Algebraic Optimization of Data Delivery Patterns in Mobile Sensor Networks

Vladimir Zadorozhny
Dept. of Info Science and Telecom
University of Pittsburgh
Pittsburgh, PA 15260
vladimir@sis.pitt.edu

Panos K. Chrysanthis
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
panos@cs.pitt.edu

Alexandros Labrinidis
Dept. of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
labrinid@cs.pitt.edu

Abstract

Database-like query processing over a network of sensors has become an attractive paradigm for building sensor applications. A sensor query is characterized by data streams among participating sensor nodes with possible in-node data filtering or aggregation, and can be described as a tree-like data delivery pattern. The data delivery pattern can also be considered as a query execution plan, or query routing tree. In this work, we propose an algebraic optimization of the query routing tree construction and reconfiguration. In particular, we aim at generating query trees that maximize collision-free concurrent data transmissions hence reducing energy and time wastes due to retransmissions. Towards this, we introduce a Data Transmission Algebra (DTA) and apply it for efficient generation of such query trees.

1. Introduction

Database-like query processing over a network of sensors has become an attractive paradigm for both network and database communities [IG99, BS00, BGS01, YG02]. We adopt a broad definition of a sensor network to be a wireless network composed of a large number of sensor nodes most of which are power-constrained [YHE02]. Such sensor nodes are small in size and capable of collecting various measurements such as light, motion, acceleration, and temperature. These sensor nodes can be attached to PDAs or other mobile devices such as mobile robots. In this way, for example, teams of humans and/or mobile robots in conjunction with stationary sensor nodes can be deployed to perform surveillance and tracking, environmental monitoring for highly sensitive areas, or execute search and rescue operations.

Sensors are deployed densely around the phenomenon to be sensed and the data combined from all sensors may aggregate to very high data rates. A *sensor query* is characterized by large data streams among participating nodes with possible in-node data filtering/aggregation, and can be described as a tree-like data delivery pattern (*query routing tree*). Minimizing sensor query response time becomes crucial in mission-critical sensor networks. At the same time, minimizing energy consumption per query is of special concern for these battery-powered devices.

One common source of energy consumption and delay in wireless sensor networks is *packet collisions*. This is also one of the major sources of energy and time waste. In general, wireless sensor nodes that share the same wireless medium are said to be in the same *collision domain (CD)*. Once any two or more nodes in the same CD transmit packets at the same time, a collision occurs, and packets are corrupted and discarded. Packet collisions can be avoided by minimizing the number of intersecting CDs and by synchronizing data transmissions among nodes within the same CD.

In this paper, we propose to reduce the number of intersecting CDs by using a query-driven approach for proper movement and positioning of sensor nodes. We view the problem of proper sensor positioning as equivalent to the problem of construction and reconfiguration of an efficient query routing tree that improves query performance in terms of both response time and energy. Towards this, we introduce a novel query routing tree optimization technique that maximizes collision-free concurrent data transmissions. Our approach is based on algebraic analysis of alternative query routing trees in which regions of movement are defined in terms of CDs.

Recent work that has focused on the construction of energy-efficient routing trees (e.g., for in-network query processing [MFHH02, BSLC03, SBLC03, CJBM01]) does not consider the negative effects of CDs in these trees and assumes that the underlying network layer handles packet collisions. The exception is some work in scheduling of transmissions between levels in a routing query tree with goal of increasing the sleep time (e.g., [CFS03]) of the sensor nodes.

In the next section, we discuss our system model. In Section 3, we propose a *Data Transmission Algebra* (DTA) that can uniformly capture the structure of data transmissions, their constraints and their requirements. In Section 4, we show how the DTA can be used in a cost based optimization to select the best query routing tree. We present an evaluation of our approach in Section 5 and our conclusions in Section 6.

Proceedings of the 15th International Workshop on Database and Expert Systems Applications (DEXA'04)
1529-4188/04 $ 20.00 IEEE

2. Background and System Model

In this paper, we assume that queries originate at a base station which then forwards the queries to the nearest sensor node. This sensor node becomes in charge of disseminating the query down to all the sensor nodes in the network and to gather the results back from all the sensor nodes. Query processing could take place either at the base station and/or within the network, depending on the type of query as well as the sensor nodes' capabilities.

We also assume that a query optimizer executes at the base station. A query optimizer generates alternative query routing trees and mobile sensor deployment plans taking into consideration the current topology of stationary sensor nodes and the applications' coverage requirements. It selects the query routing tree that allows the maximum number of concurrent transmissions without the risk of collisions and disseminates it along with the query.

Mobile sensors are moved into target positions according to the routing query tree to provide coverage while minimizing the intersection of their collision domain (CD). Mobile sensor nodes can freely move for better data sampling but their data transmissions should not violate their CDs defined by their target positions. One way to achieve this is by adjusting their transmission power.

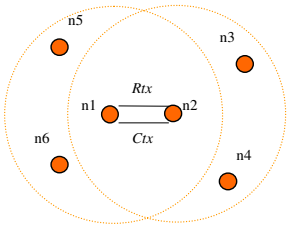


Figure 1: Collision domain of two communicating nodes.

Figure 1 elaborates on the concept of collision domain in a typical wireless network such as IEEE 802.11 and illustrates how collisions are handled in such a network (effectively solving the *hidden node* problem). In order for a sensor node *n1* to communicate with sensor node *n2*, *n1* needs to send first a request for transmission packet (*Rtx*) to *n2*, so that all other nodes in its transmission range (*n5* and *n6* in Figure 1) become aware of the communication and remain silent until *n1* ends the transmission. Sensor *n2* replies to *n1* with a confirmation packet (*Ctx*), so that the nodes in its transmission range (*n3* and *n4* in Figure 1) also become aware of the communication and avoid any transmission until the end of the current transmission. In this case, nodes *n3*, *n4*, *n5*, and *n6* belong to the same CD. In general, any two communicating nodes *ni* and *nj* specify a collision domain CD(*ni,nj*) defined as the union of the transmission ranges of *ni* and *nj*.

In the rest of the paper, for simplicity we assume that coverage is expressed in terms of regions in which at least one sensor node must be positioned in its center.

3. Data Transmission Algebra

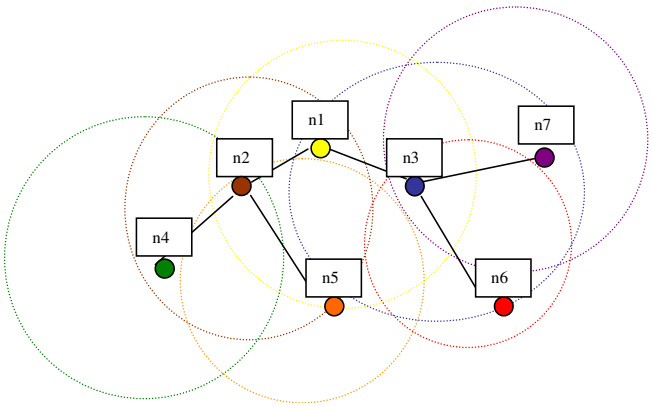
A query routing tree can be considered as a query evaluation plan. This observation motivated us to develop a *Data Transmission Algebra* (DTA) that allows a query optimizer to generate query routing trees to maximize collision-free concurrent data transmissions.

The DTA consists of a set of operations that take transmissions between wireless sensor nodes as input and produce a schedule of transmissions as their result. We call an *elementary transmission* (denoted *ni~nj*) a one-hop transmission from sensor node *ni* to node *nj.* We also use a special symbol, *null,* that denotes a completed (empty) transmission. Each transmission *ni~nj*, which is not empty is associated with a collision domain CD(*ni*, *nj*) as defined above. A transmission schedule is either an elementary transmission, or a composition of elementary transmissions using one of the operations of the DTA. The DTA includes three basic operations that can combine two transmission schedules A and B:

1. **order(A,B) ≡ *o*(A,B)**
   This is a strict order operation, that is, A must be executed before B.
2. **any(A,B) ≡ *a*(A,B)**
   This is an overlap operation, that is, A and B can be executed concurrently.
3. **choice(A,B) ≡ *c*(A,B)**
   This is a non-strict order operation, that is, either A executes before B, or vice versa.
   Thus, *c*(A,B) ≡ (*o*(A,B) or *o*(B,A)).

For an example of the DTA operations consider the query tree in Figure 2 which was generated for some query Q. It shows an *initial* DTA specification that reflects basic constraints of the query tree. The initial specification consists of a set of strict order and overlap operations. For instance, operation O1 specifies that transmission *n2~n1* occurs after *n4~n2* is completed. This constraint reflects the query tree topology. Operation A1 specifies that *n4~n2* can be executed concurrently with *n6~n3*, since neither *n3* nor *n6* belongs to CD(*n4,n2*), and neither *n4* nor *n2* are in CD(*n6,n3*).

We say that two elementary transmissions *et1* and *et2* are potentially concurrent in a query tree *T* if they do not share the same destination, and the initial specification of *T* does not include *o(et1,et2)*, i.e., there is no strict order between *et1* and *et2*.

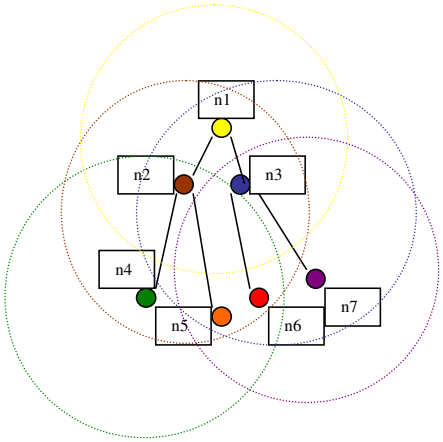


Initial DTA specification:

O1: o(n4~n2, n2~n1)
O2: o(n5~n2, n2~n1)
O3: o(n6~n3, n3~n1)
O4: o(n7~n3, n3~n1)

A1: a(n4~n2, n6~n3)
A2: a(n4~n2, n7~n3)
A3: a(n4~n2, n3~n1)
A4: a(n5~n2, n6~n3)
A5: a(n5~n2, n7~n3)
A6: a(n5~n2, n3~n1)
A7: a(n6~n3, n2~n1)
A8: a(n7~n3, n2~n1)

Figure 2: Example of a query tree and of a corresponding DTA specification



Initial DTA specification:

O1: o(n4~n2, n2~n1)
O2: o(n5~n2, n2~n1)
O3: o(n6~n3, n3~n1)
O4: o(n7~n3, n3~n1)

A2: a(n4~n2, n7~n3)
A3: a(n4~n2, n3~n1)
A8: a(n7~n3, n2~n1)

Figure 3: Example of query tree with lower degree of concurrency

Each operation of the initial specifications defines a simple transmission schedule that consists of two elementary transmissions. The DTA introduces a set of transformation rules that can be used to generate more complex schedules from the initial specification. Figure 4 shows an example of the DTA transformation rules R1-R10, and illustrates how these rules apply to generate more complex schedules A9, A10 and A11 from the initial specification in Figure 2. A9 schedules three elementary transmissions, while each of A10 and A11 schedules four elementary transmissions. None of the simple or complex transmission schedules considered so far includes all elementary transmissions of the query tree, so we call them *partial* schedules. Our goal is to generate DTA expressions for *complete schedules.* A complete schedule includes all elementary transmissions of the query tree.

Example DTA transformation rules:

R1: $o(A,B) \neq o(B,A)$
R2: $a(A,B) = a(B,A)$
R3: $c(A,B) = c(B,A)$
R4: $a(A,B) \,\&\, a(A,C) = a(A, c\,(B,C))$
R5: $c(A, c(B,C)) \,\&\, o(A,B) = c(\, o(A,B), C)$
R6: $c(c(B,C), A) \,\&\, o(B,A) \,\&\, o(C,A) = o(c(B,C), A)$
R7: $o(A,C) \,\&\, o(B,C) = o(\, c(A,B),C)$
R9: $o((A,B), A\,) = o(A,B)$
R10: $c(o(A,B), B\,) = o(A,B)$

***Example of DTA transformations* (Figure 2):**

A1,A2,R4 imply:
A9: a(n4-n2, c(n6~n3, n7~n3));
A3, A9, R4 imply:
A10: a(n4-n2, c(c(n6~n3, n7~n3), n3~n1));
A10,O3,O4,R6 imply:
A11: a(n4-n2, o(c(n6~n3, n7~n3), n3~n1));

Figure 4: Example of DTA transformation rules and DTA transformations

In general, a query tree can be characterized by its degree of *collision-free concurrency* (*cfc*). Query trees are associated with different degrees of *cfc*. We say that query tree Q1 has higher *cfc* than an equivalent query tree Q2, if Q1 allows for more concurrent transmissions without risk of collisions than does Q2. A query tree has *cfc*=1, if it allows for all potential concurrent transmission pairs to occur. Recall that potential concurrent transmission pairs do not share the same destination node and the initial DTA specification does not include a strict ordering.

For example, the query tree in Figure 2 is associated with *cfc*=1, since its initial DTA specification includes all eight of the potentially concurrent transmission pairs (A1-A8). On the other hand, Figure 3 illustrates a query tree with lower degree of *cfc*. Its corresponding initial DTA specification includes only three out of the eight potentially concurrent transmission pairs (A2, A3, and A8). Thus, its *cfc* is equal to 3/8 = 0.375.

Given a query, the coverage requirements, and the position of both stationary and mobile sensors, the *cfc* metric will allow us to shift through all the possible query trees (and mobile sensor positions) in order to generate the candidate trees with relatively high possibility for concurrent data transmissions, while adhering to the stated coverage requirements. Currently, we are developing an efficient way to generate such high-*cfc* trees.

While the *cfc* is a good indicator of the opportunities for concurrency in data transmissions, it does not specify which transmission schedule can materialize the best time and energy savings, given such a high-*cfc* query tree. In order to identify which out of the many transmission schedules would be the best one, we propose using a cost-based tree generation framework, which we describe in the next section.

4. Cost-based Query Tree Generation

In order to generate the best schedule in terms of energy and time efficiency we need to maximize the number of collision-free concurrent data transmissions, while also considering the cost of each of these transmissions. Towards this, we propose using cost-based scheduling techniques to generate the best query tree. In order to achieve this goal, the query optimizer applies the DTA transformation rules in order to estimate the quality of a candidate tree.

In general, the optimizer generates many equivalent trees and selects the one with the minimum estimated cost. Here, the cost corresponds to query execution time associated with a particular schedule. Figure 5 shows simple cost estimation expressions for each of the four basic DTA expressions.

schedule	cost
ni~nj	Tp(*ni*)+Ttx(*ni~nj*)+Tp(*nj*)
o(A,B)	cost(A)+cost(B)
a(A,B)	max(cost(A),cost(B))
c(A,B)	cost(A)+cost(B) – Tf

Figure 5: Estimating costs of schedules

For example, the execution time of elementary transmission *ni~nj* consists of local processing times Tp at nodes *ni* and *nj* plus the time Ttx requires to transmit the data from *ni* to *nj*. Local processing time Tp includes any in-node query processing (as in the case of in-network filtering/aggregation).

Execution time for strict order of schedules A and B, *o(A,B)*, is the sum of execution times for A and for B. The execution time for the overlap of A and B, *a(A,B),* is the maximum of the execution times of the schedules A and B. Finally, the execution time for choice of A and B, *c(a,B)*, is the same as the execution time of a strict order minus a predefined time factor Tf. Tf indicates that the optimizer generally prefers a choice operation over a strict order, since the latter restricts flexibility of the optimizer in query scheduling.

5. Experiments and Analysis

In order to evaluate our approach, we have implemented a DTA optimizer using Arity Prolog 32 version 1.1. We have evaluated our approach in terms of both its *costs* and *benefits.* We define benefit as the part of the time cost that the DTA optimizer is able to "hide" by scheduling transmissions concurrently. The benefit is defined recursively for each of the DTA operations. For example, the benefit of *a(X,Y)* is equal to the minimum of costs *cost(X)* and *cost(Y)*. For the rest of the DTA operations the benefit is equal to zero.

Here we report our findings using a simple, yet illustrative experiment. We do not consider mobility of the sensor nodes explicitly in these preliminary results. We have applied our DTA optimizer to generate transmission schedules of two semantically equivalent binary query trees T1 and T2. Both T1 and T2 have four levels and involve 16 sensors. While being equivalent with respect to coverage, the sensor trees T1 and T2 have *cfc=1* and *cfc=0.7,* respectively, i.e., T1 allows for more concurrency than T2.

For each tree, alternative transmission schedules were generated layer by layer starting from initial schedules with two elementary transmissions (layer 1). Layers 2, 3 and 4 represent schedules with 3, 4 and 5 scheduled transmissions. Layer 5 includes complete schedules covering all elementary transmissions of the query tree. Processing and transmission costs were generated randomly using Gaussian distributions.

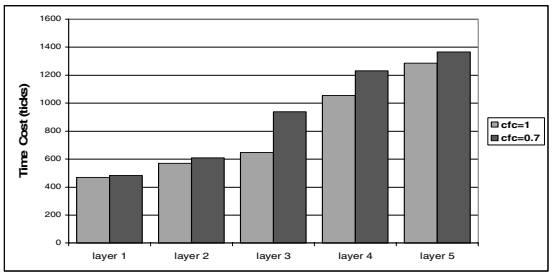


Figure 6: Comparison of Time Cost per scheduling layer for *cfc=1* and *cfc=0.7*

Figure 6 shows the average query execution time in simulation time units (ticks) for the 5 scheduling layers. We compare scheduling for query trees T1 (*cfc=1*) and T2 (*cfc=0.7*). For each scheduling layer we report the average execution time of all its schedules. We observe that at each scheduling layer, T1 outperforms T2.

Figure 7(a) shows the time costs whereas Figure 7(b) shows the relative benefits of the best complete query schedule for each of the query trees T1 and T2, compared to the initial trees. In these figures, T1 again outperforms T2, which is an expected behavior: our cost-based optimizer was able to utilize the possibilites for concurrent transmission that exist in T1 whose *cfc*=1.

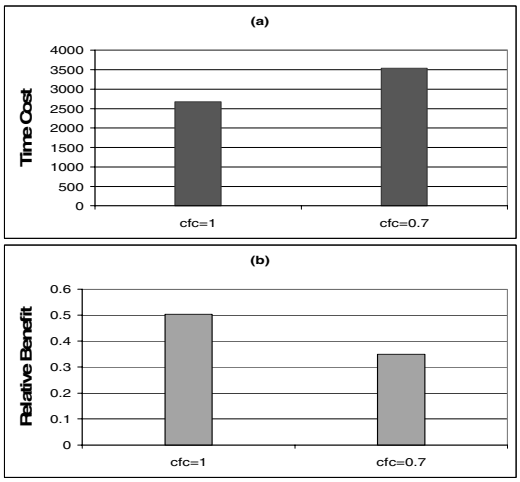


Figure 7: Comparison of Time Cost and Benefit for complete schedules for *cfc=1* and *cfc=0.7*

We can conclude from the above graphs that, in general, query trees with higher *cfc* allow the DTA optimizer to generate more time-efficient schedules. However, in some of our experiments, with a fixed execution time to select a tree, we have observed that the relative time cost *increases* for trees with higher *cfc,* especially in the cases of high complexity (density) trees. The reason is that the additional scheduling flexibility introduced by higher *cfc* may also result in higher variability in time costs among the alternative schedules. As a result, this may increase the risk for the DTA optimizer to choose more expensive query schedules while missing more efficient ones. The DTA optimizer could avoid such a risk by exhaustively enumerating all schedules. Given that exhaustive enumeration is not practical for large trees, we are currently investigating DTA scheduling techniques that would minimize the risk of generating costly schedules for the trees with high *cfc*.

6. Conclusions

Recognizing that packet collisions are a major common source of energy and time waste in mobile sensor networks, we proposed a new framework for producing query routing trees with the highest number of collision-free concurrent transmissions. The crux of our approach is the utilization of cost-based query optimization techniques to globally schedule data transmissions in the network. Our first experimental results have shown that our Data Transmission Algebra optimizer is capable of generating low-energy and time-efficient query trees.

References

[BGS01] P. Bonnet, J. Gehrke, and P. Seshadri. Towards Sensor Database Systems. *Proc. of MDM Conference*, 2001.

[BS00] P. Bonnet and P. Seshadri. Device Database Systems. *Proc. of ICDE Conference*, 2000.

[BSLC03] J. Beaver, M. A. Sharaf, A. Labrinidis, and P. K. Chrysanthis. Location-Aware Routing for Data Aggregation in Sensor Networks. *Proc. of the 2nd Hellenic Data Management Symposium*, 2003.

[CFS03] U. Cetintemel, A. Flinders, and Y. Sun. Power-Efficient Data Dissemination in Wireless Sensor Networks. *Proc. of ACM MobiDE Workshop*, 2003.

[CJBM01] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. *Proc. of Mobile Computing and Networking Conference,* 2001

[IG99] T. Imielinski and S. Goel. DataSpace - Querying and Monitoring Deeply Networked Collections in Physical Space. *Proc. of ACM MobiDE Workshop*, 1999.

[MFHH02] S. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad hoc sensor networks. *Proc. of OSDI*, 2002.

[SBLC03] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. *Proc. of ACM MobiDE Workshop*, 2003.

[YG02] Y.Yao and J.E. Gehrke. The Cougar Approach to In-network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3), 2002.

[YHE02] W. Ye, J. Heidemann, and D. Estrin, "SMAC: An Energy-Efficient MAC Protocol for Wireless Sensor Networks. *Proc. of IEEE INFOCOM Conference*, 2002.

[ZBVRU02] V. Zadorozhny, L. Bright, M.E. Vidal, L. Raschid, and T. Urhan. Efficient Evaluation of Queries in a Mediator for WebSources. *Proc. of ACM SIGMOD* 2002.