Muti-Query Processing Framework for Mobile Sensor Networks

Steven Mudda
Networking Lab
University of Applied Sciences of Southern Switzerland
Email: steven.mudda@supsi.ch

I. Introduction

Sensor networks are being used in different applications like habitat monitoring [1], environment monitoring [2] and volcano monitoring [3]. In majority of sensor network applications a central node (sink) is responsible for collecting all sensors data. Figure 1 shows the network architecture and general query processing framework of these monitoring applications. The sink frequently issues fixed data collection queries [4] [5] to collect all sensor data into a central repository. Users can issue queries to the sink and obtain relevant information from central repository. This network of sensor devices can also be viewed as a database [6] that contains environmental information. The data generated at each sensor node is analogous to data storage and routing of data to sink is analogous to query processing in databases.

Sensor devices typically operate on limited energy supply like batteries, therefore energy efficient data collection is a key design issue for any query processing framework. While querying data from these energy constrained sensor devices, the two major energy consuming components that need to be considered are: data acquisition and communication. The data acquisition cost can be reduced by collecting data only from representative sensors, which are selected by exploiting the spatial and temporal correlations between sensor data [7] [8]. The communication cost can be reduced by performing some in-network aggregation of sensor data, particularly for aggregate queries. Depending on the network topology, a set of aggregator nodes are selected to aggregate data from neighbors and relay it to the sink [9].

TinyDB [10] is a widely known query processing framework specifically designed for low power devices running on TinyOS. In TinyDB a tree-based network topology is maintained with sink as the root node. SNEE [11] is another query processing framework where sink develops an optimal query evaluation plan based on static network topology and available resources.

A. *Drawbacks in existing query processing frameworks*

- Current frameworks are designed to execute only one query and cannot handle multiple queries running simultaneously [4].
- No delay in pushing the data from source to aggregator nodes is an unrealistic assumption.
- The current frameworks are designed to query data only from static sensor nodes.

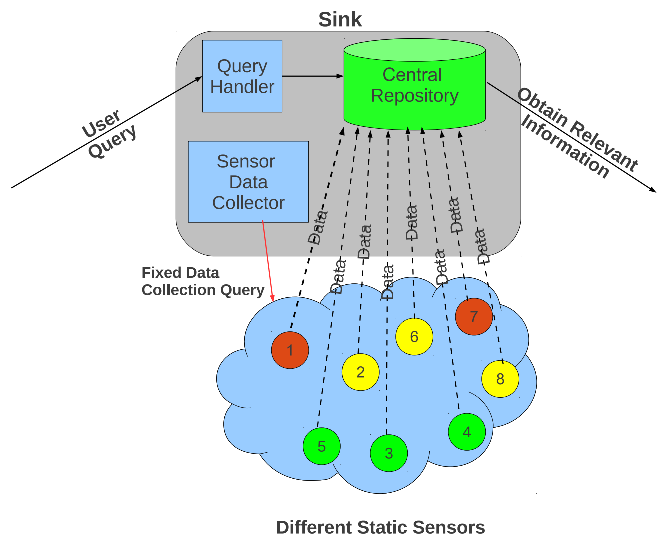

Fig. 1. General Query Processing Framework for Static Sensor Networks

- The inherent network disconnections are not taken into consideration by existing data aggregation methods

In order to address these drawbacks, we propose to develop a new query processing framework where an Intelligent Query Server (IQServer) will handle multiple user queries and frequently monitor a mobile sensor network.

II. Intelligent Query Server

The IQServer will develop an optimal query execution plan by considering current network topology and resources available on each node in the network. Further, the IQServer maintains a historical information of network topology by frequently issuing topology discovery queries into the network. When a query is received by IQS, the network topology with a high probability of occurrence is selected to develop an optimal schedule of execution.

Every query received by our IQServer is processed in three steps

- Find nodes relevant to query.
- Develop a query execution plan.
- Disseminate the query into the network.

A. Find sensor nodes relevant to query

The IQServer will find out two sets of sensor nodes that execute the query

1) Source nodes that contain data relevant to the query. If there exists some spatial and temporal correlations between source nodes then a subset of representative nodes are selected among them.
2) Aggregator nodes that aggregate data from source or representative nodes to reduce the amount of data transmissions.

B. Develop a query execution plan

The IQServer develops an optimal query execution plan considering the most probable network topology and energy constraints of all relevant nodes.

C. Disseminate the query into the network

Once the query execution plan is built the IQServer disseminates the query to all relevant nodes. If a given query can be broken down into a set of sub queries then these sub queries will be sent to corresponding aggregator nodes.

III. IQSERVER ARCHITECTURE

The overall architecture of our proposed IQServer is shown in Figure 2. The IQServer maintains the following two repositories.

1) Sensor Node Information - Used to maintain a list of resources provided by each sensor node and their energy constraints.
2) Network Topology Information - Used to maintain a historical information of network topology.

The Query Handler & Builder component of the IQServer processes all incoming queries and develops a query execution plan depending on the information available in the two repositories. The Query Executor component is responsible to disseminate query to relevant nodes. Further, it handles all sensor data obtained after query execution.

IV. CONCLUSION & FUTURE WORK

This paper intends to address the problem of multiple query processing for mobile sensor networks. We propose an Intelligent Query Server (IQServer) that processes multiple queries and develops a query execution plan considering the availability of resources on each sensor node and historical topological information of the network. We first plan to develop and test our IQServer in a simulation environment followed by implementation on real sensor devices.

V. ACKNOWLEDGEMENT

I would like to thank Prof. Silvia Giordano and Prof. Fabio Crestani for their continuous support and motivation.

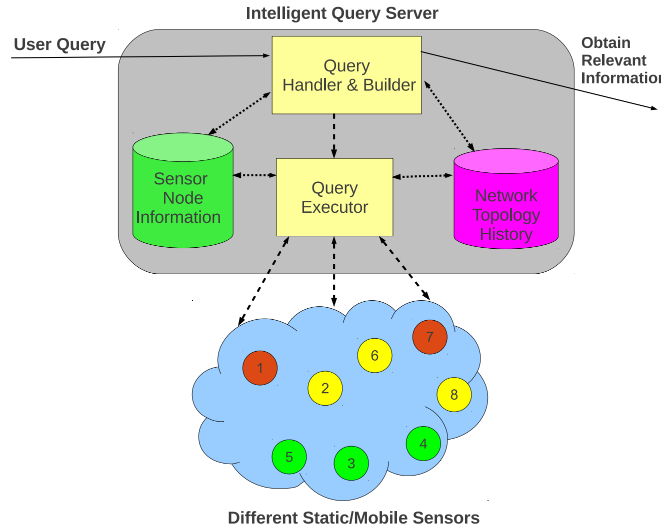

Fig. 2. Architecture of Intelligent Query Server

REFERENCES

[1] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, ser. WSNA '02. New York, NY, USA: ACM, 2002, pp. 88–97.

[2] D. C. Steere, A. Baptista, D. McNamee, C. Pu, and J. Walpole, "Research challenges in environmental observation and forecasting systems," in *Proceedings of the 6th annual international conference on Mobile computing and networking*, ser. MobiCom '00. New York, NY, USA: ACM, 2000, pp. 292–299. [Online]. Available: http://doi.acm.org/10.1145/345910.345961

[3] G. Werner-Allen, K. Lorincz, M. Welsh, O. Marcillo, J. Johnson, M. Ruiz, and J. Lees, "Deploying a wireless sensor network on an active volcano," *IEEE Internet Computing*, vol. 10, no. 2, pp. 18–25, Mar. 2006. [Online]. Available: http://dx.doi.org/10.1109/MIC.2006.26

[4] J. Gehrke and S. Madden, "Query processing in sensor networks," *Pervasive Computing, IEEE*, vol. 3, no. 1, pp. 46 – 55, jan.-march 2004.

[5] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *CIDR*, 2003.

[6] R. Govindan, J. M. Hellerstein, W. Hong, S. Madden, M. Franklin, and S. Shenker, "The sensor network as a database," 2002.

[7] C.-C. Hung, W.-C. Peng, and W.-C. Lee, "Energy-aware set-covering approaches for approximate data collection in wireless sensor networks," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 11, pp. 1993 –2007, nov. 2012.

[8] D. Apiletti, E. Baralis, and T. Cerquitelli, "Energy-saving models for wireless sensor networks," *Knowledge and Information Systems*, vol. 28, pp. 615–644, 2011.

[9] R. Gupta and K. Ramamritham, "Query planning for continuous aggregation queries over a network of data aggregators," *IEEE Trans. on Knowl. and Data Eng.*, vol. 24, no. 6, pp. 1065–1079, Jun. 2012.

[10] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tinydb: an acquisitional query processing system for sensor networks," *ACM Trans. Database Syst.*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[11] I. Galpin, C. Y. Brenninkmeijer, A. J. Gray, F. Jabeen, A. A. Fernandes, and N. W. Paton, "Snee: a query processor for wireless sensor networks," *Distrib. Parallel Databases*, vol. 29, no. 1-2, pp. 31–85, Feb. 2011.