Sensor Selection for Resource-Efficient Query Execution in IoT Environments

Maria G. Koziri
Dept. of Computer Science
Univ. of Thessaly
Lamia, Greece
email: mkoziri@uth.gr

Thanasis Loukopoulos
Dept. of Computer Science and Biomedical Informatics
Univ. of Thessaly
Lamia, Greece
email: luke@dib.uth.gr

***Abstract*—In an IoT environment, the geographically dispersed sensors that are eligible for participating in a spatial query, can scale to the orders of millions or even billions. Therefore, judiciously selecting among the candidates is of paramount importance to reduce query complexity. Such selection must minimize the total resources used while maintaining the highest possible accuracy in results. In this paper, we turn our attention to the problem of assigning query filters over a subset of the available sensor nodes, assuming that queries are resident in the system, e.g., performing monitoring activities. We present a rigorous problem formulation that captures the dependencies between query accuracy, and resource consumption, focusing in particular on energy consumption. The relevant decision problem is shown to be NP-complete, thus, we propose a heuristic based on the greedy method to solve it. Simulation experiments show that compared to an algorithm that performs random assignments, significant improvement by more than 100% (resource wise) is expected.**

Keywords-sensor networks; IoT; sensor selection; query plan; energy efficiency; resource consumption.

I. INTRODUCTION

As the number of smart devices has exceeded the population of earth and is still growing at a fast pace, the premise of IoT [5] is to enable (among others) the interoperability of the various devices that could act as potential sensors and/or actuators. At the same time, the advent of cyber physical systems (CPS) [8] that combine the physical sensors and actuators with the cyber world provide a novel ground for smart applications where the needs for interoperability and efficient resource allocation are of paramount importance.

Of particular interest in such huge scale systems is the problem of efficient spatial query execution. Consider for instance a system that gathers temperature information at the various city districts and sends warnings to health authorities in the case of extreme conditions. A simple strategy whereby all the available sensors are involved in the query (assuming a large number of them), will likely lead to waste of resources at the sensors and increased network load. In contrast, using only a subset of the available sensors per involved district location, might lead to results of almost equal quality, while saving resources.

In this paper, we consider a generic sensor system running monitoring spatial queries that involve (among others) sensor locations. We tackle the problem of sensor selection, with the goal of achieving sufficient query accuracy, while minimizing the total energy consumed, thus, improving the lifetime of the sensing nodes (assuming they operate on battery). Specifically, we illustrate a rigorous formulation for the problem and propose a greedy algorithm to solve query–sensor assignment.

The remaining of the paper is organized as follows. Section II discusses the related work. The problem formulation is illustrated in Section III, while a greedy heuristic approach is presented in Section IV. Experimental results are included in Section V. Finally, Section VI concludes the paper.

II. RELATED WORK

Our work is also closely related to in-network query processing where the problem is to assign the operators comprising a query in the network nodes. In [4] and [14], operator placement was discussed in the context of WSNs with the aim of optimizing routing cost in the query tree, while in [25] a more generic approach that aims at placing general purpose application trees is proposed.

The effects of query operator placement are largely dictated by data availability in the processing node. Techniques, such as caching [10] and data replication [13] were studied in the past in order to move data closer to where they will be required. In [26], operator placement is considered together with data caching. Query caching is also the aim of [12] where the potential of caching OLAP queries at the level of Internet proxies was examined, while in [18] caching is considered at the level of a single cell in a cellular network.

Much research has been devoted in the past on developing suitable middleware and programming frameworks in the context of wireless sensor networks. Example works include the systems described in [11][19][24] to name a few. [19] provides an adaptive mechanism for efficient data fusion and filtering. Optimal resource allocation for filtering in a distributed system is discussed in [2]. The systems of [11] and [24] are broader in scope in the sense that they model an application as a set of communicating mobile agents, that can carry any type of functionality, e.g., sensing, actuating, aggregating or controlling etc. Both systems attempt to reduce network overhead using different algorithmic techniques. Another system of similar scope, i.e., mobile agent framework for WSN is [1]. Compared to [11] and [24], it tackles equivalent

 ISBN: 978-1-61208-529-6

application scope, nevertheless, it lacks similar mechanisms for network communication reduction. In the context of IoT, agent based systems include [3] and [7]. The major challenge tackled by these systems is the interoperability of heterogeneous sensing and computing nodes. As demonstrated by the systems the agent framework provides a suitable abstraction layer for integrated applications.

Adaptive resource management in sensor networks is discussed in [6], [15] and [23]. In [23], the grouping of sensors into predefined number of clusters is discussed. [6] proposes an adaptive scheme that dynamically adjusts sampling rate in the sensor network, while [15] discusses resource/sensor allocation to cope with peaks in sampling rates. A survey on the issue of adaptive sensor network organization can be found in [21]. Of particular interest is the case where the sensor network is comprised of cameras meaning that the data to be transmitted is of high volume compared to for example monitoring temperature. In [22], methods to efficiently perform monitoring over a camera network are discussed, while [9] is rather orthogonal, studying the reduction in network consumption that is achievable in social media networks by using new video coding standards. It is worth noting that social media comprise key components of cyber physical systems and therefore any resource savings are cumulative to the ones achieved at the sensor network level.

In a previous work of ours [17], we implemented a framework that enabled for communication and coordination of various smart devices through the remote invocation of applications on them. In this paper, we envision that a voluntary participation scheme is in effect and that all the required functionality by each participant is coded as a native application. This is for instance the case with smart city environments such as [16]. A central administration entity tracks the geographic locations of participants, for instance by using some of the efficient spatial indexing schemes proposed in the literature, e.g., [20] and is responsible for selecting the nodes to participate in a system query. In the sequel, we describe the criteria and the optimization problem induced by the aforementioned selection.

III. PROBLEM FORMULATION

Let Q be the total number of queries to be executed in the system. Each query Q_i has S_i selection predicates. Let S be the total number of predicates from all queries in the system. Clearly:

$$S = \sum_{i=1}^{Q} S_i \tag{1}$$

Assuming a total ordering of the S predicates, we denote with F_k the k^{th} such predicate. Let A be a binary $Q\times S$ matrix encoding whether the predicate F_k is used by Q_i as follows: A_{ik}=1 iff Q_i contains F_k, otherwise, A_{ik}=0. Each F_k can be assigned over a number of sensing nodes (if compatible), resulting into multiple streams of data (equaling the assigned sensors) being transmitted to a base station for filtering and joining. This model reflects the scenario where sensing nodes have direct Internet connections, e.g., smart devices under cellular networks.

Let the total number of participating sensing nodes be N, and N_j denote the j^{th} such, assuming a total ordering of them. Let a binary $N\times S$ matrix C encode whether a node N_j is compatible with F_k predicate as follows: C_{jk}=1 iff N_j is compatible with F_k and C_{jk}=0, otherwise. We should note that a node might be compatible with more than one predicates. To explain it, consider a query that returns the average temperature and humidity from two different location areas. The query can be viewed as containing four predicates, i.e., the combinations of the two locations and the two measured parameters. Depending on the measurement power of a sensing node it can participate in one or more predicates (max two if location areas are disjoint).

Each N_j has a resource level $r(N_j)$ representing a generic metric of the node's processing power or energy levels in case nodes run on battery and energy preservation is deemed the most important factor. Similarly, each predicate F_k requires a resource consumption of $r(F_k)$. We would like to mention that assuming constant resource consumption by query predicates is not far from reality. For instance, in camera networks, video feeds are usually transmitted and processed at bitrates that remain almost constant.

In order for a query Q_i to execute properly, a predicate F_k must be assigned to at least R_{ik} nodes. Clearly R_{ik}=0 iff A_{ik}=0. Let B be a $Q\times S\times N$ matrix encoding the potential query benefit, whereby B_{iku} is the benefit of assigning u sensors at the F_k predicate for Q_i. We assume that: B_{iku}=0 for all $u<R_{ik}$ and $B_{iku}>0$ for $u\geq R_{ik}$.

Finally, let a Boolean matrix X of size $N\times S$ encode the predicate to node assignment as follows: X_{jk}=1 iff F_k is assigned to N_j and 0 otherwise. We are now in position to formulate the selection problem as a two function optimization one, whereby the first function aims at maximizing query benefit and the second function aims at minimizing the maximum proportional resource reduction at a node. The following equations depict the target functions:

$$D1 = \max\left\{\frac{\sum_{k=1}^{S} X_{jk} r(F_k)}{r(N_j)}, \qquad 1 \leq j \leq N\right\} \tag{2}$$

$$D2 = \sum_{i=1}^{Q}\sum_{k=1}^{S} A_{ik}\, B_{iku} \;\mid u = \sum_{j=1}^{N} X_{jk} \tag{3}$$

Therefore, the problem can be posed as follows: *Find X such that (2) is minimized and (3) is maximized, subject to the following constraints*:

$$\sum_{j=1}^{N} X_{jk} \geq R_{ik} \quad, \forall 1 \leq i \leq Q, \forall 1 \leq k \leq S \tag{4}$$

$$X_{jk} \leq C_{jk} \quad, \forall 1 \leq j \leq N, \forall 1 \leq k \leq S \tag{5}$$

$$\sum_{k=1}^{S} X_{jk} r(F_k) \leq r(N_j), \forall 1 \leq j \leq N \tag{6}$$

Constraint (4) effectively dictates that a sufficient number of sensors must be allocated to each predicate. Constraint (5) captures compatibility restrictions between predicates and sensors. Notice that an incompatible node (C_{jk}=0) leads to X_{jk}=0, i.e., the predicate will not be assigned to the node. Constraint (6) captures node capacity, i.e., it forbids the assignments to a node that would require resource consumption greater than the available one.

Before closing the section we would like to mention that the two objectives *D*1 and *D*2 are conflicting with each other. It is easy to observe that *D*1 is minimized when equality holds in (4), while *D*2 is maximized when all eligible sensors participate in a predicate (assuming that B_{iku} entries grow monotonically to u).

When optimizing two target functions one can resort to designing algorithms that produce a set of Pareto optimal solutions to choose from. Instead, we decided to convert the two function optimization problem into a single function optimization one by introducing a weighting constant α. In particular, let $min\{B_{ik}\}$ denote the (minimum) benefit when u=R_{ik} and $max\{B_{ik}\}$ the maximum such value when:

$$u = \sum_{j=1}^{N} C_{jk} \tag{7}$$

Then, b_{ik} denotes the average max to min benefit ratio for a predicate F_k at Q_i as follows:

$$b_{ik} = max\{B_{ik}\}/min\,\{B_{ik}\} \tag{8}$$

For each predicate we calculate the benefit to resource ratio (let r_{ik}) as follows:

$$h_{ik} = \left(\sum_{j=1}^{N} C_{jk}\right)/R_{ik} \tag{9}$$

$$r_{ik} = b_{ik}/h_{ik} \tag{10}$$

Clearly, (8) and (9) hold if F_k is used by Q_i, i.e., $R_{ik}\neq 0$. Then we can get an estimation of the total benefit to resource consumption ratio as follows:

$$r = \frac{\sum_{k=1}^{S}\sum_{\forall i}^{A_{ik}=1} r_{ik}\, r(F_k)}{\sum_{k=1}^{S} r(F_k)} \tag{11}$$

When comparing the maximum and minimum assignment policies r represents the ratio of benefit gains to resource consumption increase. Thus, we can identify the constant α as a proportion of r (depicting how much query benefits are favored over resource consumption).The following two equations summarize the problem formulation which now targets at maximizing the composite function *D*:

$$a = dr \tag{12}$$

$$D = \frac{a}{D1} + D2 \tag{13}$$

Solving the problem as formulated using (13) (but also in the formulation that uses (2) and (3)) can be shown to be NP-hard. In particular, it can be shown that it contains a 2-processor scheduling component as far as (2) is concerned. For this reason we resort to heuristics in order to compute solutions. In the sequel, we describe one such heuristic.

IV. Greedy Algorithm

The algorithm presented in this section is based on the greedy paradigm. It works in two steps. In the first step, it covers the constraint expressed in (4), i.e., assign just enough sensors to meet the demand for each predicate. This is done with respect to constraints (5) and (6) as follows. First, all predicates are sorted according to $r(F_k)$. Then, starting with the one with the highest resource requirement, it assigns it to R_{ik} different sensors in an iterative manner. At each iteration, all eligible sensors are considered and the assignment that incurs the minimum cost as per (2) is selected.

Having satisfied constraint (4) the algorithm then proceeds by optimizing (13) in an iterative manner. At each iteration all possible predicate to sensor assignments are considered and the one that maximizes most (13) is selected and implemented. The process continues until no eligible sensor-predicate assignments exist or (13) can't be further improved.

V. Experiments

We conducted simulation experiments using the following setup. We fixed the number of queries (unless otherwise stated) to 100, each with a predicate number varying uniformly between 3 and 10. Predicates required between 20 and 100 sensing nodes (randomly chosen) to execute properly and incurred resource consumption between 1 and 10 units. We assumed a total number of sensing nodes equaling 10,000, with each of them being compatible to 1/10th of the total predicates (randomly selected). Sensing nodes had resource capacity of 100 load units.

We compared the performance of greedy algorithm as opposed to random selection for the case where each predicate F_k must be assigned to R_{ik} nodes exactly. Results depict the average of 10 runs.

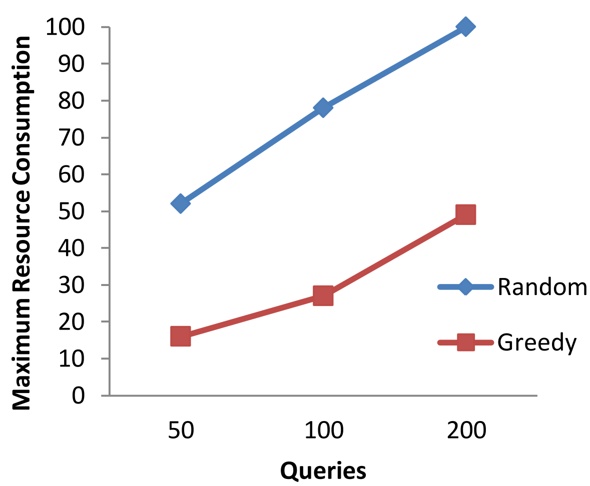

Figure 1. Maximum resource consumption for increasing number of queries (sensor nodes=10,000).

ISBN: 978-1-61208-529-6

Figure 1 shows the results for the baseline scenario of 100 queries over 10,000 sensors as well as the performance when the number of queries are doubled or halved keeping the rest of the setup the same. It is evident that the performance difference between Greedy and Random is significant. In most cases, Greedy incurs a maximum resource consumption of less than half compared to Random. As expected peak resource consumption rises to the number of queries introduced in the system.

To further confirm the viability of the proposed heuristic in Figure 2 we plot the performance of Greedy and Random as the number of available sensors increases. Notice, that the maximum resource consumption with Greedy exhibits a steeper decline compared to Random when moving from the baseline scenario to the one having double the sensors.

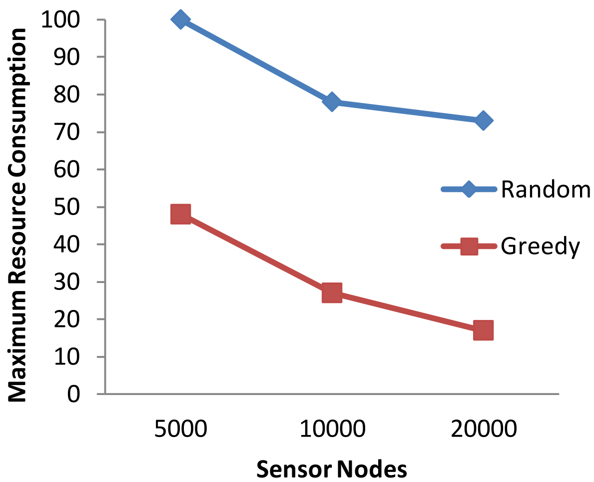

Figure 2. Maximum resource consumption for increasing number of sensor nodes (queries=100).

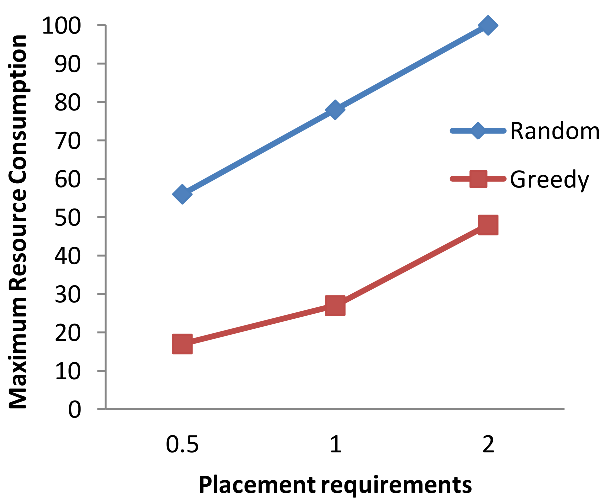

Figure 3. Maximum resource consumption for increasing placement requirement of predicates (queries=100, sensor nodes=10,000).

Last, in Figure 3 we evaluated the performance of the algorithms when each predicate exhibits half (0.5 in x-axis) and double (2 in x-axis) the requirements for sensors to be placed at, compared to the baseline scenario (1 in x-axis). Results are comparable to the ones exhibited in the Figures 1 and 2, with Random incurring between 2 and 3 times more overhead compared to Greedy.

VI. CONCLUSIONS

In this paper, we tackled the problem of assigning query operators to sensing nodes in IoT environments, whereby a huge number of potential participants exist. We provided a rigorous problem formulation that captures typically the trade-off between increasing quality of query results and resource consumption. We proposed a heuristic based on the Greedy paradigm to tackle the problem and compared its performance against Random assignment. Preliminary experimental results indicate that Greedy incurs between half and one third of the overhead of Random.

ACKNOWLEDGMENT

This work was partially supported by travel grant of the postgraduate program: “Informatics and Computational Biomedicine”, School of Science, Univ. of Thessaly.

REFERENCES

[1] F. Aielo, G. Fortino, R. Gravina, and A. Guerrieri, “A Java-based Agent Platform for Programming Wireless Sensor Networks,” The Computer J., vol 54(3), pp. 439-454, 2010.

[2] N. Assimakis, M. Adam, M. Koziri, S. Voliotis, and K. Asimakis, “Optimal decentralized Kalman filter and Lainiotis filter,” Digital Signal Processing, vol. 23(1), pp. 442-452, 2013.

[3] I. Ayala, M. Amor, and L. Fuentes, “The Sol agent platform: Enabling group communication and interoperability of self-configuring agents in the Internet of Things,” JAISE, vol. 7(2), pp. 243-269, 2015.

[4] G. Chatzimilioudis, A. Cuzzocrea, D. Gunopoulos, and N. Mamoulis, “A Novel Distributed Framework for Optimizing Query Routing Trees in Wireless Sensor Networks via Optimal Operator Placement,” J. of Computer and System Sciences, vol. 79(3), pp. 349-368, 2013.

[5] J. Gubbi, R Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol. 29(7), pp. 1645-1660, 2014.

[6] N. Hu, T. F. La Porta, and N. Bartolini, “Self-Adaptive Resource Allocation for Event Monitoring with Uncertainty in Sensor Networks,” MASS 2015, pp. 370-378.

[7] E. Jung, I. Cho, and S. M. Kang, “iotSilo: The Agent Service Platform Supporting Dynamic Behaviour Assembly for Resolving the Heterogeneity of IoT,” Int. J. of Distributed Sensor Networks, vol. 10, 2014.

 ISBN: 978-1-61208-529-6

[8] S. K. Khaitan and J. D. McCalley, “Design Techniques and Applications of Cyber Physical Systems: A Survey,” IEEE Systems Journal, vol. 9(2), pp. 350-365, June 2015.

[9] M. G. Koziri, et al., “A framework for scheduling the encoding of multiple smart user videos,” SMAP 2016, pp. 29-34.

[10] N. Laoutaris, G. Smaragdakis, A. Bestavros, I. Matta, and I. Stavrakakis, “Distributed Selfish Caching,” IEEE TPDS, vol. 18(10), pp. 1361-1376, 2007.

[11] H. Liu, T. Roeder, K. Walsh, R. Barr, and E. G. Sirer, “Design and implementation of a single system image operating system for ad hoc networks,” in Proc. MobiSys 2005, pp. 149-162.

[12] T. Loukopoulos and I. Ahmad, “Policies for Caching OLAP Queries in Internet Proxies,” IEEE TPDS, vol. 17(10), pp. 1124-1135, 2006.

[13] T. Loukopoulos, N. Tziritas, P. Lampsas, and S. Lalis, “Implementing Replica Placements: Feasibility and Cost Minimization,” IPDPS 2007, pp. 1-10.

[14] Z. Lu, Y. Wen, R. Fan, S. - L. Tan, and J. Biswas, “Toward Efficient Distributed Algorithms for In-Network Binary Operator Tree Placement in Wireless Sensor Networks,” IEEE JSAC, vol 31(4), pp. 743-755, 2013.

[15] N. Nguyen and M. M. H. Khan, “A closed-loop context aware data acquisition and resource allocation framework for dynamic data driven applications systems (DDDAS) on the cloud,” Journal of Systems and Software, vol. 109, pp. 88-105, 2015.

[16] F. Paganelli, S. Turchi, and D. Giuli, “A Web of Things Framework for RESTful Applications and Its Experimentation in a Smart City,” IEEE Systems Journal vol. 10(4), pp. 1412-1423, 2016.

[17] P. Papadopoulos, T. Loukopoulos, I. Anagnostopoulos, N. Tziritas, and M. Vassilakopoulos, “RAC: a remote application calling framework for coordination of mobile apps,” PCI 2015, pp. 394-399.

[18] K. Poularakis, G. Iosifidis, and L. Tassiulas, “Approximation algorithms for mobile data caching in small cell networks,” IEEE Trans. on Communications, vol. 62(10), pp. 3665-3677, 2014.

[19] U. Ramachandran, et al., “Dynamic data fusion for future sensor networks,” ACM TOSN, vol. 2(3), pp. 404-443, 2006.

[20] G. Roumelis, M. Vassilakopoulos, T. Loukopoulos, A. Corral, and Y. Manolopoulos, “The xBR^+ -tree: An Efficient Access Method for Points,” DEXA (1) 2015, pp. 43-58.

[21] C. Sengul, A. C. Viana, and A. Ziviani, “A survey of adaptive services to cope with dynamics in wireless self-organizing networks,” ACM Comput. Surv. 44(4), pp. 23:1-23:35, 2012.

[22] P. J. Shin, J. Park, and A. C. Kak, “A predictive duty cycle adaptation framework using augmented sensing for wireless camera networks,” ACM TOSN, vol. 10(2), pp. 22:1-22:31, 2014.

[23] M. N. Tahan, M. Dehghan, and H. Pedram, “Upper and lower bounds for dynamic cluster assignment for multi-target tracking in heterogeneous WSNs,” JPDC vol. 73(10), pp. 1389-1399, 2013.

[24] N. Tziritas, et al., “Middleware Mechanisms for Agent Mobility in Wireless Sensor and Actuator Networks,” S-CUBE 2012, pp. 30-44.

[25] N. Tziritas, T. Loukopoulos, S. U. Khan, and C. - Z. Xu, “Distributed Algorithms for the Operator Placement Problem,” IEEE TCSS, vol. 2(4), pp. 182–196, Feb. 2015.

[26] L. Ying, Z. Liu, D. Towsley, and C. H. Xia, “Distributed Operator Placement and Data Caching in Large-scale Sensor Networks,” INFOCOM 2008, pp. 977-985.