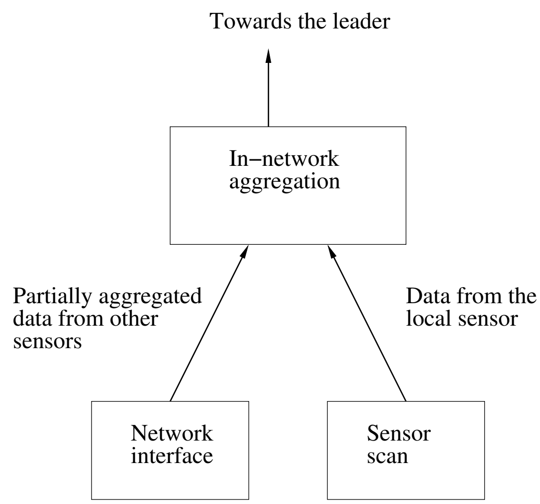

Figure 3: Query Plan at a Source Sensor

Figure 3 shows the query plan for a non-leader node that participates in the query. Non-leader nodes have a scan operator to read sensor values periodically and to send them to the leader node. In addition, their plan contains an aggregation operator to aggregate data from other sensors. Figure 4 shows the query plan for the leader node, which contains an `AVG` operator to compute the average value over all sensor readings received in the last round of the query, and a `SELECT` operator that checks if the result is above the threshold.

At query start time, the query plans are disseminated to the query proxies of all relevant sensor nodes. The query proxies will register the query, create a local operator tree, active relevant sensors, and return records according to the specification of the query plan. The leader will generate a record only if the average temperature is above the user-defined threshold.

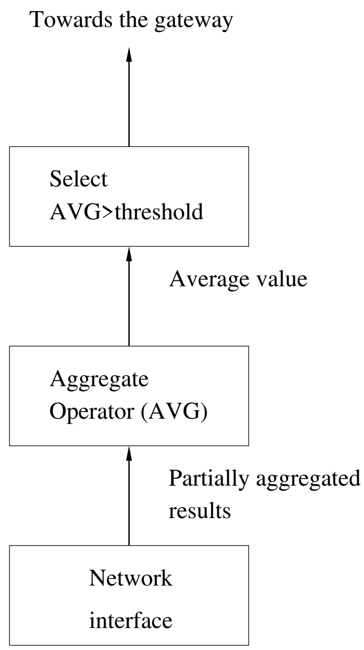

Figure 4: Query Plan at the Leader

3 Research Problems

In this section, we give an overview of some research problems in in-network query processing for sensor networks using the architecture described in Section 2.3 as a starting point.

3.1 Aggregation

Aggregation refers to delivering data from distributed source sensor nodes to a central node for computation. It is one of the most popular computation *and* communication pattern for sensor network. (Recall that example query Q from the previous section is an aggregation query.) A join of sensor readings from different groups could be another example; all records need to be collected at a central node where the join takes place.

Aggregation involves two important issues. First, from a computational point of view, the aggregation has to complete at a "leader" node (unless the final computation of the aggregate is delegated to a gateway node or happens outside of the network). Second, the data records have to be delivered from source sensor nodes to the designated leader. Note that the system has to be designed to tolerate the volatility of the underlying communication layer: messages could get lost, nodes could die, and the network could be partitioned for a while. Let us shortly contemplate leader selection and data delivery in the following two paragraphs.

If the computation is designated to a leader node, such a leader needs to be selected among the sensor nodes. There are several basic requirements for the leader selection policy. First, the leader should be dynamically maintained in case of sensor or link failure. (We imagine backup leaders to reduce the cost in case of leader failure.) Here we can draw upon a large literature about algorithms in distributed systems and the leader election problem. Second, we would like to select a leader with physically advantageous location: The cost of data delivery from source sensors to the leader *and* the delivery cost from the leader to the gateway node need to be taken into account to save communication energy.

How can we deliver data from sensor nodes to their leader node? The simplest way is to send all data records directly to the leader along multi-hop routes, and to do all the computation directly at the leader. This is a reasonable solution for a small networks. However, if we consider the computation of aggregates over larger regions, this scheme will generate many messages and consume a lot of sensor battery power. An alternative solution is to push partial computation from the leader to internal nodes along the path to reduce data size on-the-fly. This solution works for aggregation operators that are incrementally computable, such as avg, max, and moments of the data. The final answer can be computed from partially aggregated intermediate results, which are usually as small as the original data record. The only caveat is that this in-network computation requires synchronization between sensor nodes along the communication path, since a node has to "wait" to receive results to be aggregated. In networks with high loss rates, broken links are hard to differentiate from long delays due to high loss rates, making synchronization a non-trivial problem.

3.2 Query Languages

We believe that sensor networks will be deployed in various environments, and they will be used by diverse applications. By utilizing different types of sensors, it is very easy to extend the functionality of sensor networks to support more applications. Some of these applications have been around for quite a while and are already implemented in prototype systems (such as monitoring a physical environment or tracking moving objects). In other cases, technology development is driving the applications, and applications of sensor networks in fields like biology or geology are just emerging. Thus the development of a query language is at a difficult point: There are classes of queries such as aggregation queries for which we know that they are very useful and that any query language for sensor data should support them. Other aspects of a query language are less clear at this point, and time and emerging applications will tell what common functionality will be required. One possible approach for the development of a query language is to look at the properties of the sensor data itself, and to abstract computational patterns that are possible with the type of data generated in sensor networks.

3.3 Query Optimization

In-network query processing can reduce energy consumption and thus prolong sensor network lifetime. A complex query may consist of a large number of parameters and operators, in addition to various user requirements on the query answers, such as specification of a maximum permissible latency and accuracy of the query result. There is usually a large space of query processing plans for such complex queries, and it is the query optimizer's responsibility to select a good plan within a large space of possible query execution plans.

A good plan could be the plan with minimum energy usage, or it could make a good tradeoff between various requirements given the limitations of available resources. Thus the notion of the *cost* of a query plan has changed, as the critical resource in a sensor network is power, and query optimization and query processing have to be adapted to take this optimization criterion into account.

Recall from our example, that the plan generated by the query optimizer will describe both the data flow inside the network and the computation flow within each sensor node. Since a sensor network is highly dynamic, the optimizer needs to make these decisions with inherent uncertainty about the catalog information. In addition, the query plan needs to react to changes in catalog information such as the network topology and the power level at sensor nodes. We can envision both semi-centralized as well as completely adaptive query processing strategies, but more research is needed to quantify the benefits of each approach.

3.4 Catalog Management

To generate a good plan for a user query, the optimizer requires metadata about the status of the sensor network to evaluate the costs and benefits (latency and accuracy) of different plans. A catalog could be built and maintained at the server to maintain important information, like sensor position (potentially aggregated), density and connectivity, system workload, and network stability. System-generate queries could be used to update the catalog periodically, or the catalog could be assembled dynamically through gossip-style information dissemination. Due to the size of the metadata and the dynamics of the sensor network, it is likely prohibitive to collect all metadata at a central node, and to keep them always sufficiently up-to-date. It is an interesting research problem to define efficient synopsis data structures, that are cheap to create and maintain, but still contain sufficient details for query optimization.

3.5 Multi Query Optimization

Multi query optimization is another challenging problem. The sensor network is usually shared by many users (and this could be reflected in the architecture by having several gateway nodes that connect to the different users). In this case, multiple queries flow into the network through different gateway nodes, and it is likely that many users pose similar queries such that queries can share intermediate results.

4 Related Work

In this section, we discuss past work that is relevant to data management in sensor networks. We do not claim by any means that the set of topics and citations is exhaustive, and any omissions are unintentional.

4.1 Ad-hoc and Wireless Routing

Research in dynamic wireless communication networks has a long history, dating back to DARPA's PRNET (Packet Radio Network) [23] and SURAN (Survivable Adaptive Networks) projects [41], and a plethora of papers has been published on routing protocols for ad-hoc mobile wireless networks [6, 10, 21, 22, 35, 36, 37]. Metrics for evaluating these protocols are packet loss, routing message overhead, message latency, and route length. We believe that a sensor data management infrastructure requires additional metrics for evaluation, such as quality of the returned answer, and overhead for metadata management.

4.2 Power Awareness

The networking community has recently started to investigate the issue of power-aware networking infrastructures. Chang and Tassiulas [8] and Pottie et al. [38] suggest to select routes in an ad-hoc network based on available energy in order to increase network lifetime. Heinzelman et al. present the SPIN family of network protocols for communication of large messages in sensor networks [17]; they propose to first distribute metadata instead of flooding the network with actual data. The distribution of metadata eliminates duplicate transmission of the same data record.

The PAMAS MAC-level protocol turns radios off when they are not transmitting or receiving packets [44]. TDMA protocols have been proposed to reduce energy consumption in sensor networks [38]. By reducing the duty cycle these protocols can trade idle-time energy consumption for latency, similar to the tradeoffs we envision for query answers. We believe TDMA MAC protocols will be important for power-constrained networks and use of application-level information from the database layer can further improve power conservation.

Ramanathan and Rosales-Hain developed protocols for adjusting transmit power in ad-hoc networks [40]. By controlling the radio transmission power they try regulate the number of neighbors of each node in the network with the goal to improve connectivity in sparse networks and collisions in dense networks. PicoNet proposes an integrated design of radios, small, battery powered nodes, and MAC and application protocols that minimize power consumption [5]. IEEE 802.11 supports ad-hoc network configuration and provides power management controls [31].

4.3 Distributed Database Systems.

Work on query processing in distributed database systems is relevant to this field, although it is unlikely that existing techniques can be applied directly. There are several excellent surveys and books on distributed query processing, such as work by Yu and Chang [51], Ceri and Pelagatti [7], Özsu and Valduriez [33], Yu and Meng [50], and Kossmann [26].

4.4 Adaptive Query Processing.

We believe that techniques for adaptive query processing will be very relevant for data management in sensor networks. Chen and Rousopoulos designed an adaptive selectivity estimation schema that adds statistics-gathering to regular query processing [9]; we can envision the use of similar techniques in a sensor network setting where small feedback is piggybacked on results to long-running queries. Lack of perfect global knowledge is also an inherent problem in distributed and heterogeneous database systems. One approach to adapting to this uncertainty is to send subqueries to remote sites for local processing [32, 46]. Query scrambling can deal with unexpected delays when processing queries in a wide-area network, a setting similar to a sensor network [3, 49]. Kabra and DeWitt proposed to reoptimize parts of queries after blocking operators [24].

There is also a lot of work on adaptive query operators, an area we believe to be relevant to sensor networks. Examples include work on memory-adaptive sorting and hashing [13, 28, 30, 34, 53, 54], and online aggregation algorithms [15, 18, 39, 48]. Eddies push the idea of feedback on a tuple-by-tuple basis in online aggregation to adapting join orders at the same frequency [4].

Other relevant work includes sequence query processing [42, 43], and temporal and time-series databases [52].

4.5 Related Projects

We conclude our discussion of related work with a short collection of related projects.

- The CoSense Project at Xerox PARC.
  `http://www2.parc.com/spl/projects/cosense`
  Object tracking and identification through collaborative signal processing and distributed statistical hypothesis management.

- The SCADDS Project at UCLA and ISI.
  `http://www.isi.edu/scadds/`
  Networking and coordination between sensor nodes.

- The WebDust Project at Rutgers.
  `http://www.cs.rutgers.edu/dataman/webdust`
  Routing protocols and predictive monitoring.

- Agent-based Tasking of Massive Sensor Networks at Maryland.
  `http://www.cs.umd.edu/users/vs/senseit.html`
  Concentrates on multi-query optimization and high-level languages.

- Reactive Sensor Networks at Penn State.
  `http://strange.arl.psu.edu/RSN/`
  Object tracking and mobile code.

- TinyOS at Berkeley.
  `http://today.cs.berkeley.edu/tos/`
  Operating systems support for sensor networks.

- The Telegraph Project at Berkeley.
  `http://telegraph.cs.berkeley.edu/`
  Adaptive query processing strategies.

- Location-Centric Distributed Computation and Signal Processing at Wisconsin. http://www.ece.wisc.edu/~sensit/ Communication primitives, tracking, and data fusion.

5 Conclusions

Sensor networks will become ubiquitous, and the database community has the right expertise to address the challenging problems of tasking the network and managing the data in the network. We described one possible architecture of a sensor data management system, and we discussed how previous work relates to the system we envision. We also laid out a set of challenging research problems, including distributed in-network processing, query optimization, query languages, catalog management, and multi-query optimization.

We have started at Cornell to design and implement a prototype that allows us to experiment with the design space of various algorithms and data structures. We are currently working on in-network aggregation, integration of query processing with the routing layer, query optimization, and a probabilistic data model for sensor data.

We anticipate that the emergence of new applications, as well as the implementation and usage of our prototype system will lead to other research directions. For more information on the current status, and to play with our prototype, visit:

http://www.cs.cornell.edu/database/cougar

Acknowledgments. We thank the DARPA SensIT Community for helpful discussions about the role of query processing in sensor networks. The Cornell Cougar Project was envisioned by Praveen Sheshadri, and Philippe Bonnet made influential initial contributions. The Cougar Project is supported by the DARPA SensIT program, a National Science Foundation CAREER grant, the Cornell Information Assurance Institute, a grant from Lockheed Martin, and by gifts from Intel and Microsoft. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] ACM SIGMOBILE. *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM-98).* ACM Press, 1998.

[2] ACM SIGMOBILE. *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom-99.* ACM Press, 1999.

[3] L. Amsaleg, M. J. Franklin, A. Tomasic, and T. Urhan. Scrambling query plans to cope with unexpected delays. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems, December 18-20, 1996, Miami Beach, Florida, USA*, pages 208–219. IEEE Computer Society, 1996.

[4] R. Avnur and J. M. Hellerstein. Eddies: Continuously adaptive query processing. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 261–272. ACM, 2000.

[5] F. Bennett, D. Clarke, J. Evans, A. Hopper, A. Jones, and D. Leask. Piconet: Embedded Mobile Networking. *IEEE Personal Communications*, 4(5):8–15, Oct. 1997.

[6] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. [1], pages 85–97.

[7] S. Ceri and G. Pelagatti. *Distributed Database Design: Principles and Systems.* MacGraw-Hill (New York NY), 1984.

[8] J.-H. Chang and L. Tassiulas. Energy conserving routing in wireless ad-hoc networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00)*, pages 22–31, Los Alamitos, Mar. 26–30 2000. IEEE.

[9] C.-M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. In R. T. Snodgrass and M. Winslett, editors, *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, Minneapolis, Minnesota, May 24-27, 1994*, pages 161–172. ACM Press, 1994.

[10] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00)*, pages 3–12, Los Alamitos, Mar. 26–30 2000. IEEE.

[11] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. [2], pages 263–270.

[12] A. Faradjian, J. Gehrke, and P. Bonnet. Gadt: A probability space adt for representing and querying the physical world. In *International Conference on Data Engineering*, 2002.

[13] G. Graefe, R. Bunker, and S. Cooper. Hash joins and hash teams in microsoft sql server. In A. Gupta, O. Shmueli, and J. Widom, editors, *VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA*, pages 86–97. Morgan Kaufmann, 1998.

[14] L. M. Haas and A. Tiwary, editors. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*. ACM Press, 1998.

[15] P. J. Haas and J. M. Hellerstein. Ripple joins for online aggregation. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 287–298. ACM Press, 1999.

[16] D. L. Hall and J. Llinas, editors. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[17] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. [2], pages 174–185.

[18] J. M. Hellerstein, P. J. Haas, and H. Wang. Online aggregation. In J. Peckham, editor, *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 171–182. ACM Press, 1997.

[19] D. Hildebrand. An architectural overview of qnx. http://www.qnx.com/, 2001.

[20] J. Hill, R. Szewczyk, A. Woo, D. Culler, S. Hollar, and K. Pister. System architecture directions for networked sensors. *ACM SIGPLAN Notices*, 35(11):93–104, Nov. 2000.

[21] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and M. Degermark. Scenario-based performance analysis of routing protocols for mobile adhoc networks. [2], pages 195–206.

[22] D. B. Johnson and D. A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Sereies in Engineering and Computer Science*. Kluwer Academic Publishers, 1996.

[23] J. Jubin and J. D. Tornow. The DARPA packet radio network protocol. *Proceedings of the IEEE*, 75(1):21–32, Jan. 1987.

[24] N. Kabra and D. J. DeWitt. Efficient mid-query reoptimization of sub-optimal query execution plans. In Haas and Tiwary [14], pages 106–117.

[25] J. M. Kahn, R. H. Katz, and K. S. J. Pister. Next century challenges: Mobile networking for "smart dust". [2], pages 271–278.

[26] D. Kossmann. The state of the art in distributed query processing. *Computing Surveys*, 32, 2000.

[27] J. Kymissis, C. Kendall, J. Paradiso, and N. Gershenfeld. Parasitic power hasvesting in shoes. In *In Proceedings of the Second IEEE International Conference on Wearable Computing (ISWC)*, pages 132–139. IEEE Computer Society Press, October 1998.

[28] P.-Å. Larson and G. Graefe. Memory management during run generation in external sorting. In Haas and Tiwary [14], pages 472–483.

[29] S. Madden and M. J. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *18th International Conference on Data Engineering*, 2002.

[30] M. Nakayama, M. Kitsuregawa, and M. Takagi. Hash-partitioned join method using dynamic destaging strategy. In F. Bancilhon and D. J. DeWitt, editors, *Fourteenth International Conference on Very Large Data Bases, August 29 - September 1, 1988, Los Angeles, California, USA, Proceedings*, pages 468–478. Morgan Kaufmann, 1988.

[31] L. M. S. C. of the IEEE Computer Society. Wireless lan medium access control (mac) and physical layer (phy) specification. IEEE Std 802.11, 1999.

[32] F. Ozcan, S. Nural, P. Koksal, C. Evrendilek, and A. Dogac. Dynamic query optimization on a distributed object management platform. In *CIKM '96, Proceedings of the Fifth International Conference on Information and Knowledge Management, November 12 - 16, 1996, Rockville, Maryland, USA*, pages 117–124. ACM, 1996.

[33] M. T. Özsy and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Englewood Cliffs, 1991.

[34] H. Pang, M. J. Carey, and M. Livny. Partially preemptive hash joins. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 59–68. ACM Press, 1993.

[35] V. Park and S. Corson. Temporally-ordered routing algorithm (tora) version 1 functional specication. Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-tora-spec-02.txt, 1999.

[36] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, Aug. 1994.

[37] C. E. Perkins. Ad hoc on demand distance vector (aodv) routing. Internet Draft, http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-04.txt, October 1999.

[38] G. J. Pottie and W. J. Kaiser. Embedding the Internet: wireless integrated network sensors. *Communications of the ACM*, 43(5):51–51, May 2000.

[39] V. Raman, B. Raman, and J. M. Hellerstein. Online dynamic reordering for interactive data processing. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 709–720. Morgan Kaufmann, 1999.

[40] R. Ramanathan and R. Rosales-Hain. Topology control of multihop wireless networks using transmit power adjustment. In *In Proceedings of the IEEE Infocom*, pages pages 404–413, March 2000.

[41] N. Schacham and J. Westcott. Future directions in packet radio architectures and protocols. *Proceedings of the IEEE*, 75(1):83–99, January 1987.

[42] P. Seshadri, M. Livny, and R. Ramakrishnan. Seq: A model for sequence databases. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan*, pages 232–239. IEEE Computer Society, 1995.

[43] P. Seshadri, M. Livny, and R. Ramakrishnan. The design and implementation of a sequence database system. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *VLDB'96, Proceedings of 22th International Conference on Very Large Data Bases, September 3-6, 1996, Mumbai (Bombay), India*, pages 99–110. Morgan Kaufmann, 1996.

[44] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. [1], pages 181–190.

[45] E. E. Systems. White dwarf linux. http://www.whitedwarflinux.org, 2001.

[46] G. Thomas, G. R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, and B. Hartman. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22(3):237–266, 1990.

[47] uClinux. The linux/microcontroller project. http://www.uclinux.org, 2001.

[48] T. Urhan and M. J. Franklin. XJoin: A reactively-scheduled pipelined join operator. *IEEE Data Engineering Bulletin*, Feb. 2000.

[49] T. Urhan, M. J. Franklin, and L. Amsaleg. Cost based query scrambling for initial delays. In Haas and Tiwary [14], pages 130–141.

[50] C. Yu and W. Meng. *Principles of Database Query Processing for Advanced Applications*. Morgan Kaufmann, San Francisco, 1998.

[51] C. T. Yu and C. C. Chang. Distributed query processing. *ACM Computing Surveys*, 16(4):399–433, Dec. 1984.

[52] C. Zaniolo, S. Ceri, C. Faloutsos, R. Snodgrass, R. Zicari, and V. S. Subrahmanian. *Advanced Database Systems*. Morgan Kauffmann Publishers, 1997.

[53] H. Zeller and J. Gray. An adaptive hash join algorithm for multiuser environments. In D. McLeod, R. Sacks-Davis, and H.-J. Schek, editors, *16th International Conference on Very Large Data Bases, August 13-16, 1990, Brisbane, Queensland, Australia, Proceedings*, pages 186–197. Morgan Kaufmann, 1990.

[54] W. Zhang and P.-Å. Larson. Dynamic memory adjustment for external mergesort. In M. Jarke, M. J. Carey, K. R. Dittrich, F. H. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 376–385. Morgan Kaufmann, 1997.