

NETWORKING SUPPORT FOR QUERY PROCESSING IN SENSOR NETWORKS

By ALEC WOO, SAM MADDEN, and RAMESH GOVINDAN

Networking and query processing must be co-designed to allow data self-organization for flexible but efficient in-network storage, access, and processing.

Sensor networks have the potential to support applications ranging from habitat and structural monitoring, to home and building automation, to supply chain management. Users are typically interested in continuous streams of information representing the evolving status of systems, combined with periodic statistical reports about specific phenomena. Query processing systems, including Directed Diffusion [3], TinyDB [6], and Cougar [10], provide high-level interfaces that allow users to collect and process such continuous streams. They are especially attractive as ways to efficiently implement monitoring applications without forcing users to write complex, low-level code for managing multihop network topologies or for acquiring samples from sensors.

Researchers are beginning to formulate languages and enumerate the types of queries needed by users of sensor networks [2]. Sensor networks also need distributed query processing, whereby the data is stored and retrieved from nodes within the network. The performance and functionality of query processing systems is significantly affected by internode communication mechanisms provided by the networking layer. However, today's systems have prompted the exploration of only a limited number of the networking mechanisms that will be needed by tomorrow's sensor network applications.

The research goal is the co-design of a query processing and networking subsystem to enable efficient

and scalable self-organized data retrieval and in-network processing in a reliable and timely fashion. In addition to the usual constraints (such as energy, computational power, and lossy wireless communication) in today's sensor networks, a unified query processing/networking system involves even greater challenges. Different applications have widely varying requirements in terms of information-transfer rates, timeliness, quality, and storage. Moreover, tension between optimizing network topology and improving the efficiency of query processing must be resolved; the most reliable network topology is often not the topology best able to promote in-network query processing. Finally, there is a substantial software engineering challenge, as these subsystems must be reusable in isolation from one another, despite the fact they were co-designed and optimized to work together.

Each sensor network is unique in that it involves strict but wide-ranging performance requirements and is notably difficult to program. In it, the query processing subsystem is in the unusual position of being able to drive research in novel networking mechanisms. Here, we take a first step toward identifying some of these directions.

In-Network Querying Systems

Several in-network querying systems have been built for sensor networks—with Diffusion (developed in 2000 by the Information Sciences Institute at the University of Southern California) being the pioneering

work. Here, we explore TinyDB and Diffusion; TinyDB and Cougar are similar in concept. In TinyDB [6], users specify declarative queries that reflect their data processing needs. Queries specify the types of readings (such as light and temperature), as

processor might apply a number of optimizations (without the user's direct input). For example, in the simple query just outlined, TinyDB can check the predicate on `light` or the predicate on `loc` first, depending on the energy required to acquire these attributes and the probability that the predicates over them succeed; one ordering may be much more preferable than the other.

Queries in TinyDB are flooded throughout the network. Query answers are collected via a routing tree, with the root node being the endpoint of the query (usually the user's physical location). Every other node maintains a parent node one step closer to the root from where it is, along with other routing information.

Figure 1 outlines the high-level architecture of TinyDB. Queries are input, parsed, and optimized at the user's PC, then injected into the tree-based sensor-network for processing. Results flow up and out of the network to the PC where they are displayed in one or more result interfaces and stored in a disk-based DBMS for later access.

Directed Diffusion [3] takes a slightly different approach. It does not specify a query language, instead allowing application writers to choose a domain-specific query processing language. Diffusion focuses on two other dimensions of query processing—query-routing mechanisms and flexible in-network processing—while providing a family of routing algorithms, all focusing on the idea that queries in the network are described by interest messages. Interests contain the particulars of the query, expressed through attribute-value pairs. For example, an interest message containing `location=[(100,100), (10,200)]`, `temperature=[10,20]` would return temperature data from all nodes within the specified region whose temperature is within the specified range. Such data is said to match the interest and is specified using attribute-value pairs. Thus, a data message that matches this interest might be: `location=[125,15]`, `temperature=13`.

An interest is originated by a “sink” node and distributed hop-by-hop throughout the sensor network. A node that lacks data matching an interest may forward the interest to its neighbors. The decision about to which location to forward the interest is based on cues provided within the data itself—a powerful idea called data-centric routing. In the temperature query example outlined earlier, the location attribute is the cue. As the interest is forwarded throughout the network, nodes build routing tables used to return data to the query originators, which can be human users or system components. A node with matching data (a “source”) can respond immediately.

These ideas are being used to design several routing

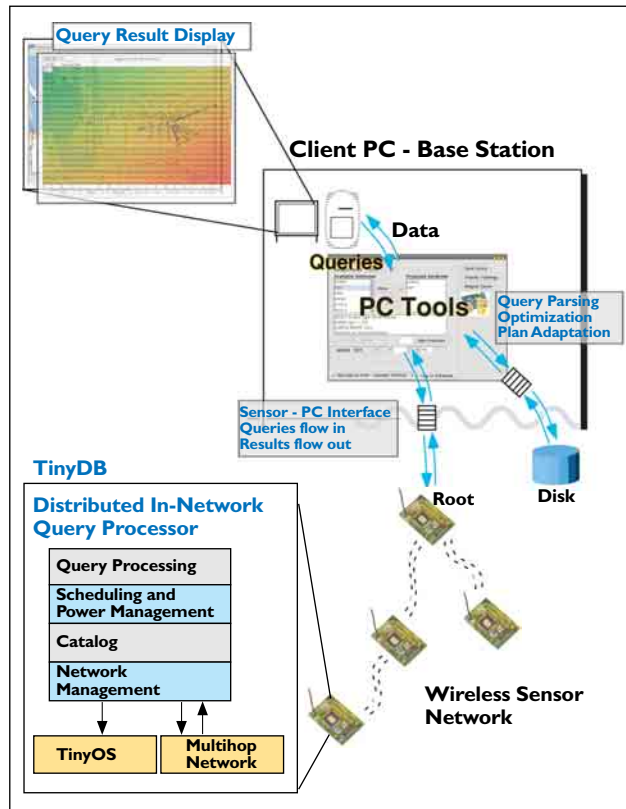


Figure 1. TinyDB architecture.

well as the subset of nodes of interest, along with simple transformations over the data. Queries are written in a SQL-like language; for example, to express the query “Report the average temperature of the bright nodes in the region defined by the rectangle (0,0,100,100),” a user would input the following query, along with a user-defined rate at which data is to be collected:

```
SELECT AVG(temp)
FROM sensors
WHERE loc in (0,0,100,100) and
light > 1000 lux
SAMPLE PERIOD 10 seconds
```

TinyDB queries are typically input on a PC that then sends the query into the sensor network. Within the network, the query executor must satisfy queries in a power- and communication-efficient manner using a variety of in-network processing techniques and cross-layer optimizations to report answers at the appropriate rates. Because the queries are specified in a high-level SQL-like language, the TinyDB query

Dimensions	Descriptions
Scope	Determines which nodes are involved in processing queries. Global scope involves the entire network. Geographical scope involves a particular geographic region. Logical scope involves nodes that satisfy a certain predicate (such as having a chemical sensor of a certain kind).
Volume	Communication costs per unit time of query responses. Large-volume queries involve high-fidelity or frequent sampling across many nodes. Low-volume queries need only low sample rates or aggregate answers.
Complexity	Multiple concurrent queries. Multiphase queries: Output of the first phase (such as computing the average) is required in the second phase (such as identifying outliers) Multiresolution queries: Sensor data is archived and queried at different levels of detail along either spatial or temporal dimensions.
Timeliness	Tolerance to delays between an event and when it is reported.
Quality or Accuracy	Quality of the query responses (such as percentage of missing readings) that can affect overall communication and systems resources.

Table 1. The design domain of networking mechanisms for query processing. Not all points are useful in light of sensor network constraints. Not addressed are design choices related to hardware platforms and deployment environments that might significantly affect query processor implementation. For example, a heterogeneous network topology, including a range of sensing platforms (such as motes and Hewlett-Packard iPAQ PDAs), may be implemented differently from a network consisting of only a single platform.

algorithms; for example, two-phase pull supports a small number of multiple sinks and a large number of sources; one-phase pull builds a tree for the common case of a single sink, while push supports applications with one source and many sinks.

Like TinyDB, Diffusion permits in-network processing; nodes aggregate or cache data items within the network to provide cues for interest-forwarding. Unlike

TinyDB, in which the set of available in-network operators is fixed by the query language, in-network processing in Diffusion requires application-specific code residing in network nodes.

TinyDB and Diffusion are thus conceptually similar, each providing a query-like interface for in-network processing. However, each also reflects a number of philosophical differences. TinyDB requires queries specified in a particular language using a particular set of operators, thus making it less general than Diffusion, while still enabling automatic query optimization. Query optimization is a result of the semantics of the operators in TinyDB being known by the query executor, allowing it to reorder certain operations or suppress the output of certain operators. Implementations of Diffusion also tend to give users more control over the types of network topology and patterns of communication than is provided by the current TinyDB implementation.

Significant practical advances are also being made in

networking support for query processing. The constraints faced in trying to move data and instructions around a network are illustrated in several recent measurement studies, including [9, 11]. They have firmly established the noisy nature of wireless connectivity in sensor networks; for much of the communication range of a node, a gray region exists where communication is unreliable and unpredictable, with both link quality and number of asymmetric radio links varying significantly in different physical environments. Thus, dynamically discovering and maintaining a set of reliable neighbors using a limited amount of resources is a challenge for the network architect. Moreover, shortest-path routing is likely to select long and unreliable links that are lossy and unpredictable and that require many retransmissions.

A routing layer implemented in [9] surmounts these challenges. The basic idea is that each node dynamically maintains a subset of frequently heard neighbors, allowing statistics to be collected for estimating current link quality. This information can then be used to build a stable, reliable, low-loss multihop routing tree that tends to require fewer transmissions, on average, than would be required in previously proposed shortest-path routing techniques. The routing layer in [9] has enabled various successful multihop deployments, including a 70-node TinyDB deployment in the University of California Botanical Garden in Berkeley, CA, and a 100-node deployment of a habitat monitoring application on Great Duck Island (off the coast of Maine) (see the article by Szewczyk et al. in this sec-

	Scope	Volume	Complexity	Timeliness and Quality
Query Processing Needs	Scope based on semantic value	Large and small	Multiple, multiphase, multiresolution	Selective: latency vs. accuracy
Current Networking Support	Broadcast (global or hop-count based)	Small	Single, simple selections, and aggregations	Best effort for every packet

Table 2. Query processing needs and current networking support for each design dimension.

tion). Link-quality estimators have also been used to avoid using asymmetric or unreliable links in several Diffusion-based applications deployed at the James San Jacinto Mountains Reserve Wildlife Observatory (Idyllwild, CA) run by the University of California.

Novel Networking Mechanisms

TinyDB and Diffusion are relatively mature research prototypes that give some idea of how future sensor network query processing systems will function. However, these future systems will be significantly more sophisticated than any of today's prototypes, despite involving many novel networking requirements. To understand these requirements, queries may be classi-

fied along five dimensions—scope, volume, complexity, timeliness, and quality—that dictate the design of networking mechanisms for query processing (see Table 1). These dimensions demarcate the types of design options for networking support of query processing in sensor networks.

This classification shows the existing systems (such as TinyDB) focus on a single design point: globally scoped queries issuing from outside the network for low-volume responses, along with best-effort timeliness and response quality. This observation motivates our central thesis—that networking for query processing is in its infancy (see Table 2). Substantiating it are two emerging directions in networking research—query-informed routing and efficient rendezvous for storage and correlation. They also prompt a more general question: How do network architects design a unified networking infrastructure that supports these needs, in the same way the TCP/IP stack supports the vast variety and large number of Internet applications?

Query-informed routing. Today’s query processing systems use tree-like routing structures to resolve queries. One emerging research direction is to consider techniques that influence the construction of the trees by leveraging the characteristics of the query. One idea in this direction is semantic broadcast. Recall that some queries may inherently specify a limited logical scope. Rather than flooding the entire network, the querying system and the networking layer might instead coordinate to provide efficient data dissemination and semantically scope floods. A coordinated approach requires the query system to define the policies (such as to which nodes to send data and queries) on top of the mechanisms provided by the networking layer. During a query flood, the query processor must specify whether a message should be forwarded, as well as which nodes (if any) it should be forwarded to. These two policies thus define the scope of the flood.

A query system may define this policy so query messages are delivered only to nodes that produce results for a particular query. For example, the notion of a semantic routing tree [5] allows query dissemination to be scoped to nodes whose readings are within a particular range, avoiding unnecessary query forwarding and reducing flooding overhead. An example, developed by [12] for target tracking, is to discover querying paths to nodes close to the target by optimizing an objective function that balances the usefulness of the sensor data and the corresponding communication costs along the paths.

Another example of query-informed routing is when the query processor influences network topology formation to optimize in-network processing. It allows the application to achieve timeliness, support

multiple concurrent queries, and trade off quality for greater energy savings. These added benefits contrast with the common approach cited in the sensor network literature (such as [9]) where trees are optimized for reliable, shortest-path delivery. Surprisingly, it may be better for the query processing system to choose a less-reliable routing path that achieves more aggregation or correlation opportunities, as such paths can reduce the overall transmission load on the network. An ideal routing tree would thus be able to exploit in-network processing as much as possible while still deliv-

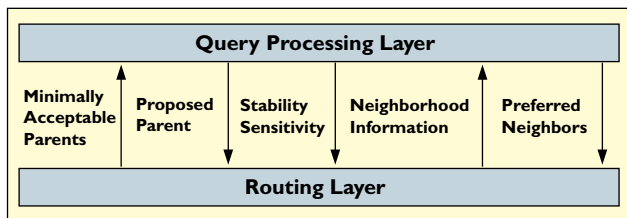


Figure 2. The interface between the sensor network and the query processor requires flexibility for optimal query processing and networking co-design.

ering the end results to their destinations.

A query processing system designed to build a tree optimized for in-network processing must influence parent selection at the routing layer. Such a routing policy would consider both the semantic information from the query (such as information about the locations most likely to produce data that can be aggregated) and link-layer reliability and connectivity properties learned from neighboring nodes.

A related issue concerns the scheduling of network adaptation. Traditional ad hoc routing schemes adapt only to network changes (such as link failures and connectivity fluctuations). But for query-sensitive routing, the topology must also change in response to variations in the query load and data distribution. The networking layer must also retain sole responsibility for selecting the correct route; input from the query processor is only a hint the networking layer might consider after ensuring algorithmic correctness, that, say, the route is under some maximum length or is cycle-free.

A final key issue the routing layer must consider is how the application and the network itself might balance the stability and speed of topology adaptation, or agility. Many query processing algorithms perform better when the topology is relatively stable, as they must rebuild data structures as the topology changes. If the topology changes too frequently, the benefit of building a topology for in-network processing can be

overwhelmed by additional maintenance costs. However, some adaptation is always necessary, as failure to adapt eventually results in network topologies that are unable to route data properly.

The tight relationship between the routing layer and the query processing layer represents a radical departure from the traditional networking literature, which focuses on careful layering of functionality and isolation between layers. In the sensor network context, energy constraints dictate that system software must be designed using porous layering (see Figure 2)

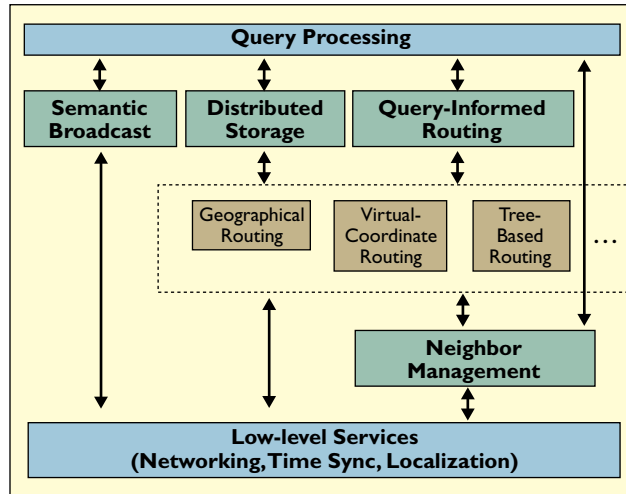


Figure 3. A unified query processing framework for sensor networks. A practical instantiation must also provide secure storage, routing, and query processing.

that allows more control information to permeate throughout the network to improve system efficiency.

Efficient rendezvous for storage and correlation. The second thread of research is motivated by the following observation: Given that communication is expensive, it is reasonable to assume that data generated by sensors is stored within the network and that queries go searching for and “find” matching data. In existing sensor networks, queries are flooded throughout the network. As sensor networks scale to many simultaneous queries or when queries are short-lived, the overhead in terms of energy and network capacity of flooding can be significant.

In the simple tree-based topologies we have considered thus far, query flooding is necessary to ensure the query originator rendezvous with nodes that might contain data matching the query. However, query flooding is not the only way to ensure rendezvous. To illustrate an alternative approach, consider the following simplified example. Suppose a node **D** in a network wishes to find all data matching a predicate P (an example of such a predicate might be temperature $>50^\circ$, though more sophisticated predicates are possible). Suppose, too, that nodes **A** and **B** have data that matches P . An efficient rendezvous method for these

nodes would be to store the node’s data at a node **C** whose identity was determined entirely by some function of P . **D** could then compute that same function and retrieve the data directly from **C** without having to flood the network.

One approach to achieving this efficient rendezvous is to map, or hash, the predicate P to a geographic location L and store data matching the predicate at the node whose location is closest to L . The querying node could then independently compute L and retrieve the data. This approach relies on efficient geographic routing protocols that depend either on nodes having physical location information or on nodes having the ability to determine their locations in a virtual coordinate system.

The ability to rendezvous efficiently within the network can be extended to the development of hierarchical storage structures that allow more sophisticated queries. For example, it is possible for the query processing system to efficiently support multidimensional range queries of the form: List all events whose temperature is between 50° and 60° , and whose light level is between 1,000 and 1,500 lux using essentially a distributed index. Such an index is built by storing similar data, or data close to other data in attribute space, at the same node. It is also possible to build a hierarchy of nodes within the network to store progressively higher-resolution sensor data at lower levels of the hierarchy. Such a distributed data structure allows efficient multiresolution queries along either the spatial or the temporal dimension.

While ongoing research [1, 4, 8] is examining strands of this thread, much work remains to be done. For example, in sensor networks with many distributed data structures, query optimization (determining the optimal order of access to and location of data structures) and storage management represent especially interesting challenges. Another challenge is robustness, or how to replicate stored data to increase availability and avoid hotspots that receive many requests. Finally, this research requires geographic routing protocols able to localize nodes; several virtual coordinate systems (see [7] for an example) have been proposed to provide this functionality, though it is unclear whether these systems are useful for reliably storing data.

Toward a Unified In-Network System

Future query processing systems will require networking mechanisms (such as query-informed routing) that provide greater system efficiency or enable new functionality (such as efficient rendezvous for distributed storage). However, what’s not clear today is whether all applications will be able to leverage all these new networking mechanisms (or even all possible query types).

Application-specific dependency, together with the need to reuse code, suggests that future sensor network query processing systems will be structured in a modular fashion like the one outlined in Figure 3. Because different applications will use subsets of these modules and mechanisms, a key challenge is to devise the right set of APIs to allow maximum code reuse and flexibility.

Figure 3 also highlights an important architectural trend. Unlike in traditional networked systems, the line between the higher levels of the architecture (such as the query processor) and the lower levels of the network stack is blurred. Unlike the narrow, “hourglass” API of traditional network stacks, the API is complex, allowing the query processor to inform networking decisions about network topology and the proper allocation of in-network storage while exposing routing information from the network layer up to the query processor.

This requirement is dictated by the most daunting constraints faced by sensor networks. Energy efficiency requires squeezing as much performance as possible out of the network. One way to do it is to leverage information from the query processor.

Interestingly, the narrowest API in the sensor network domain may be between the query processor and the application; the easy-to-use, high-level nature of queries provides a convenient and simple interface for application developers to isolate them from many of the more arcane aspects of the sensor network domain.

A related trend (reflected in many other APIs) is the separation of mechanism at the lower levels from policy specified at the higher levels by the query processor (and user queries). Separating mechanism from policy in a sensor network makes it possible for the storage and topology-formation layers to be used for multiple purposes. Default policies help combat the increased complexity introduced by the richness of mechanism and policy APIs. Separation also makes it possible for multiple, competing query processing solutions (including TinyDB and Diffusion) to exist without replicating substantial pieces of one another’s functionality.

Conclusion

In-network systems must support multiple complex queries and rich communication patterns beyond the basic tree-based data collection used by query processing systems today. Needed for delivering this advanced functionality are more sophisticated topology-construction algorithms and facilities for in-network storage and correlation than are available in traditional computer networks. We have thus described a basic architecture for these next-generation distributed sen-

sor network systems and highlighted some of the key aspects of API design in this domain. **C**

REFERENCES

1. Ganesan, D., Greenstein, B., Perelyubskiy, D., Estrin, D., and Heidemann, J. An evaluation of multiresolution search and storage in resource-constrained sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)* (Los Angeles, Nov. 5–7). ACM Press, New York, 2003, 89–102.
2. Gehrke, J. and Madden, S. Query processing in sensor networks. *IEEE Pervasive Comput.* 3, 1 (Jan. 2004), 46–55.
3. Intanagonwiwat, C., Govindan, R., and Estrin, D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MobiCOM)* (Boston, Aug. 6–11). ACM Press, New York, 2000, 56–67.
4. Li, X., Kim, Y., Govindan, R., and Hong, W. Multidimensional range queries in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)* (Los Angeles, CA, Nov. 5–7). ACM Press, New York, 2003, 63–75.
5. Madden, S., Franklin, M., Hellerstein, J., and Hong, W. The design of an acquisitional query processor for sensor networks. In *Proceedings of ACM SIGMOD 2003* (San Diego, June 9–12). ACM Press, New York, 2003, 491–502.
6. Madden, S., Hong, W., Hellerstein, J., and Franklin, M. *TinyDB: A Declarative Database for Sensor Networks*; see telegraph.cs.berkeley.edu/tinydb.
7. Rao, A., Papadimitriou, C., Ratnasamy, S., Shenker, S., and Stoica, I. Geographic routing without location information. In *Proceedings of the 9th International Conference on Mobile Computing and Networking (MobiCOM)* (San Diego, Sept. 16–18). ACM Press, New York, 2003, 96–108.
8. Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., and Shenker, S. GHT: A geographic hash table for data-centric storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks* (Atlanta, Sept. 28). ACM Press, New York, 2002, 78–87.
9. Woo, W., Tong, T., and Culler, D. Taming the underlying challenges for reliable multihop routing in sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)* (Los Angeles, Nov. 5–7). ACM Press, New York, 2003, 14–27.
10. Yao, Y. and Gehrke, J. Query processing in sensor networks. In *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research* (Asilomar, CA, Jan. 2003).
11. Zhao, J. and Govindan, R. Understanding packet delivery performance in dense wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)* (Los Angeles, Nov. 5–7). ACM Press, New York, 2003, 1–13.
12. Zhao, F., Liu, J., Liu, J., Guibas, L., and Reich, J. Collaborative signal and information processing: An information-directed approach. *Proceed. IEEE* 91, 8 (Aug. 2003), 1199–1209.

ALEC WOO (awoo@cs.berkeley.edu) is a Ph.D. candidate in the Computer Science Division at the University of California, Berkeley.
SAMUEL MADDEN (madden@csail.mit.edu) is an assistant professor in the Electrical Engineering and Computer Science Department at MIT and a member of the MIT Computer Science and Artificial Intelligence Laboratory in Cambridge, MA.
RAMESH GOVINDAN (ramesh@usc.edu) is an associate professor in the Computer Science Department at the University of Southern California in Los Angeles.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.