Ad Hoc Networks 47 (2016) 82–98

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Efficient and robust serial query processing approach for large-scale wireless sensor networks

A. Boukerche [a], A. Mostefaoui [a,b,*], M. Melkemi [c]

[a] PARADISE Research Laboratory, University of Ottawa, 800 King Edward Avenue Ottawa, Ontario, K1N 6N5, Canada
[b] Femto-ST Institute, University of Franche-Comte, 1 Rue Engel Gros, Belfort 90000, France
[c] LMIA, University of Haute-Alsace, 4 rue des freres Lumiere, 68093, Mulhouse, France

ARTICLE INFO

Article history:
Received 21 November 2015
Revised 9 April 2016
Accepted 28 April 2016
Available online 10 May 2016

Keywords:
Wireless sensor networks
Query processing
Serial algorithms
Localized algorithms
Robustness

ABSTRACT

The goal of query processing in WSN is to get reliable information of interest from sensor nodes whilst preserving, as much as possible, the network resources, mainly energy. Among the various approaches proposed in the literature to tackle this issue, serial approaches, in which the query is carried out serially from node to node, have shown noticeable improvements in terms of query processing responsiveness and communication overhead reduction when compared to centralized and distributed ones. Nevertheless, they suffer two main drawbacks: (a) they are intrinsically very vulnerable and (b) they require the construction of a Hamiltonian path through the network, which is known to be a NP-Complete problem. In this paper, we investigate the issue of efficient and robust query processing by proposing a novel approach, which we refer to as GBT (Greedy & Boundary Traversal). GBT is of a serialized and localized nature (i.e., each node does not maintain any knowledge about the topology of the network). Furthermore, in GBT the selection of the next hop is totally independent from the previous hops (i.e., no path is defined in advance). This feature enforces the robustness of GBT as attested by the simulation results we obtained (a mean improvement of almost 50% reduction in terms of communication, energy and query responsiveness in large-scale network topologies). We also provide a complexity analysis (time, space and communication) of our query processing algorithm as well as formal proof of its correctness (i.e., termination and completeness).

1. Introduction

Wireless Sensor Networks (WSNs) consist of numerous sensors (from several hundred to several thousand) deployed in a region of interest in order to monitor either physical or environmental conditions according to the requirements of an application. They are capable of generating and collecting an enormous amount of data. Although the ultimate goal is to derive information of interest through queries sent to nodes containing such data, the used method to process these queries has a great impact on the overall performance of the network, particularly when the sensors have limitations on their energy.

This paper deals with query processing in WSNs: upon requesting information from the base station, also called *Sink*, sensor nodes in the network collect data and send it back to the sink, which is in charge of answering end-application/user queries. This process is referred to as *query processing* [1]. Because of its inherent constraints (i.e., limited energy, low communication and node reliability, large deployment, etc.), the way the network performs this task has a deep impact on its overall performance (e.g., required communications, consumed energy, query responsiveness, etc.).

One of the most obvious ways to handle queries in sensor networks is known as warehousing [2], also called centralized approaches, where the access to the sensor network and the processing of queries are separated. In other terms, raw data is first collected at the sink, whereas a centralized DBMS can be used to provide access to the collected data through classical database queries. While the warehousing approach has the advantage of being simple and efficient in terms of answering queries, it suffers from two limiting drawbacks: accumulation of highly redundant data at the sink, and more importantly, over-utilization of communications resources in the WSN. As communications are known to be the most energy consuming tasks in the WSN [1], their usage must then be handled carefully in order to prolong the network

* Corresponding author. Fax: +33381994661.
E-mail addresses: boukerch@site.uottawa.ca (A. Boukerche), ahmed.mostefaoui@univ-fcomte.fr (A. Mostefaoui), Mahmoud.melkemi@uha.fr (M. Melkemi).

http://dx.doi.org/10.1016/j.adhoc.2016.04.012
1570-8705/

lifetime, since sensors are driven by limited energy provisions (i.e., batteries). Hence, instead of sending all individual data to the sink, it would be more efficient, in terms of reducing communications and consequently reducing consumed energy, to partially process the query locally and send this information as a single packet to the sink. This process is referred as *in-network processing*.

Three main design considerations have to be taken into account when developing an in-network query processing technique: (a) its **cost** in term of resources usage, mainly energy, (b) its ability to scale with the growing of nodes in the network (i.e., **scalability**) and (c) its **robustness** against link and nodes failures as WSNs are known to be frequently prone to failures.

1.1. Related work

Distributed approaches [3–6], based on in-network processing, were proposed as interesting alternatives to centralized approaches, which experience poor scalability and robustness against link and node failures. In such approaches, nodes do not need to hold global knowledge about the current network topology, since each node communicates only with its immediate neighbors. The query answer is then successively carried out through local computation from the exchanged data. The advantages of such approaches are numerous: (a) no central base station is required, as every node holds the estimate of the unknown parameter; (b) multi-hop communications are avoided (only direct communications between neighbors are used) and consequently maintaining rooting data is no longer required; (c) better behavior is observed in front of communication and unreliability of nodes.

Flooding [7], for instance, is the first and simplest approach for in-network distributed query processing. In this approach, each sensor node broadcasts its local data (e.g., stored and received data) to its immediate neighbors. The process is repeated until all nodes hold all sensed data. Afterwards, each node can locally compute the query answer. However, this technique requires a significant amount of transmissions and a large storage memory, which wastes the network's resources and decrease its lifetime considerably. Alternatively, iterative approaches [8,9] were proposed to reduce local storage requirements. In fact, in these approaches, each node maintains a local estimate of the query answer (i.e., unknown parameter) that is updated iteratively with weighted data received from immediate neighbors. After several iterations, depending on the application requirements, the process is proven to converge to the right value of the unknown parameter. Again, iterative approaches, while they experience better robustness, require a great deal of communications. Furthermore, because of their synchronized iterations, these approaches generate an important number of packet collision which may decrease the overall network performance, in particular query responsiveness.

Another class of distributed approaches, named *Serial approaches* [10–12], were specifically designed to reduce communication overhead, as previously observed in both centralized and iterative approaches. The basic idea behind these approaches is that one node initiates the query and the answer is successively updated from node to node until all nodes in the network contribute to it; i.e., the query visits all nodes serially. The last visited node holds the right answer of the query [10,11]. As stated in [10], serial algorithms observe significant performances in reducing needed communications in comparison to centralized and iterative approaches. Furthermore, because of their serial nature, these approaches do not require a sophisticated MAC layer since all the communications in the network are serial; i.e., at each step, only one node is required to communicate. Our study indicates that this interesting feature noticeably improves the query response time (see Section 8).

1.2. Motivations

Although theoretical foundations of serial approaches, specifically convergence proofs, have been discussed and provided in [10,11,13], some practical issues, to the best of our knowledge, remain open: in fact, serial approaches require the construction of a Hamiltonian path through the network (we note that previous research works make implicit the assumption of the existence of such a path, which is not true for any network configuration). Furthermore, even when such a Hamiltonian path exists, finding it is known to be a NP-Complete problem [14] which is not easy to meet, particularly in sparse and large-scale networks. The cost of finding such a path, in a *decentralized manner* to ensure scalability, could generate a prohibitive communication cost. To overcome this limitation, the authors in [12] propose to use *space filling curves* to derive a path, that is not necessarily Hamiltonian (i.e. it could visit a node more than once). Nevertheless, the proposed approach does not handle all network topologies. Indeed, it remains suitable for dense and regular network topologies without holes, whereas it is not able to ensure **query completeness** (i.e., visiting all nodes in the network) in any topology. To illustrate, we take the example of Fig. 1.

As provided in [12], we take the same *scan curve* with the orientation from left to right and from top to bottom (see Fig. 1). Hence, all the nodes are "mapped" to this curve according to their locations; i.e., each node has its curve indices. Then, the selection of the next hop is performed according to the curve indices of the unvisited neighbors; i.e., the current node selects among the unvisited neighbors, the one with the next coordinate in the curve order. In the example of Fig. 1, node labeled 3 will select node 9 instead of node 4 because it is far in the curve order. This process is repeated whether there are unvisited neighbors. Similarly, when node 16 holds the query, it selects node 17 than node 21. This process is repeated until reaching a node that has no unvisited neighbors. This is the case of node 20. When the selection of unvisited neighbors is not possible, the authors propose to perform backtracking until a node with unvisited neighbors is reached. In the example, backtracking, illustrated with blue arrows, is stopped at node 16 since its neighbor 21 has not been visited yet. The process can then be continued.

In the example of this network configuration with a hole, all red colored nodes (i.e., nodes labeled 4 to 8) are then left unvisited and consequently do not contribute in the query. Assume that the query is looking for the maximum and one of these left nodes holds this maximum. In this case, the returned result is totally wrong.

We must note that the presence of holes in real WSN deployments is the norm and not the exception due to many environmental facts (bad weather conditions, presence of obstacles, etc.). For this reason, we have taken into account, in the earlier stage of the design of our algorithm, the presence of holes as well as the query completeness requirement. In fact, many realistic applications impose the query completeness. For instance, queries looking for the maximum, the minimum or counting the number of live nodes have to visit all nodes, whatever the network configuration is, otherwise they are not able to provide correct results.

Another inherent limitation of serial approaches is their vulnerability. In fact, as the query is carried out following a path (e.g., serial nature), any cut in this path, particularly at the end of it, will interrupt the query. This risk becomes higher when, on the one hand, the derived path is predefined in advance (as is the case with previous approaches based on space-filling curves), and on the other hand when the network observes several perturbations (e.g. link failures).