INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS
Int. J. Commun. Syst. 2007; **20**:889–908
Published online 18 September 2006 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/dac.850

A taxonomy of distributed query management techniques for wireless sensor networks

S. Chatterjea*,† and P. Havinga‡

Department of Computer Science, University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands

SUMMARY

In the not too distant future, wireless sensor networks are envisioned to proliferate through the entire spectrum of the 'environmental monitoring' market allowing users to monitor a multitude of environments. Thousands or even millions of sensor nodes may span vast geographical areas enabling various environmental parameters to be monitored with significantly higher spatial and temporal resolutions than what is achievable using existing monitoring technologies. In order to manage the large amount of data that will be generated by these numerous sensor nodes, novel querying methods are needed to extract the required information in an energy-efficient manner. This paper studies the techniques used to manage the queries in a distributed manner and classifies the current state-of-the-art in this field into four main categories: in-network processing, acquisitional query processing, cross-layer optimization and data-centric data/query dissemination. This taxonomy not only illustrates how query management techniques have advanced over the recent past, but also allows researchers to identify the relevant features when designing sensor networks for different applications. Copyright © 2006 John Wiley & Sons, Ltd.

Received 20 January 2006; Revised 15 June 2006; Accepted 30 June 2006

KEY WORDS: distributed query processing; sensor networks

1. INTRODUCTION

The last few years have seen the field of wireless sensor networks (WSNs) transform into a multi-faceted research area spanning across a wide range of disciplines within the field of computer science. With the pioneering work focussing on the hardware aspects, the data-centric nature of WSNs and various communication-protocol-related issues, sensor network research is gradually branching out into other diverse fields.

*Correspondence to: S. Chatterjea, Department of Computer Science, University of Twente, P.O. Box 217, 7500AE Enschede, The Netherlands.
†E-mail: supriyo@cs.utwente.nl
‡E-mail: havinga@cs.utwente.nl

Researchers from the database community have added yet another interesting perspective by introducing the idea of viewing a WSN from the point of view of a database management system. The primary motivation for this approach is to help create an abstraction between the end-user and the sensor nodes thus allowing the user to solely concentrate on the data that needs to be collected rather than bothering with the intricacies of mechanisms deciding *how* to extract data from a network in the most energy-efficient manner.

While there are many concepts from the field of databases that can be beneficial to the area of sensor networks, the unique nature of sensor networks gives rise to a whole new set of issues that makes it inappropriate to apply concepts from databases directly. However, it is possible to adapt existing database solutions to a certain extent to suit the requirements of WSNs.

This paper introduces the reader to the various distributed query management techniques designed specifically for WSNs proposed thus far and classify them into specific categories. After giving a brief introduction describing the characteristics of WSNs we highlight the four essential building blocks which constitute a distributed query management framework for WSNs, Table I. We then go on to describe the state-of-the-art for each category and discuss the salient points of the various query management methods. We conclude the paper in Section 4.

2. CHARACTERISTICS OF WSNS

Wireless sensor networks are typically made up of battery-powered nodes that have built-in wireless transceivers. The nodes usually have a small form-factor and can have a host of sensors attached to them depending on the application they will be used for. Instead of simply having sensors equipped with wireless transceivers transmitting directly to a base station, what differentiates wireless sensor nodes is the fact that every node possesses a small amount of computation capability due to the presence of an on-board CPU running at a few MHz and limited memory (Figure 1). The nodes can be deployed in a dense network and communicate with one another via a multi-hop network. This helps to minimize energy consumption and allows the network to cover a large geographic area. The high-density deployment allows WSNs to monitor environments at extremely high spatial and temporal resolutions, which are impossible using conventional monitoring techniques.

While the limited source of energy from the battery contributes to severe *power constraints*, WSNs have a number of other properties, which make the problem of distributed data management even more challenging. Regardless of which WSN application scenario might be considered, such as various forms of environmental monitoring, asset tracking, and surveillance, etc., sensor nodes need to be completely *self-configuring*. This implies that whether they have been deployed randomly or in an ordered manner, with a static or dynamic network topology, the nodes should be able to configure themselves automatically during network start-up and begin performing their intended task the moment they are powered up without anyone having to tend to each node individually. Furthermore, once operation begins, nodes need to be able to adjust their operation on-the-fly to ensure that they continue to operate in an energy-efficient manner that best suits the current dynamics of the network. It should be noted that the self-configuring property is not only vital for the efficient operation of networking-related attributes, but also for the distributed query management components residing on a node. These components should be able to adapt their decision-making operations to suit the dynamics of the network, its usage patterns and the environment being monitored.

DOI: 10.1002/dac

Table I. A summary of distributed data management techniques for WSNs.

Name of the project/mechanism author (a)	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimization	Data-centric query/data dissemination	Simulation	Mote-class	PDA-class	PC-class
Directed Diffusion [1], 1999	• Uses filters • Suppression of duplicate messages	x	x	• Publish/ subscribe API based on named data • Gradients used to route data from source to sink		✓	✓	
COUGAR [2], 2000	• Aggregation at cluster leaders • Packet merging	x	x	x			✓	
TinyDB [3], 2002	• Evaluation of operators such as MIN, MAX, SUM, COUNT, AVERAGE, etc.	• Centralized query optimization based on collected metadata • Priority given to cheapest sensor when evaluating multiple predicates • Multi-query optimization: Grouping of multiple identical events	• Communication scheduling primarily for upstream data flow • Nodes able to decide on schedule themselves using hop count • Minimizes burden on MAC • Scheme unusable for multiple concurrent queries with different epoch requirements	• Semantic routing • Designed for range queries • Maintenance algorithm may be too costly if measured attribute changes too rapidly	✓	✓		
TiNA [4], 2003	• Exploits temporal correlations • Readings transmitted only when user-specified threshold is exceeded	x	x	x	✓			
Deligiannakis *et al*. [5], 2004	• Transmits approximated readings generated using base signals	x	x	x	✓			✓
WaveScheduling [6], 2004	x	x	• Nodes scheduled to minimize MAC collisions • Saves energy, but increases latency	x	✓			

DOI: 10.1002/dac

Table I. *Continued.*

Name of the project/mechanism author (a)	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimization	Data-centric query/data dissemination	Simulation	Mote-class	PDA-class	PC-class
Bonfils *et al.*, [7] 2003	• Decentralized query operator placement	x	x	x	✓			
Coman *et al.* [8], 2005	x	x	x	• Queries forwarded based on spatial and temporal ranges	✓			
REED [9], 2005	• Joins between static, external tables and sensor nodes	x	x	x	✓	✓		
BBQ [10], 2004	x	• Minimize sampling by predicting sensor readings using centralised statistical models • Identify temporal and spatial correlations	x	x	✓			
Ken [11], 2006	• Nodes keep local model synchronized with central model • Able to detect outliers	x	x	x	✓			
TD-DES [12], 2003	x	x	• Event scheduler dynamically allocates and multiplexes upstream and downstream time slots for each event type • Root works out schedule	x	✓			
DTA [13], 2004	x	x	• DTA defines rules controlling order of data transmission • Root works out schedule	x	✓			✓
AI-LMAC [14], 2004	x	x	• Adapts operation of MAC based on requirements of application	• Directs queries only to relevant parts of the network	✓			

DOI: 10.1002/dac

Table I. *Continued.*

Name of the project/mechanism author (b)	Characteristics							
	Essential conceptual features				Platform			
	In-network processing	Acquisitional query processing	Cross-layer optimization	Data-centric query/data dissemination	Simu-lation	Mote-class	PDA-class	PC-class
			• Distributed bandwidth allocation based on expected traffic for a particular query • MAC operates with low duty-cycle in inactive areas of network	• Queries directed based on attribute name and value				
GHT [15], 2002	x	x	x	• Hashes attribute name into geographic co-ordinates • Nodes must be location aware • Uses perimeter refresh protocol to improve robustness • Uses structured replication to eliminate bottlenecks during occurrence of multiple identical events	✓			✓
DIFS [16], 2003	x	x	x	• Supports range queries • Hashes attribute name and value into geographic co-ordinates • Nodes must be location aware • Does not deal adequately with node failures and packet losses	✓			
DIM [17], 2003	x	x	x	• Supports multiple range queries • Hashes event to a zone • Nodes must be location aware • Re-assigns zones when nodes enter/leave • ACK scheme improves robustness			✓	✓

DOI: 10.1002/dac

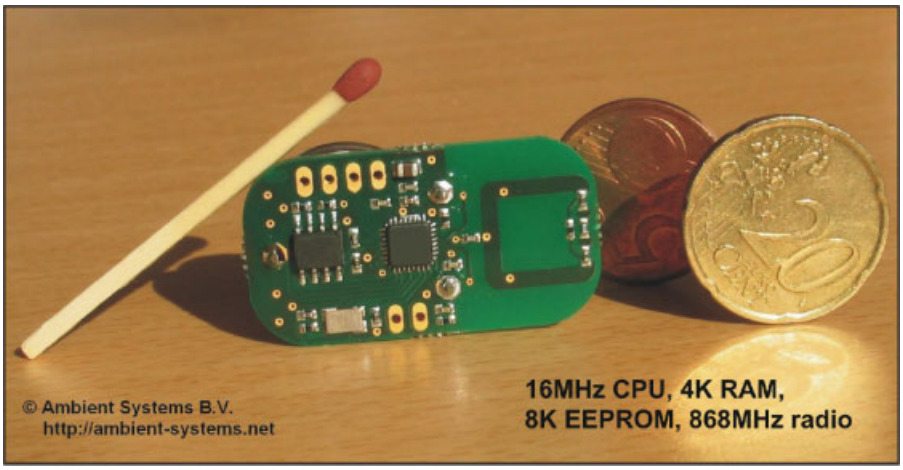

Figure 1. An example of a typical wireless sensor node.

With the current state of technology, sensor nodes are notorious for their unreliability. For example, the signal strength of nodes varies greatly with time due to a host of parameters, e.g. varying environmental conditions, noisy radio channels, etc. The sensors attached to nodes also contribute to the system's unreliability as they are prone to Byzantine errors especially when battery levels run low or the nodes are exposed to hostile environmental conditions. The degree of impact of such errors can be quite substantial if the level of acceptable fault tolerance defined by the user is set to a low threshold. Thus it is imperative to ensure that *reliability* is taken into account when developing data management schemes.

3. ESSENTIAL CONCEPTUAL BUILDING BLOCKS

While there are a host of features that are necessary to form a full-fledged distributed data management system for WSNs, we highlight four essential conceptual building blocks in Figure 2 that have been widely mentioned in the existing literature. Below, we provide a very brief overview of each building block and describe their specific roles. This is followed by a discussion detailing how the various components are related to each other thus enabling the reader to visualize how the various components fit into the 'bigger picture'.

- In-network processing
  This involves moving various types of computation that are traditionally done on the server-side to within the sensor network itself. It is generally used for filtering and processing of messages flowing within the network thus preventing the transmission of unnecessary information.
- Acquisitional query processing
  Energy consumption in sensor nodes depends on two main factors: Operation of the transceiver and operation of the sensors. Acquisitional query processing helps minimize energy consumption by targeting the sensors, i.e. sampling of the various attached sensors is carried out in an energy-efficient manner. For example, a user may be presented with sensor readings generated using certain statistical methods rather than actually sampling sensors.

DOI: 10.1002/dac

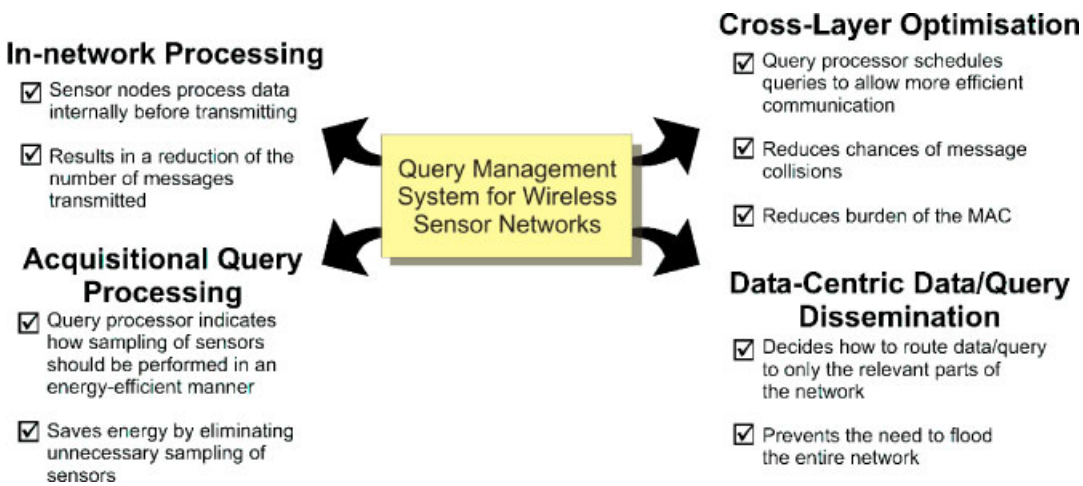

Figure 2. Essential building blocks of a distributed query management system for wireless sensor networks.

- Cross-layer optimization
  Unlike conventional computer networks which can generally be used to perform a wide variety of tasks, WSNs are usually designed for a particular application. This makes it possible to design the various components of the WSN architecture, e.g. the routing and MAC protocols specifically for the application in mind. This could mean that the MAC and routing protocols may be able to adapt to the changing requirements of the application. This is fundamentally different from the conventional OSI model used for typical networks where the lower-layer protocols operate completely independently from the higher layer protocols.
- Data-centric data/query dissemination
  Unlike conventional routing protocols which do not actually bother about the content of the data message being transmitted, the path taken by a message being routed by a data-centric data/query dissemination protocol for a WSN is completely dependent on the contents of the message. This allows messages to be routed more efficiently.

While Figure 2 illustrates the pertinent features of every building block, it does not illustrate the relationship between them. We present this relationship in Figure 3. Probably the most noticeable feature is that we have placed acquisitional query processing, cross-layer optimization and data-centric data/query dissemination all within the class of in-network processing. The reason for this can be traced back to the way data is usually collected in conventional databases using the *warehousing* approach. Under this model, data is initially collected from various sources (e.g. sensors with wireless transmitters) and stored at a central location. The data is then processed centrally to extract the required information. This model is highly unsuitable for WSNs as it involves excessive transmission overhead, and also prevents users from accessing real-time, streaming data. The only viable alternative is to migrate from off-line processing to processing the data within the network as close to the data source as possible. The practice of processing data at the sensor nodes themselves is known as *in-network*

DOI: 10.1002/dac

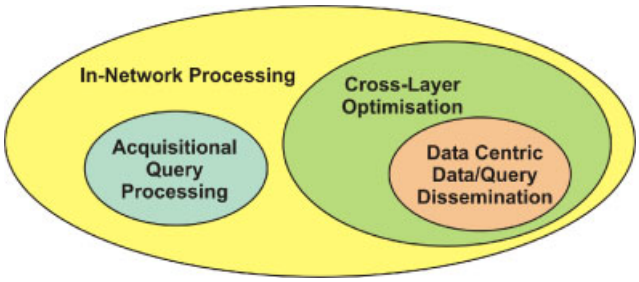

Figure 3. Relationship between the building blocks.

processing and can result in a significant reduction in the number of messages transmitted. The ability to perform in-network processing is a cornerstone of WSNs.

In order to distinguish between sensors attached with wireless transmitters and wireless sensor nodes, which are actually capable of performing simple computations within them, and also because in-network processing is a very generic term (i.e. it could essentially refer to any sort of processing that takes place within a node) we have placed acquisitional query processing, cross-layer optimization and data-centric data/query dissemination all within the class of in-network processing. All these three sub-classes require a sensor node to analyse the data it has at hand and make certain decisions based on the outcome of the analysis instead of transmitting all the data to a central server for off-line processing or centralized decision-making. Such processing is not possible in normal sensors that only have attached wireless transmitters. The kind of processing that can take place within a wireless sensor node varies greatly, which is why we have these three other sub-categories.

Cross-layer optimization generally refers to schemes where the application influences the operation of the routing or MAC layers. Data-centric data/query dissemination is classified as a sub-class of cross-layer optimization as it involves creating links between the application and routing layers. In the cross-layer optimization category, we cover literature which studies the relationship between the MAC and the application layers.

In order to simplify matters, and also because the features present in a particular class are not present in its super class, we analyse each building block separately in the following subsections. However, the reader should keep in mind the existing relationships indicated in Figure 3. The following subsections describe each category in greater detail and illustrate the different methods that have been employed to incorporate the various features.

3.1. In-network processing

While the precise manner in which in-network processing is carried out on sensor nodes may differ between various publications, the fundamental objective is still the same — to save energy by reducing message transmissions. Directed diffusion [1] performs in-network processing using *filters*. When a node receives a message that matches its filter, the message is first handed over to the application module within the node for processing instead of forwarding it to the next node. For example, the application might carry out a suppression of duplicate messages indicating the detection of the occurrence of an event so as to prevent a sudden burst of identical messages when a bunch of nodes detect the same event. The main drawback of directed diffusion

DOI: 10.1002/dac

however, is that it has a non-declarative interface, and thus does not rank very highly in terms of usability.

COUGAR [2], which was one of the first projects to view a sensor network as a distributed database with a declarative interface, uses a clustered approach to in-network processing. A network may consist of several clusters, each of which is made up of a single leader node and a group of child sensor nodes belonging to the leader. Child nodes periodically send their readings to the leader node which then aggregates the received readings and only continues to forward the computed result toward the root of the network. Computation at the leader only takes place once all the child nodes have responded. Additionally, since sending multiple small packets is more expensive than sending one larger packet (due to the packet header payload and the cost of reserving the medium) COUGAR performs *packet merging* by combining several packets into one. This method is particularly beneficial when servicing queries that generate holistic aggregates where intermediate nodes cannot perform any partial aggregation and all data must be brought together to be aggregated by the node evaluating the query, e.g. the median operator or even the collection of raw sensor readings. It is important to note however, that while COUGAR claims to be designed for WSNs, it was deployed on PDA-class devices that had significantly larger processing power and could even run Windows CE and Linux [18]. Their design does not consider the impoverished power and computational constraints of conventional sensor nodes, e.g. XML, which is known for its verbosity, is used to encode messages. Apart from packet merging, Cougar lacks any other features that make it energy efficient. It also fails to address the issues of reliability and self-organization and relies instead on the underlying 802.11 MAC protocol.

TinyDB [3] supports a number of aggregation operations (e.g. MIN, MAX, SUM, COUNT, AVERAGE, etc.) over certain user-specified sample intervals. As sensor readings flow up the communication tree, they are aggregated by intermediate nodes that are able to meet the requirements of the query (Figure 4). Without aggregation, every node in the network needs to transmit not only its own reading, but also those of all its children. This causes a bottleneck close to the root node, and also results in unequal energy consumption, i.e. the closer a node is to the root node, the larger the number of messages it needs to transmit which naturally results in higher energy consumption. Thus, nodes closer to the root node die earlier. Losing nodes closer to the root node can have disastrous consequences on the network due to network partitioning. Using in-network aggregation however, every intermediate node aggregates its own reading with that of its children and eventually transmits only one combined result. This also naturally implies that the size of the message remains constant as it traverses from the source nodes to the root. TinyDB also illustrates how aggregation can be extended to perform more complex tasks such as vehicle tracking and isobar mapping [19]. TinyDB depends on MAC-level acknowledgements and runs a tree-maintenance algorithm continuously in order to ensure that communication between nodes can continue seamlessly at all times.

The temporal coherency-aware in-network aggregation (TiNA) [4] scheme improves upon TinyDB as it allows users to specify a threshold which in turn is used to reduce the amount of information that is transmitted by individual nodes. Instead of transmitting every reading at fixed intervals (like in TinyDB), TiNA exploits temporal correlations of readings to suppress certain messages, i.e. if the newly acquired reading differs from the previous reading by an amount that is less than the user-specified threshold, the reading is not transmitted. In order to ensure reliability, nodes in TiNA send periodic *heartbeat* messages to their parent nodes so that

DOI: 10.1002/dac

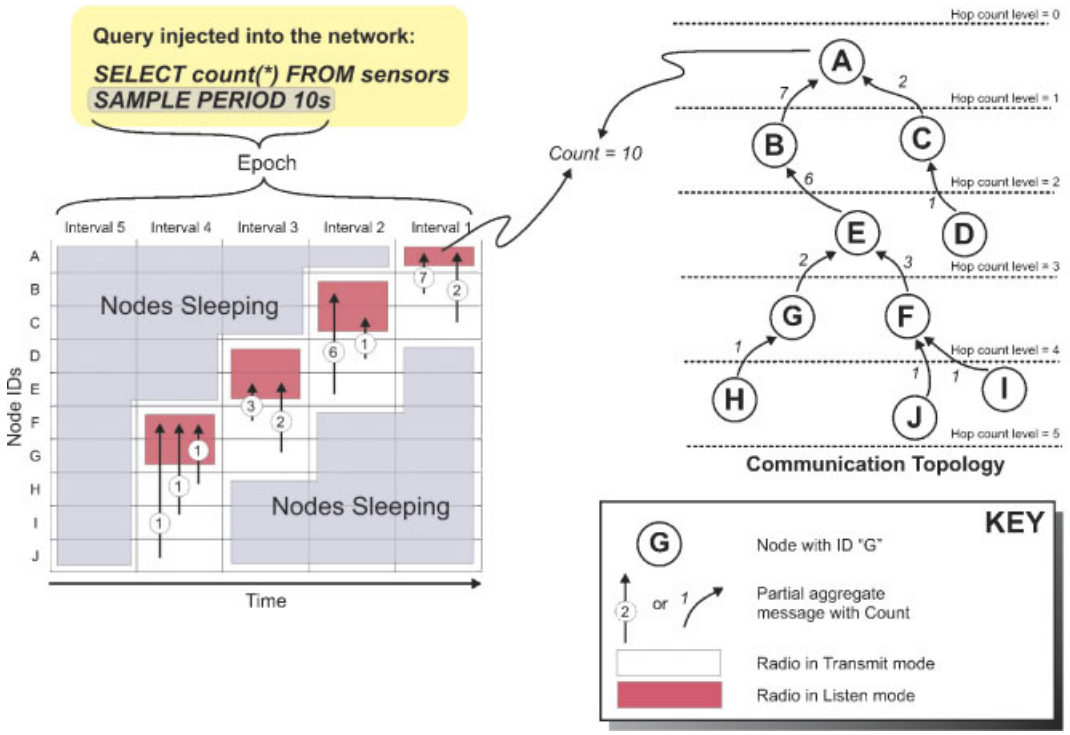

Figure 4. An aggregation operation using interval-based communication scheduling [4].

the parent node knows that the child node is alive during periods when the acquired readings are fairly constant and fall within the user-specified threshold.

Similar to TiNA, Deligiannakis *et al.* [5] also present a scheme where approximated readings are transmitted to the querying node thus resulting in energy savings. The main idea is to first generate a *base signal* that captures the prominent features of the sampled data. The acquired data is subsequently partitioned into intervals that can be efficiently approximated as some linear projections of some part of the base signal. While their scheme presents better accuracy and robustness when compared to other approximation schemes, the feasibility of their scheme is questionable as it was tested on a computer with a 300 MHz processor, which is nearly 19 times faster than the sensor node shown in Figure 1.

Bonfils and Bonnet [7] describe another form of in-network processing that allows a node to run a completely localized algorithm (based on information from one-hop neighbours) that ensures that in-network processing is carried out in an efficient manner. This is done by placing aggregation operators such that the amount of data that needs to be transmitted is minimized. Once the operators have been placed on an arbitrary node, the localized algorithm ensures that the operator progressively reaches its local optimal placement by greedily moving to the neighbour with the lowest estimated cost. This method is particularly suited for handling holistic aggregates. It is also useful for carrying out join operations between tables stored on pairs of

DOI: 10.1002/dac

sensors in a network. While the work allows nodes to take decisions in a decentralized manner, there is no mention of any fault tolerant operator placement algorithms.

Rather than focussing on joins between pairs of sensors, REED (*robust, efficient filtering and event detection in sensor networks*) [9] concentrates on joins between static external tables (i.e. tables stored outside the network) storing filter conditions and sensor nodes within the network. External tables are typically injected into the network along with the query. If a receiving node is able to store the entire table in its limited memory, the node performs a join operation and returns the result to the root node. In the event that the external table is too large to be accommodated within a single node, REED partitions the table horizontally into smaller fragments. The fragments are then disseminated to a group of nodes (which are within close proximity of one another) such that each node in the group stores a single fragment. The join operation is performed within the group and the result is subsequently returned to the root. REED can also transmit fragments of the table into the network, forcing nodes which do not have entries in the table to be joined externally. Timeouts and periodic advertisements are used to improve the robustness of the system.

While the majority of the literature in this category focuses on utilizing different strategies to ensure energy-efficient operation, none of the authors investigate the performance of unreliable message transmissions. As current sensor nodes are known to be unreliable and the performance of the various in-network processing schemes can change dramatically with varying degrees of reliability, this remains as a topic which is open to further research.

3.2. Acquisitional query processing

In conventional database systems, when posing a query to the system, a user need not bother about how the data should be extracted in the most efficient way. The same concept holds true in distributed database systems where the requested data might be stored in small fragments spanning the entire network. In this case, the user does not require knowledge of where the data resides. In other words, the techniques used to locate data or the methods followed to extract data in an efficient manner, are processes that are completely transparent to the user. The user simply injects the query into the network and waits for the result to appear. It is the responsibility of the query processing system to handle the above-mentioned tasks and act as an interface between the user and the data sources.

One of the tasks of a query processor for WSNs is to generate an optimized query execution plan that outlines an energy-efficient strategy to execute a query. While conventional database query optimization techniques calculate the cost of executing a query based on a number of parameters such as CPU instructions, I/O operations, messages transmitted, etc. the model necessary for WSNs is slightly different due to its unique characteristics described earlier. The tight power constraints of WSNs have driven researchers to find novel ways to minimize the two main sources of power consumption on a sensor node — the operation of the radio transceiver and the sampling of the sensors to obtain readings. The idea of managing sensor sampling as one of the tasks of the query processor was first introduced in TinyDB [3] and was termed as *acquisitional query processing*.

TinyDB carries out query optimization at the root node. Metadata such as the energy and time required to sample a particular sensor, information about the costs of processing and delivering data, etc. are periodically copied from the nodes to the root for use by the query

DOI: 10.1002/dac

optimizer. The optimizer also has the task of initializing, merging and updating the final value of partial aggregate records as they propagate through the network towards the root.

Before a query is injected into the network through the root node, the query optimizer can optimize the query in various ways using the collected metadata. For example, since the difference in power consumption of sampling various sensors on a single node can differ by several orders of magnitude, the order in which sensors are sampled when evaluating a query can have a substantial impact on network lifetime.

Consider the scenario where a user requires readings from a light sensor when the temperature and humidity sensors reach thresholds t and h, respectively. If the node uses a humidity sensor that requires 100 times more energy to sample than the temperature sensor, in most cases, it would be a lot more energy efficient to sample the temperature sensor first to check if the threshold has been met and proceed with the sampling of the humidity sensor *only* if the result of the temperature threshold check is found to be *True*. In fact, in such a scenario, sampling the humidity sensor first could be up to an order of magnitude more expensive.

TinyDB also performs multi-query optimization on event-based queries in order to reduce costs due to sensor sampling and transmission. Consider a query Q that requests temperature readings only when a certain event E occurs. The occurrence of a string of event Es within a short time interval would trigger multiple instances of the query to run simultaneously. This results in high energy consumption as every instance of a query performs its own sensor sampling and transmission of results. To alleviate this problem TinyDB optimizes such queries by rewriting them so that all occurrences of event E of the last k seconds are buffered. When a sensor reading is obtained, it is compared against the buffered events and the temperature readings are returned. Thus, no matter how frequently events of type E are triggered, only one query is required to run.

Deshpande *et al.* [10] extend the initial acquisitional query processing concepts introduced in TinyDB by utilizing certain statistical modelling techniques. The Barbie-Q (BBQ) query system creates models that provide answers that are more meaningful, and also help to extend the lifetime of the network by returning approximated results coupled with probabilistic confidences. There are three major steps involved in query processing in the BBQ architecture:

Building the model. This involves the collection of raw data from every node. This historical data is used to build a model based on time-varying multivariate Gaussians and allows one to observe the correlations and statistical relationships between sensor readings on various nodes.

Generating an observation plan. Users inject queries that are similar to SQL except for the fact that they are allowed to include error tolerances and target confidence bounds that specify the degree of uncertainty a user is willing to accept. Using the probabilistic model mentioned earlier, coupled with the accuracy specifications of the injected query, the system takes the liberty of determining the most efficient way of servicing the query and generates the corresponding *observation plan*. Depending on the specified error tolerance, the observation plan may only poll a small proportion of all the sensors that are actually capable of servicing the query. It may even retransform a query by querying a different sensor from the one specified in the original injected query, if it is more energy efficient and both sensor readings are found to be correlated.

Updating the model. BBQ relies on Markovian models to keep the model updated regarding changing environmental parameters and to ascertain any temporal correlations. This helps to generate a probability density function (pdf) of all parameters at time $t+1$ given the pdf at time t.

DOI: 10.1002/dac

Thus, unlike TinyDB and COUGAR, BBQ does not interrogate all sensors every time a query is injected into the network. Instead, sensors are used to acquire data only when the model itself is not sufficiently rich to answer the query with acceptable confidence. As time passes, the model may update its approximations of sensor readings to reflect expected temporal changes in the data.

As BBQ builds a statistical model centrally, it is unable to detect sudden changes that may occur within the network, e.g. spikes in temperature readings or changes in network topology. Ken [11], which is similar in certain respects to BBQ, tackles the problem of detecting outliers, by storing models within the sensor nodes instead of storing them centrally. Every node also transmits its model to the central server. When a node acquires a reading, it checks the acquired reading against the locally stored model. If the acquired reading falls outside the reading predicted by the model by a margin that is larger than the user-specified threshold, the node re-computes a new model and transmits this new model together with the newly acquired reading. Ken too however, is unable to adapt to changes in the network topology. It should be noted at this juncture that Ken however, does not actually fall in the class of acquisitional query processing. Instead, it should fall in the super class, in-network processing, as Ken does not actually deal with sensor sampling. Instead, it aims to minimize communication by making use of dynamic probabilistic models. We have however, mentioned it here as its operation is very closely linked to BBQ.

3.3. Cross-layer optimization

Traditional system architectures are commonly known to adhere to the layered protocol stack design where every layer operates in a completely independent manner. The Internet browser on a PC, for example, does not require any knowledge about the kind of network connectivity available. It works regardless of whether the user is connected via Ethernet or 802.11b. Such a layered approach however, is not optimal from the energy efficiency point of view especially when considering WSNs. This is because unlike the network being used in an office LAN, WSNs are typically *application-specific* networks. Thus, it only makes sense to try and make the various components of the WSN architecture more *application-aware* by performing certain cross-layer optimizations thereby helping to improve network lifetime. Note however, that while the term 'cross-layer optimization' in the field of sensor networks might refer to the optimization between the application and routing layers for instance, in this case we refer to the more radical approach where optimization is performed between the application and MAC layer.

As usage of the transceiver is an energy-consuming task, it is imperative that maximum benefit is derived during the time it is operational. Thus, rather than encountering energy-wasting collisions during data transmission or actively waiting for messages that do not arrive, current cross-layer optimization techniques use a variety of methods to try and schedule tasks in an energy-efficient manner. We now review a number of these techniques.

TinyDB [3] uses an interval-based communication scheduling protocol to collect data where parent and child nodes receive and send data (respectively) in the tree-based communication protocol. The cross-layer optimization in TinyDB involves (i) reducing the burden on the MAC using specifications from the injected query and (ii) routing data from the source nodes to the root. Each node is assumed to produce exactly one result per epoch (time between consecutive samples), which must be forwarded all the way to the base station. As shown in Figure 4, every epoch is divided into a number of fixed-length intervals which is dependent on the depth of the

DOI: 10.1002/dac

tree. The intervals are numbered in reverse order such that interval 1 is the last interval in the epoch. Every node in the network is assigned to a specific interval, which correlates to its depth in the routing tree. Thus, for instance if a particular node is two hops away from the root node, it is assigned the second interval. During its own interval, a node performs the necessary computation, transmits its result and goes back to sleep. In the interval preceding its own, a node sets its radio to 'listen' mode collecting results from its child nodes. Thus, data flows up the tree in a staggered manner eventually reaching the root node during interval 1.

While TinyDB's slotted scheduling protocol does help conserve energy by keeping nodes asleep a significant proportion of time, it is primarily designed for servicing a single query posed to the entire network. The scheme is unusable if there are multiple concurrent queries with different epoch requirements.

WaveScheduling [6] is a scheme that helps minimize collisions at the MAC layer by carefully scheduling nodes. While the scheme results in energy savings, the drawback is that latency is increased. The solution is however, a bit 'rigid' as the nodes need to be arranged in a grid in order to operate properly. This may not be feasible in all applications.

Cetintemel *et al.* [12] describe *Topology-Divided Dynamic Event Scheduling* (TD-DES), which is an event-based communication model for WSNs that allows a node to switch its radio to a low-power standby mode when it is not interested in certain queries/events that have been disseminated into the network. However, rather than being a MAC layer itself, TD-DES is intended as an application overlay to a CSMA/CA wireless MAC layer. TD-DES sets up a wireless multi-hop network tree where the root node generates a *data dissemination schedule* describing when nodes should switch on their transceivers to capture certain events. This schedule is disseminated throughout the entire network so that nodes can follow it. The schedule is divided into fixed-sized time slots. Each slot is further divided into three sections, a *control event receive slot*, a *control event send slot* and an *event data slot*.

Every node in the network has a specific subscription profile that describes the list of measured parameters or events that the node is interested in. The node may either be a subscriber, i.e. it is interested in a particular event or a generator of certain events. The *control event send slot* holds scheduling information of events and is transmitted to neighbouring nodes. A node listens out for scheduling information from a neighbouring node during the *control event receive slot* and the *event data slot* is used to transfer data regarding the actual event. A node listens to the relevant portion of the *event data slot* in receive mode only if it discovers an event that is of interest to itself or one of the nodes in its sub-tree listed in the scheduling information received from a neighbouring node in the *control event receive slot*.

TD-DES differs from TinyDB's scheduling scheme in the sense that TD-DES addresses both upstream and downstream data dissemination while TinyDB focuses solely on upstream aggregation. Also, while TinyDB assumes that every node in the network takes part in servicing the injected query, in TD-DES, nodes not subscribing to an event do not own any scheduled slot.

Zadorozhny *et al.* [13] describe a framework that helps to extend the synergy between the MAC layer and query optimization. This is achieved with the help of a *data transmission algebra* (DTA) that provides the semantics for defining rules that control the order in which nodes transmit data. For example, if there is a node that needs to perform Task A and another node that needs to perform Task B and both tasks require the use of the RF medium, the DTA is used to work out a schedule such that both tasks are performed with minimal chance of collisions

DOI: 10.1002/dac

from occurring. The DTA includes three basic operations that combine the two transmissions schedules of tasks A and B as follows:

- Either Task A is scheduled to perform before Task B or Task B before Task A
- Task A *MUST* be performed before Task B
- Task A and Task B can be performed simultaneously (e.g. when nodes involved are not within transmission range of one another)

The primary aim of this framework is for the root node to work out schedules for all nodes in the network using the DTA scheduler. It is then up to the nodes to decide how to behave within a set of constraint intervals specified by the schedule. As the number of possible schedules grows exponentially with the number of sensor nodes, heuristic-based pruning methods are used to eliminate suboptimal alternatives.

The above-mentioned scheduling techniques clearly indicate that current cross-layer optimization schemes stop just short of actually altering the operation of the MAC protocol itself. Instead, the schemes simply employ different strategies to turn the MAC on and off at different times therefore decreasing the burden placed on the MAC. If collisions still do occur, it is the responsibility of the MAC to recover from them.

Chatterjea *et al.* [14] however, describe an *adaptive, information-centric and lightweight MAC protocol for wireless sensor network* (AI-LMAC) that adapts its operation based on the requirements of the application. The amount of bandwidth that is allocated to a node is proportional to the amount of data that is expected to flow through it in response to the query it is servicing. Bandwidth allocation is done in a distributed manner and is not static but changes depending on the injected query. Information about the expected data traffic through a node is obtained using a completely localized data management framework that helps capture information about traffic patterns in the network. A major advantage of this approach is that the MAC protocol reduces its duty cycle on nodes that are *not* taking part in servicing the query, thus improving energy-efficiency and limiting communication activity only to areas of the network where it is actually required. The data management framework is also used for efficient query dissemination (i.e. directing queries to only the relevant parts of the network) and query optimization. Thus cross-layer optimization in AI-LMAC addresses the entire spectrum of data management issues, i.e. operation of the MAC, routing, and query optimization.

3.4. Data-centric data/query dissemination

Deciding how to route or disseminate data within a communication network is typically performed using IP-style communication where nodes are identified by their end-points, which can be directly addressed using an address that is unique to the entire network. Such addressing schemes are used in the Internet or office LAN. Due to their application-specific nature however, WSNs tend to use a *data-centric* addressing scheme where 'names' are used to create an abstraction of node network addresses. For example, instead of requesting the reading of a node with a particular ID, a typical query usually requests for an attribute of a phenomenon instead. Data dissemination schemes generally address three main issues:

- How queries are routed only to the relevant nodes from the node injecting the query into the network (flooding a query to all nodes is not energy-efficient)
- How results are routed back to the querying node

DOI: 10.1002/dac

- Robustness: how the scheme copes with dynamic network topologies, i.e. node failures and node mobility

We review a number of data dissemination schemes which use distributed techniques to ensure that queries only propagate towards sensors capable of serving the injected queries.

Directed diffusion [1] is one of the pioneering data-centric data dissemination paradigms developed specifically for WSNs. It is based on a publish/subscribe API where the sink injects an interest (e.g. interest for a particular type of *named data*, such as the 'Detection of a bird') into the network. Every receiving node keeps a copy of the interest in its cache. The entry in the cache also stores a *gradient* that indicates the identity of the neighbouring node that originally sent the interest. Gradients are formed gradually at every node as the interest propagates from one node to another eventually flooding the entire network (Figure 5). When a source node is able to service the interest, it sends data back to the sink along the path of the gradients set up initially by the interest. In the event data starts flowing toward the sink along multiple gradient paths, the source node reinforces one or a subset of these paths. Reliable paths are usually reinforced while unreliable ones are removed by expiration due to lack of reinforcements or explicit negative reinforcements. Such gradients allow the *local repair* of failed or degraded paths and do not require the re-flooding of the interest. However, it is necessary to perform flooding when a new interest is injected into the network.

While directed diffusion performs routing based on named data, TinyDB performs routing using a *semantic routing tree* (SRT), which is based on the actual values of sensor readings. It is

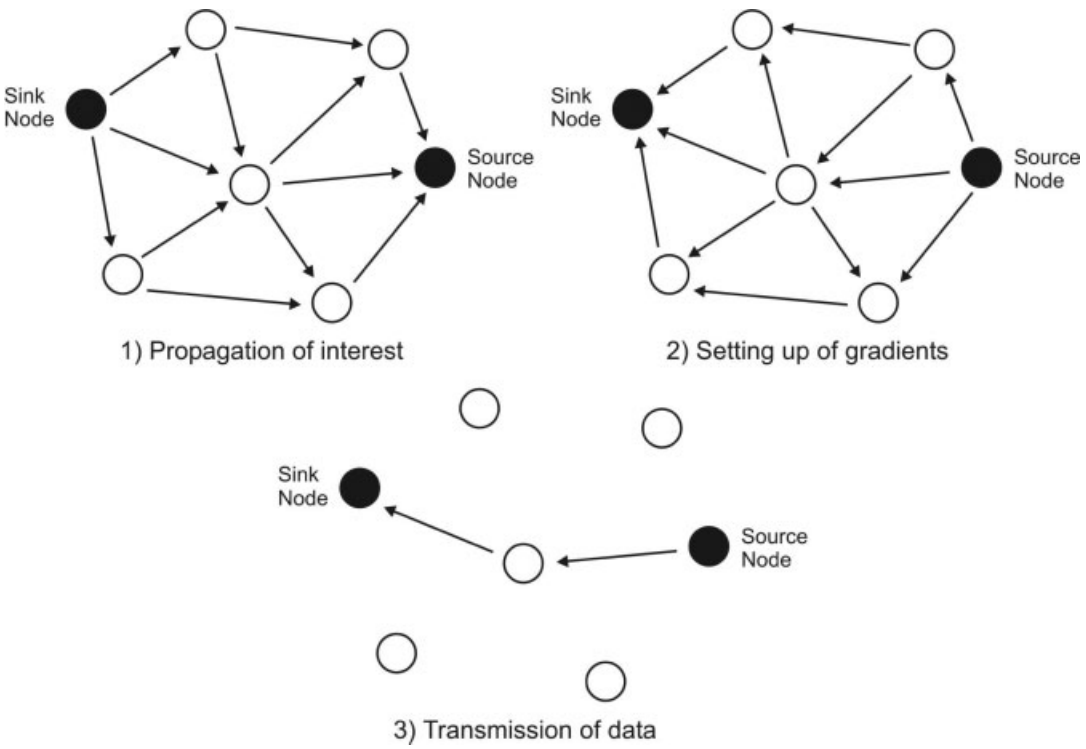

Figure 5. Seting up of gradients in directed diffusion.

DOI: 10.1002/dac

useful for servicing *range queries*. An SRT is an index built over some constant attribute A and is stored locally at every node in the network. The index at a node consists of a one-dimensional interval representing the range of A values being generated not just by the node itself, but also by all its descendants. When a node encounters a query, it only forwards it to its immediate children which are reported to be transmitting values matching the required range specified in the query. The readings may have been generated either by any of the immediate children or by any of the nodes within the sub-trees rooted at the immediate children. Additionally, a node executes a query by itself if it can be serviced locally and subsequently transmits the result to its parent. The result eventually propagates up the tree towards the root. If the query cannot be serviced by the node or any of its children, it is dropped. The entry of a child node expires from the SRT of a parent node if the parent node does not receive any updates from the child within a predefined timeout period. The parent then updates its interval information by querying its children, and also informs nodes higher up the hierarchy if any changes are detected. While the SRT-maintenance algorithm is capable of reflecting changes in the network dynamics (e.g. death of a node), the cost of updating ranges could be prohibitive if the value of the measured attribute changes too rapidly.

Coman *et al*. [8] present a framework for processing queries that specify the spatial area and the temporal range the answers must belong to. The framework has two phases. In the first phase, a path is searched from the query originator to a sensor node located within the query's spatial window. Next, the located sensor assumes the role of query co-ordinator and gathers results from all the relevant sensors from within the spatial window and transmits the results back to the query originator. Note that it is assumed that all the nodes are location aware.

Unlike directed diffusion and TinyDB's SRT, Ratnasamy *et al*. [15] deal not just with the data-centric routing aspects but also integrate it with the issue of storage within the network. They propose using *data-centric storage* (DCS) where all events are named. A vital assumption is that all nodes are location aware. There are two main operations within DCS — storing and retrieving data. When a node detects a particular event, it stores the data by name in a node within the network. The *geographic hash table* (GHT) scheme performs a hashing on the name of the data into geographic co-ordinates thus deciding in which part of the network data should be stored. GHT is also used in a similar manner when retrieving data. In order to address the issue of robustness, GHT uses the *perimeter refresh protocol*. This protocol replicates the event data at nodes around the location to which the hashing was originally made thus ensuring that queries can still be serviced even if certain nodes fail. The occurrence of multiple identical events (i.e. events with the same name) would all hash to the same location thus creating a bottleneck in the network for both store and retrieve operations. In order to alleviate this problem, GHT employs *structured replication* such that events hashing to the same node are mirrored in different parts of the network.

GHT is effective in saving unnecessary transmissions by preventing the need to flood the entire network with queries by performing hashing on an attribute. However, DIFS [16] achieves even greater savings by also including support for *range queries*, i.e. queries where only events with attributes in a specific range are required. This is because events defined by attributes with values that fall within a specified range are by definition less common. For example, while there may be many humidity sensors in a network, there may only be a small fraction, which have readings higher than a certain threshold. DIFS constructs a multi-rooted hierarchical index where non-root nodes can have multiple parents. Thus, if a child node has p number of parents and maintains a range of values r, each index of a parent node maintains an

DOI: 10.1002/dac

equal proportion of r, i.e. r/p. However, the narrower the value range covered by a node, the wider the spatial extent an index node knows about. In other words, higher-level nodes cover smaller value ranges detected within large geographic regions while lower-level nodes cover a wider range of values from within a smaller geographic region. DIFS also reduces the possibility of bottlenecks occurring close to the root node as queries can be injected into any node in the tree. Unlike GHT which can hash to any part of the network, the DIFS hash function restricts its output to the area that a node in the hierarchy has to cover. The function outputs a location upon receiving the following inputs: event name, event value and event location. DIFS however, does not deal adequately with the issue of node failures and packet losses.

The *distributed index for multi-dimensional data* (DIM) [17] goes a step further by building on top of the above-mentioned methodologies by including support for queries that specify multiple range conditions. For example, a query might require all events to be reported that fall within a particular temperature range *and* a particular humidity range. Such queries can be particularly useful for correlating multiple events and subsequently triggering certain actions. DIM runs a distributed algorithm on every node that eventually divides the sensor field such that there is a single node in each zone. Next, using the values of multiple attributes, the hashing function hashes the event to a zone. The event is subsequently routed to the node that *owns* the zone and is stored there. DIMs include the following features to address the issue of robustness:

- use of a mechanism to help re-assign zones when nodes join or leave the network
- use of a distributed algorithm to minimize the chance of data loss by carrying out replication of data
- an ACK scheme to improve resilience to packet loss

Query and data dissemination schemes that prevent the need to flood the entire network have progressed markedly in the recent past. From initially just considering event types or ranges of actual sensor readings, they currently support multiple range queries and use various hashing functions to direct queries to the appropriate sections of the network using the attribute types and values as inputs. Furthermore, they incorporate in-network storage as an integral part of the query dissemination mechanism. However, these newer schemes assume that all nodes are location aware thus increasing the complexity of the system.

4. CONCLUSION

As time progresses, WSN deployments are gradually going to grow larger and certain deployments may even be enlarged in stages. This makes it increasingly necessary to improve support for heterogeneous networks, multiple roots and optimization of multiple simultaneous queries that may overlap partially over sensor types, readings, and spatial and temporal parameters. Cross-layer optimizations from the application layer also need to dig in deeper into the network layers and attempt to eventually influence the operation of the MAC. Query optimizations need to be pushed into the network to prevent large amounts of metadata being sent back to the root. An interesting observation from the summary presented in Table I shows that only a handful of the projects have actually been implemented on sensor node platforms. This clearly reflects that while certain projects have begun making inroads into the various essential conceptual features mentioned in this paper, current distributed data management techniques only touch the tip of the iceberg.

DOI: 10.1002/dac

REFERENCES

1. Intanagonwiwat C, Govindan R, Estrin D, Heidemann J, Silva F. Directed diffusion for wireless sensor networking. *ACM/IEEE Transactions on Networking* 2002; **11**(1):2–16.
2. Yao Y, Gehrke J. Query processing for sensor networks. *Proceedings of the 1st Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, U.S.A., January 2003.
3. Madden SR, Franklin MJ, Hellerstein JM, Hong W. TinyDB: an acquisitional query processing system for sensor networks. *ACM Transaction on Database Systems* 2005; **30**(1):122–173.
4. Beaver J, Sharaf MA, Labrinidis A, Chrysanthis PK. Power-aware in-network query processing for sensor data. *In proceedings of second hellenic data Management Symposium (HDMS)*, Athens, Greece, September 2003.
5. Deligiannakis A, Kotidis Y, Roussopoulos N. Compressing historical information in sensor networks. *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, Paris, France, 2004.
6. Trigoni N, Yao Y, Demers A, Gehrke J, Rajaraman R. WaveScheduling: energy-efficient data dissemination for sensor networks. *Proceedings of the 1st International Workshop on Data Management for Sensor Networks*, Toronto, Canada, 2004.
7. Bonfils, BJ, Bonnet P. Adaptive and decentralized operator placement for in-network query processing. *Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, U.S.A., 2003.
8. Coman A, Sander J, Nascimento MA. An analysis of spatio-temporal query processing in sensor networks. *Proceedings of the 1st IEEE International Workshop on Networking Meets Databases*, Tokyo, Japan, 2005.
9. Abadi D, Madden S, Lindner W. REED: robust, efficient filtering and event detection in sensor networks. *Proceedings of VLDB*, Trondheim, Norway, 2005.
10. Deshpande A, Guestrin C, Madden S, Hellerstein JM, Hong W. Model-driven acquisition in sensor networks. *Proceedings of VLDB*, Toronto, Canada, 2004.
11. Chu D, Deshpande A, Hellerstein JM, Hong W. Approximate data collection in sensor networks using probabilistic Models. *Proceedings of ICDE*, Atlanta, GA, U.S.A., 2006.
12. Cetintemel U, Flinders A, Sun Y. Power-efficient data dissemination in wireless sensor networks. *Proceedings of the 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access*, San Diego, CA, U.S,A., 2003.
13. Zadorozhny VI, Chrysanthis PK, Krishnamurthy P. A framework for extending the synergy between MAC layer and query optimization in sensor networks. *Proceedings of the 1st International Workshop on Data Management for Sensor Networks*, Toronto, Canada, 2004.
14. Chatterjea S, van Hoesel LFW, Havinga PJM. AI-LMAC: an adaptive, information-centric and lightweight MAC protocol for wireless sensor networks. *Proceedings of the 1st international Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Melbourne, Australia, 2004.
15. Ratnasamy S, Karp B, Shenker S, Estrin D, Govindan R, Yin L, Yu F. Data-centric storage in sensornets with GHT, a geographic hash table. *Mobile Networks and Applications (MONET)* 2003; **8**(4):427–442.
16. Greenstein B, Ratnasamy S, Shenker S, Govindan R, Estrin D. DIFS: a distributed index for features in sensor networks. *Ad Hoc Networks* 2003; **1**(2–3):333–349.
17. Li X, Kim YJ, Govindan R, Hong W. Multi-dimensional range queries in sensor networks. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, U.S.A., 2003.
18. The COUGAR Sensor Database Project, http://www.cs.cornell.edu/boom/2002sp/extproj/www.cs.cornell.edu/database/cougar/default.htm, 2002.
19. Hellerstein J, Hong W, Madden S, Stanek K. Beyond average: Towards sophisticated sensing with queries. *Proceedings of the First Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, U.S.A., 2003.

AUTHORS' BIOGRAPHIES

Supriyo Chatterjea is currently a full-time PhD student at the Department of Mathematics, Electrical Engineering and Computer Science at the University of Twente. After graduating from Nanyang Technological University, Singapore, with a Bachelor's degree in Electrical Engineering in 2001, he obtained a Master's degree with distinction in Computing and Internet Systems from King's College London, U.K., in 2002. He was also awarded the 'Best MSc Project' award for his work on service discovery in mobile *ad hoc* networks at IBM Research Laboratory, Zurich, Switzerland. His research interests lie in the field of distributed data management for wireless sensor networks.

DOI: 10.1002/dac

Dr Paul J.M. Havinga is an associate professor in the Computer Science department at the University of Twente in the Netherlands. He received his PhD on the thesis entitled 'Mobile Multimedia Systems' in 2000, and was awarded with the 'DOW Dissertation Energy Award' for this work. His research interests are in the area of large-scale, heterogeneous wireless systems, sensor networks, energy-efficient architectures and protocols, ubiquitous computing, personal communication systems, and wireless communication networks. This research has resulted in over 180 scientific publications in journals and conferences.

He is the project manager of the Bsik project Smart Surroundings, on ambient intelligence, the European project EYES, on energy-efficient sensor networks, the Dutch projects Featherlight on distributed system software, and CONSENSUS, on application support for collaborative sensor networks. He is the workpackage leader in the EU IST projects CoBis, e-Sense, Aware, and co-ordinates research exchange in Embedded WiSeNts on sensor networks and applications.

He is the editor of several journals and magazines. He is involved as program committee chair, member, and reviewer for many conferences and workshops. He has been a visiting researcher at the University of Pisa in 1998, and the Communications Research Laboratory in Yokosuka Japan in 2000.

He regularly serves as an independent expert for reviewing and evaluation of international research projects for the EU, the U.S., and international governments.

DOI: 10.1002/dac