

# A Framework for a Distributed and Adaptive Query Processing Engine for Wireless Sensor Networks

S. Chatterjea and P. Havinga  
Department of Computer Science, University of Twente  
P.O. Box 217, 7500AE Enschede, the Netherlands  
Tel +31-53-489-3770, Fax +31-53-489-4590, E-mail {supriyo, havinga}@cs.utwente.nl

## ABSTRACT

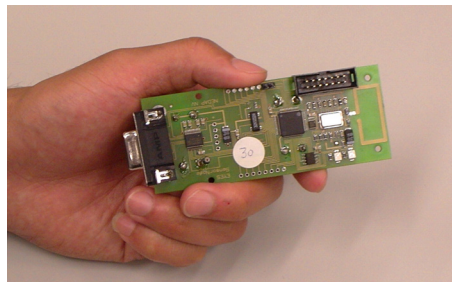
Wireless sensor networks (WSNs) are formed of tiny, highly energy-constrained sensor nodes that are equipped with wireless transceivers and can be used primarily in environmental monitoring applications. The nodes communicate with one another by autonomously creating ad-hoc multihop networks which are subsequently used to gather sensor data. WSNs also process the data within the network itself and only forward the result to the requesting node. This is referred to as in-network data aggregation and results in the substantial reduction of the amount of data that needs to be transmitted by any single node in the network. We present a framework for WSNs which would allow optimised query execution plans to be generated in a distributed manner within the network thus preventing the need to transmit network metadata all the way to a central server. Query plans also adapt continuously by monitoring varying network conditions to maintain energy-efficient operation thus maximising network lifetime.

*Keywords: wireless sensor networks, distributed, adaptive, query processing engine*

## INTRODUCTION

Wireless sensor networks (WSNs) are formed of tiny, extremely low-powered sensor nodes that are equipped with built-in wireless transceivers. Figure 1 shows a picture of a development board of an EYES [3] sensor node together with some of its specifications. It should be noted that the transmission costs are dependent on the type of radio and processing platform used. The EYES node is currently used for testing purposes and the final version of it will be around the size of a one Euro coin. These nodes may be deployed in large numbers and are capable of communicating with one another by autonomously creating multihop ad-hoc networks which are subsequently used to gather sensor data. Considering the fact that these battery-powered nodes are supposed to operate for months (possibly even years) and that it is assumed that battery replacement is not a viable option due to the large numbers involved, one of the primary concerns of wireless sensor networks is how to extend the longevity of the network to the furthest possible extent by devising novel ways of managing energy reserves.

Our framework describes a distributed and adaptive query processing engine for wireless sensor networks that addresses two main problem areas. In existing systems [1, 6] network metadata is relayed back to a central node by all the nodes in a sensor network at



Specifications: 16-bit, 5MHz processor, 60kB ROM, 2kB RAM, 2MB EEPROM, 115.2kbps data transfer rate, Power consumption: 16mW (Transmit), 14.4mW (Receive), 0.015mW (Standby)

Figure 1. A development board of an EYES sensor

regular intervals. This information is used to obtain a global view of the network which in turn helps to generate optimised query execution plans. The central node subsequently sends out instructions to specific nodes to perform certain tasks, e.g. data aggregation, routing, etc. Moreover, to generate query execution plans that are up-to-date with the current network dynamics, it is essential that the central node always has a fresh view of the network. This mechanism has two fundamental problems. Firstly, the transmission of all network metadata by every node in the network to a

central node at regular intervals is an extremely energy consuming task and secondly, in large networks, the collection of network metadata is a particularly high-latency process. Thus the generated query plan may be rendered useless due to backdated network metadata. In view of these problems, we suggest a framework where the generation of query execution plans is performed within the network based on locally available knowledge thus cutting back drastically on expensive transmissions to a central node. Consequently a node evaluates incoming queries and generates query execution plans based on the network metadata relayed by its neighbouring nodes.

We currently assume that queries will be similar to the SQL-style queries used in TinyDB [6]. The example below retrieves the average temperature of every floor in a building for a period of 60 minutes at 30 second intervals.

```
SELECT AVG(temp), floor
FROM sensors GROUP BY floor
SAMPLE PERIOD 30s for 3600s
```

However, apart from including support for long-running periodical queries, we also intend to introduce methods to service event-based queries.

Instead of “static” query execution plans where the central node instructs a certain node in the network to perform a specific task, an additional feature of our framework is that it allows query plans executing within the network to adapt to varying network conditions. This is done by nodes calculating possible execution costs on a regular basis. If for instance, a node feels that it can perform an aggregation task more efficiently than its neighbour (by comparing execution costs), the task is transferred to it. This process ensures that all nodes consume their energy at around the same rate thus prolonging network lifetime.

In this paper we first give a brief overview of the related work in this area. We then present a framework for a distributed query processing engine for wireless sensor networks that will allow wireless sensor nodes to autonomously gather and analyse data on-site in an energy-efficient manner. We also describe how the framework adapts automatically to varying network conditions. We finally conclude the paper by stating the work that needs to be done in the future to build up on the framework presented.

Our work on WSNs is performed as part of the NWO funded CONSENSUS project [2] and the European

EYES project (IST-2001-34734) [3] on self-organising and collaborative energy-efficient sensor networks. It addresses the convergence of distributed information processing, wireless communication and mobile computing.

## RELATED WORK

The most energy consuming operation a node can perform is the transmission of data. In fact, transmitting just 1Kb of data a distance of 100 metres is approximately equal to the cost of executing three million CPU instructions [7]. Apart from simply collecting data, WSNs are designed to process data within the network itself and subsequently forward the result to the requesting node. This is referred to as in-network data aggregation and results in the substantial reduction in the amount of data that needs to be transmitted within the network as a whole which in turn translates into substantial energy saving. Data aggregation can be performed by intermediate nodes that lie between the sink (a node that injects a query into a network) and source (a node that responds to a query by sensing some physical parameter) node. These intermediate nodes carry out partial computation of the data obtained from sensor nodes thus ensuring that each node only has to transmit one data message. This also implies that bandwidth requirements between neighbouring nodes remain constant regardless of their position in the tree.

While performing data aggregation within the network may result in extending the operational lifetime of sensor nodes, it is important to note that there may be numerous ways to evaluate any particular query. Out of all these possibilities, only a handful might actually lead to energy savings. Thus it is important to develop a system that can analyse every incoming query and work out the optimal solution using the current network dynamics to ensure accurate decision making.

In certain existing query processing systems such as COUGAR [1] and TinyDB [6], network statistics (or network metadata) such as patterns of data produced by individual nodes, location information and energy reserves of nodes, etc. are sent back periodically to the central node (server) which originally injected a query into the network.

Using the centrally collected data, the server, which now has a detailed overview of the status of the entire network of nodes, calculates the optimal method in which the query may be evaluated. So the server

generates a set of instructions that are then sent out to the individual nodes explaining the role individual nodes will play in evaluating the query, e.g. the server may stipulate which specific nodes would be required to perform aggregation of data for a particular query.

Another well known data-centric aggregation framework is Directed Diffusion [4] that is based on a publish/subscribe API. This however, does not use a database approach towards managing sensor data. Instead it uses a lower-level approach and does not hide the networking specifics from the user thus making deployment of a system more difficult. DFuse [5] has a dynamic approach of placing fusion points but once again fails to provide the user with a SQL-like interface that is highly user-friendly. Also, DFuse is unable to handle data-centric queries and assumes that the specific addresses of nodes are known at the time of a query.

## DESIGN OF THE QUERY PROCESSING ENGINE

We suggest a new framework for a *completely* distributed and adaptive query processing engine (QPE) for wireless sensor networks. The primary difference from the existing models is that in our framework, we transfer the task of generating query plans from a central node to the sensor nodes that lie within the network. Since nodes generate query plans using information that is available locally they greatly reduce the number of transmissions to the server. Every node would be able to detect changes in its vicinity and make necessary changes to its query execution methods almost immediately. Also, the query evaluation procedure is carried out autonomously by the nodes and is completely transparent to the user. The user need not be concerned about the current network dynamics.

In this section, we highlight two of key features of our framework. First we describe its architecture which primarily deals with the individual building blocks of the QPE that help in the efficient processing of an incoming query. The second feature ensures that the framework continuously adapts to the varying network dynamics thus enabling the aggregation of data to be carried out by using the least amount of resources. Both these features would help to extend the network lifetime.

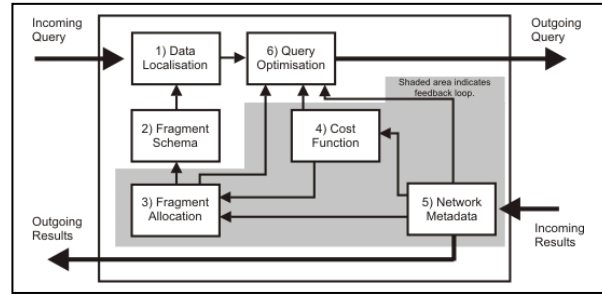


Figure 2. Framework of the distributed query processing engine

### Architecture of Framework

The QPE has a hybrid design involving a combination of centralised and distributed architectures. It consists of two sections – one residing on the main server (i.e. the central node) and the other residing on every node within the network. The section residing on the server carries out optimisations based solely on incoming queries. It does not require any information about the distribution of data within the network.

The second section of the QPE lies on top of the operating system in every node within the network. It is structured as shown in Figure 2. The six components shown in Figure 2 can be separated into two sections based on their functionality. The following subsections describe the role of each block in Figure 2.

- *Query decomposition*

This is the only block that is located on the central node. Queries injected into a network may be in binary SQL format which may be modified to support more complex data aggregation functions, depending on the requirements of the application. Query semantics are checked and redundancies are eliminated. A list of operator trees is generated describing various ways of executing the same query. Transformation rules are used to restructure operator trees in a systematic way so that “bad” operator trees are eliminated at the very first stage itself. The remaining operator trees are far from optimised though as no information about the distribution of data is used at this stage.

- *Data localisation*

The main role of the data localisation block is to localise the query’s data using data distribution information which is obtained from the Fragment Schema block. Thus data localisation examines the incoming query from the query decomposition block and determines which fragments of data are involved in

the query. It also retransforms incoming queries into simpler and more optimised forms by using different reduction techniques depending on how the data has been fragmented. For example, when selections on fragments are made that have a qualification contradicting the qualification of a fragmentation rule, empty relations (or redundancies) are generated. Reduction rules ensure that such empty relations are eliminated.

Consider a relation *sensors* containing readings of light and temperature of different floors in a building which is split into three horizontal fragments *sensors<sub>1</sub>*, *sensors<sub>2</sub>*, *sensors<sub>3</sub>* and is defined as follows:

$$\begin{aligned} sensors_1 &= \sigma_{Floor \leq "4"}(sensors) \\ sensors_2 &= \sigma_{"4" < Floor \leq "8"}(sensors) \\ sensors_3 &= \sigma_{"8" < Floor \leq "12"}(sensors) \end{aligned} \quad (1)$$

Thus the localisation program for such a horizontally fragmented relation would be as follows:

$$sensors = sensors_1 \cup sensors_2 \cup sensors_3 \quad (2)$$

Such horizontal fragmentation can help to simplify certain operations as it is possible to produce empty relations and subsequently eliminate them. This can be illustrated in the case of a selection operation as shown in the following SQL statement:

```
SELECT nodeid, temp, light
FROM sensors
WHERE Floor = "7"
SAMPLE PERIOD 1s for 60s
```

The statement above gathers temperature and light readings from the 7th floor every second for a whole minute. The naïve way to process this query would be to replace every instance of sensors with (*sensors<sub>1</sub>*  $\cup$  *sensors<sub>2</sub>*  $\cup$  *sensors<sub>3</sub>*). It is evident that *sensors<sub>1</sub>* and *sensors<sub>3</sub>* would then produce empty relations which are undesirable. Thus the reduced query is only applied to *sensors<sub>2</sub>*. Such query simplifications can subsequently be used to limit the broadcast of a query to a particular section of the network thus resulting in energy consumption due to the reduction of transmissions.

- *Fragment schema*

The Fragment Schema describes how it is possible to reconstruct a relationship from the various fragments. It extracts the information from the Fragment Allocation block. In other words it may describe how to apply a set

of UNION operations to recreate a relation from the fragments.

- *Fragment allocation*

The task of the Fragment Allocation block is to decide at which node a certain fragment of a relation should be stored. Suppose there are a set of fragments  $F = \{F_1, F_2, \dots, F_N\}$  and a cluster of sensor nodes,  $S = \{S_1, S_2, \dots, S_N\}$ , the Fragment Allocator needs to find the optimal distribution of  $F$  to  $S$ . There are numerous parameters that need to be considered during the optimisation process, e.g. cost of communication between any two nodes,  $S_i$  and  $S_j$ , varying access patterns of various nodes, mobility patterns, cost of storing each  $F_i$  at a site  $S_j$ , cost of querying  $F_i$  at a site  $S_j$ , remaining energy reserves of a node and memory and performance parameters such as throughput and response time.

- *Cost function*

The Cost Function block stores the cost of some static parameters such as cost of polling a sensor or radio characteristics and are generally defined prior to deployment of the sensor network.

- *Network metadata*

This block gathers network statistics based on every single query or result that the node hears, i.e. the query or result may not be addressed to it specifically but it might overhear a certain message from a neighbouring node that is within its transmission range, e.g. sizes of relations, patterns of query flow, energy and memory reserves of a particular node

- *Query optimisation*

The Query Optimisation block receives several execution strategies (or operator trees) for a single query from the Data Localisation block. It is within this block that the QPE actually takes into account the distribution of data fragments, various costs involved and the current network dynamics in order to generate a query execution plan.

## Adapting to Network Dynamics

We introduce an *adaptation mechanism* that allows aggregation points to shift such that it is performed only at nodes which spend the least amount of resources. We refer to the “amount of resources” as the cost calculated by the Query Optimisation block for the respective query optimisation plan. The general idea is that the neighbours of say a node  $A$  performing a certain aggregation task monitor the cost incurred by



operation.

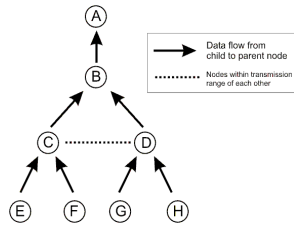


Figure 5. Downward shift

From Figure 5, if node B lacks memory resources, it indicates this when sending data to its parent, node A. Node C overhears the transmission and realises that it can perform the task better as it has ample memory. It thus requests node B for a task transfer. Assuming that nodes C and D are within transmission range, and the task being performed at node B requires data from C and D, node C may request D to become its child node. Thus the routing tree may be changed in this case.

## CONCLUSION AND FUTURE WORK

In this paper, we have described a framework for a distributed and adaptive query processing engine for wireless sensor networks where we concentrate on two primary problems – (i) reducing transmissions by generating query optimisation plans based on local information, (ii) introducing an adaptive mechanism that ensures that a query is always executed in the most efficient manner even with varying network dynamics. Now that we have laid out the basic framework of our aggregation framework, the next step is to implement it in OMNET++ and perform a quantitative analysis of the system. We shall also be looking into the memory requirements of the framework.

## REFERENCES

- [1] P. Bonnet, J. Gehrke and P. Seshadri. Towards sensor database systems. In *2nd International Conference on Mobile Data Management*, Hong Kong, January 2001
- [2] Consensus homepage: <http://www.consensus.tudelft.nl/>
- [3] EYES homepage: <http://eyes.eu.org>
- [4] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the ACM/IEEE International Conference on Mobile Computing and Networking*, pages 56-67, Boston, MA, USA, Aug. 2000. ACM.

- [5] R. Kumar, M. Wolenetz, B. Agarwalla, J. Shin, P. Hutto, A. Paul and U. Ramachandran. DFuse: A Framework for Distributed Data Fusion. In *1<sup>st</sup> International Conference on Embedded Networked Sensor Systems*, Los Angeles, CA, USA, Oct. 2003. ACM.
- [6] S. Madden, R. Szewczyk, M. J. Franklin and D. Culler. Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks. In *4th IEEE Workshop on Mobile Computing Systems and Applications*, Callicon, NY, June 2002.
- [7] G. Pottie and W. Kaiser, Wireless integrated network sensors. In *Communications of the ACM*, 2000.