Processing Event-Monitoring Queries in Sensor Networks

Vassilis Stoumpos [#1], Antonios Deligiannakis [#2], Yannis Kotidis [*3], Alex Delis [#4]

[#] *Department of Informatics, University of Athens*
Athens, Greece
[1] stoumpos@di.uoa.gr [2] adeli@di.uoa.gr [4] ad@di.uoa.gr

[*] *Department of Informatics, Athens University of Economics and Business*
Athens, Greece
[3] kotidis@aueb.gr

***Abstract*— In this paper we present algorithms for building and maintaining efficient collection trees that provide the conduit to disseminate data required for processing monitoring queries in a wireless sensor network. We introduce and formalize the notion of event monitoring queries and demonstrate that they can capture a large class of monitoring applications. We then show techniques which, using a small set of intuitive statistics, can compute collection trees that minimize important resources such as the number of messages exchanged among the nodes or the overall energy consumption. Our experiments demonstrate that our techniques can organize the data collection process while utilizing significantly lower resources than prior approaches.**

I. Introduction

Many pervasive applications rely on sensory devices that are able to observe their environment and perform simple computational tasks. Driven by constant advances in microelectronics and the economy of scale it is becoming increasingly clear that our future will incorporate a plethora of such sensing devices that will participate and help us in our daily activities. Even though each sensor node will be rather limited in terms of storage, processing and communication capabilities, they will be able to accomplish complex tasks through intelligent collaboration.

It is generally agreed that one cannot simply move the readings necessary for processing an application request out of a large-scale sensor network and then perform the required processing in a designated node such as a *base station*. Wireless sensor nodes have limited energy capacity and such an approach will not only result in overburdening their radio links, but will also quickly drain their energy as radio transmission is by far the most important factor in energy consumption [5]. Thus, most recent proposals rely on building some type of ad-hoc interconnect for answering a query such as the *aggregation tree* [4], [8]. This is a paradigm of in-network processing that can be applied to non-aggregate queries as well [3]. In this paper we concentrate on building and maintaining efficient *data collection trees* that will provide the conduit to disseminate all data required for processing many concurrent queries in a sensor network.

While prior work [2], [6], [7] has also tackled similar problems, previous techniques base their operation on the assumption that the sensor nodes that collect data relevant to the specified query need to include their measurements (and, thus, perform transmissions) in the query result at every query *epoch*. However, in many monitoring applications such an assumption is not valid. Monitoring nodes are often interested in obtaining either the actual readings, or their aggregate values, from sensor nodes that detect interesting events. The detection of such events can often be identified by the readings of each sensor node. For example, in vehicle tracking and monitoring applications high noise levels may indicate the proximity of a vehicle. In all of these scenarios, each sensor node is not forced to include its measurements in the query output at each epoch, but rather such a *query participation* is evaluated on a per epoch basis, depending on its readings and the definition of interesting events. In this paper we term the monitoring queries where the participation of a node is based on the detection of an event of interest as *event monitoring queries* (EMQs). It is important to note that typical monitoring queries, considered in the bulk of research so far, are a subclass of EMQs, as the former correspond to the case where the participation of sensor nodes in the query result at each epoch is fixed (either true, or not) throughout the query execution.

II. Motivation

An important characteristic of EMQs, which is not taken into account by existing algorithms that design collection trees, is that each sensor node may participate in the query evaluation, by including its reading in the query result, only a limited number of times, based on how often the inclusion conditions are satisfied. We can thus associate an *epoch participation frequency* P_i with each sensor node S_i, which specifies the fraction of epochs that this node participated in the query result in the recent past.

A. Intuition

Given estimates of the epoch participation frequencies, one can design significantly more efficient collection trees than prior approaches. Consider the sample scenario depicted in Figure 1(a). In this figure, 36 sensor nodes are placed in a grid. The sensor identifiers appear next to each sensor node.

978-1-4244-1837-4/08/$25.00 © 2008 IEEE

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 30, 2008 at 13:54 from IEEE Xplore. Restrictions apply.

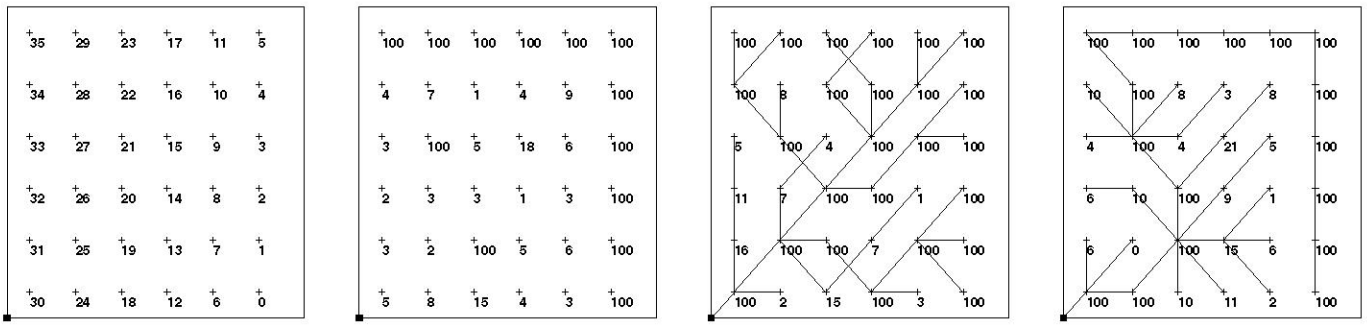


Fig. 1. (a) Identifiers of sensors in grid arrangement; (b) Estimated number of participations in query result in 100 epochs; (c) Collection tree for MinHops algorithm. Cost = 2579 transmissions; (d) Collection tree for our algorithm. Cost = 1839 transmissions.

We also distinguish the `Root` node at the lower left corner, a monitoring node that performs queries over the data collected by the sensor nodes. In our sample network we assume that each sensor node can communicate with its immediate horizontal, vertical or diagonal neighbors, while only node S_{30} can communicate with the `Root` node. In Figure 1(b) we depict sample estimates for the number of times each sensor node will participate in the query result within the next 100 epochs. Thus, the epoch participation frequencies for all the sensor nodes, can be derived by dividing these values by 100. In the above scenario, given the presented epoch participation frequencies, two interior nodes along with all the boundary nodes on the upper and rightmost edges of the network always detect events, while the remaining interior nodes detect events with a lower probability, whose average value is about 5%. For the aforementioned sample scenario, in Figure 1(c) we depict a sample collection tree chosen by an algorithm, termed as *MinHops* that seeks to minimize the number of hops that each node's data needs to traverse until it reaches the `Root` node. Next to each node we depict the actual number of transmissions that each node performed within these 100 epochs. Similarly, in Figure 1(d) we present the collection tree that our algorithms created for the evaluation of the SUM aggregate. A significant observation is that our algorithm seeks to forward the query results from nodes with high epoch participation frequencies through a limited number of interior nodes, compared to the *MinHops* algorithm. One can easily establish the significant reduction in the number of transmissions that our algorithm achieved (1839 vs 2579 or, equivalently, a 40% reduction).

A first observation is that our algorithm seeks to forward the query results from nodes with high epoch participation frequencies through a limited number of interior nodes, compared to the *MinHops* algorithm. Moreover, we note that our algorithm does not necessarily route its messages through the neighboring node with the highest epoch participation frequency. For example, node S_9 has chosen to forward its results through node S_{14} and not through nodes S_{15} or S_8, even though $P_{14} = 1\%$ is lower than both $P_{15} = 18\%$ and $P_8 = 3\%$. We note that what is important when forming the collection tree is the estimated impact that each of the algorithm's decisions has on the desired minimization metric. For example, when selecting a *parent* node in the collection tree, the impact may involve (depending on the minimization metric) the estimated increase in the transmitted messages in the path from the parent node towards the `Root` node. Given this brief explanation, we note in our example that node S_{14} is an immediate neighbor of S_{19}, a node with a high epoch participation frequency. Thus, since the nodes in the path of S_{19} to the `Root` node will probably be forced in making several transmissions, due to the events transmitted by S_{19}, the additional messages transmitted by having S_9 select S_{14} as its parent node will mostly influence the path between S_{14} and S_{19}. Such an observation cannot be made for S_{15} or S_8, since they lie one hop further from a node with a high epoch participation frequency.

B. Algorithm Idea

The algorithm is initiated with the query propagation phase. The query is propagated from the base station through the network using a flooding algorithm. In densely populated sensor networks, a node S_i may receive the announcement of the query from several of its neighbors and selects one of these nodes as its *parent node*. The chosen parent will be the one that exhibits the lowest *attachment cost*, meaning the lowest expected increase in the objective minimization function. For example, if our objective is to minimize the total number of transmitted messages, then the selection will be the node that is expected to result in the lowest increase in the number of transmitted messages in the *entire* path from that sensor until the `Root` node (and similarly for the rest of the minimization metrics). The result of this process is a collection tree towards the base station. A key point in our framework is that the preliminary selection of a parent node may be revised in a second step where each node evaluates the cost of using one of its sibling nodes as an alternative parent.

III. Experiments

We developed a simulator for testing the algorithms proposed in this paper under various conditions. In our discussion we term our algorithm for minimizing the number of

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 30, 2008 at 13:54 from IEEE Xplore. Restrictions apply.

	Aggregate SUM Query			
Sensors	MinMesg	MinEnergy	MinHops	MinCost
36	109.339	109.341	161.278	136.354
144	70.129	68.971	139.821	121.640
324	71.662	68.703	146.425	106.416
576	65.921	64.717	127.315	104.156
900	67.107	64.077	128.299	102.708
	Non-Aggregate "SELECT *" Query			
Sensors	MinMesg	MinEnergy	MinHops	MinCost
36	381.483	292.231	335.920	303.270
144	515.215	344.489	390.806	344.213
324	687.083	444.157	523.670	448.816
576	624.817	457.788	547.147	471.845
900	756.902	549.262	640.830	559.183

TABLE I

AVERAGE POWER CONSUMPTION (IN MJ) FOR SYNTHETIC DATASET

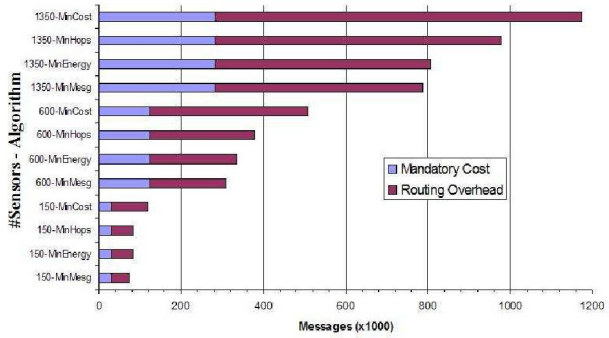


Fig. 2. Transmissions - Trucks data

transmissions as *MinMesg*, and our algorithm for minimizing the overall energy consumption as *MinEnergy*. Our techniques are compared against two intuitive algorithms. In the *MinHops* algorithm, each sensor node that receives the query announcement randomly selects as its parent node a sensor amongst those with the minimum distance, in number of hops, from the `Root` node [4]. In the *MinCost* algorithm, each sensor seeks to minimize the sum of the squared distances amongst the sensors in its path to the `Root` node, when selecting its parent node i.e. selects paths with low communication cost.

We initially experimented with synthetic datasets: we placed 36 sensor nodes in a 300x300 area, and then scaled up to the point of having 900 sensors. The maximum broadcast range was set to 90m and the `Root` node was placed on the lower left part of the sensor field. We set the epoch participation frequency of the sensor nodes with the maximum distance, in hop count, from the `Root` to 1; with probability 8% some interior node assumed an epoch participation frequency of 1, while the epoch participation frequency of the remaining interior nodes was set to 5%. We evaluated a SUM aggregate and a "SELECT *" non-aggregate query over the values of epoch participating sensor nodes using all algorithms and found that the MinEnergy algorithm built very different collection trees for the two types of queries. Our *MinMesg* algorithm achieves a significant reduction in the number of transmitted messages which compared to the *MinHops* and *MinCost* algorithms is up to 64% and 105%, respectively, with an average gain of 48% and 94%, respectively. The corresponding average energy consumption by the sensor nodes for each case is presented in Table I. The MinEnergy algorithm performs very well in both aggregate and non-aggregate queries. Compared to the MinHops algorithm, it achieves up to a 2-fold reduction in the power drain for aggregate queries and up to 19% for non-aggregate queries. Compared to the MinCost algorithm the energy savings are smaller but still significant (i.e., up to 79% in the aggregate query). The MinMesg algorithm is obviously a very poor choice, with respect to the energy consumption, for non-aggregate queries.

We also present results for the **Trucks** data set that contains trajectories of 276 moving trucks [1]. For this data set we initially overlaid a sensor network of 150 nodes over the monitored area. We set the broadcast range such that interior sensor nodes could communicate with at least 5 more sensor nodes. Moreover, each sensor could detect objects within a circle centered at the node and with radius equal to 60% of the broadcast range. We then scaled the data set up to a network of 1350 sensors, while keeping the sensing range steady. In Figure 2 we depict the total number of transmissions by all algorithms for computing the SUM of the number of detected objects in the Trucks data set. In our scenario, nodes that do not observe an event make a transmission only if they need to propagate measurements/aggregates by descendant nodes. We found our algorithms to achieve significant savings in both metrics: the MinCost algorithm, which exhibits lower power consumption than the MinHops algorithm, still drains 50% more energy than our MinEnergy algorithm. Moreover, both our MinMesg and MinEnergy algorithms significantly reduce transmitted messages by up to 42% and 73% when compared to the MinHops and MinCost algorithms, respectively.

IV. CONCLUSIONS

The focus of this work is on building and maintaining efficient collection trees in support of event monitoring queries in wireless sensor networks. Our experimental evaluation demonstrates that is it possible to create efficient collection trees that minimize important network resources using a small set of statistics that are communicated in a localized manner during the construction of the tree topology. Our algorithms can handle a mix of event monitoring queries (EMQs) including aggregate and non-aggregate queries.

REFERENCES

[1] Rtree Pportal. http://www.rtreeportal.org.

[2] Jae-Hwan Chang and Leandros Tassiulas. Energy Conserving Routing in Wireless Ad-hoc Networks. In *INFOCOM*, 2000.

[3] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Dissemination of Compressed Historical Information in Sensor Networks. *VLDB Journal*, 16(4), 2007.

[4] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A Tiny Aggregation Service for ad hoc Sensor Networks. In *OSDI Conf.*, 2002.

[5] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The Design of an Acquisitional Query processor for Sensor Networks. In *ACM SIGMOD*, 2003.

[6] S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *MobiCom*, 1998.

[7] N. Trigoni, Y. Yao, A.J. Demers, J. Gehrke, and R. Rajaraman. Multi-query Optimization for Sensor Networks. In *DCOSS*, 2005.

[8] Y. Yao and J. Gehrke. The Cougar Approach to In-Network Query Processing in Sensor Networks. *SIGMOD Record*, 31(3):9–18, 2002.

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 30, 2008 at 13:54 from IEEE Xplore. Restrictions apply.