Phenomenon-aware Sensor Database Systems

M. H. Ali

Department of Computer Science, Purdue University
mhali@cs.purdue.edu

Abstract. Recent advances in large-scale sensor-network technologies enable the deployment of a huge number of sensors in the surrounding environment. Sensors do not live in isolation. Instead, close-by sensors experience similar environmental conditions. Hence, close-by sensors may indulge in a correlated behavior and generate a "phenomenon". A phenomenon is characterized by a group of sensors that show "similar" behavior over a period of time. Examples of detectable phenomena include the propagation over time of a pollution cloud or an oil spill region. In this research, we propose a framework to detect and track various forms of phenomena in a sensor field. This framework empowers sensor database systems with phenomenon-awareness capabilities. Phenomenon-aware sensor database systems use high-level knowledge about phenomena in the sensor field to control the acquisition of sensor data and to optimize subsequent user queries. As a vehicle for our research, we build the *Nile-PDT* system, a framework for Phenomenon Detection and Tracking inside *Nile*, a prototype data stream management system developed at Purdue University.

1 Introduction

A large body of research in the database systems area focuses on handling massive amounts of data that is streamed from sensor networks, e.g., see [7, 9, 10, 12, 19, 20, 29]. The main goal is to provide efficient query processing techniques for sensor data. However, emerging sensor-network applications call for new capabilities that are beyond traditional online query processing techniques. Examples of these applications include surveillance [24], object tracking [12], and environmental monitoring [25]. Mainly, these applications go past simple data retrieval to show their evolving interest in data analysis and field understanding.

In this research, we focus on extending sensor database systems with phenomenon-awareness capabilities as a step towards the understanding of sensor data. A phenomenon appears in a sensor field if a group of sensors show "similar" behavior over a period of time. In particular, phenomenon-aware sensor databases (or PhenomenaBases, for short) have two major tasks: First, it detects and tracks various forms of phenomena in space. Second, it utilizes the knowledge about phenomena in the space to optimize subsequent user queries. Although individual sensor readings can be useful by themselves, *phenomenon detection* exploits various notions of correlation among sensor data and provides a global view of the underlying environment. Then, *phenomenon tracking*

monitors the propagation of detected phenomena to reflect the changes in the surrounding environmental conditions. Given the knowledge about phenomena in the surrounding space, phenomenon-aware optimizers bridge the gap between the low-level sensor readings and the high-level understanding of phenomena to answer user queries efficiently.

1.1 Motivation

In this section, we identify five major points through which sensor-network applications benefit from *Phenomenon Detection and Tracking* (*PDT*, for short) techniques. These points can be summarized as follows:

1. **High-level description of the sensor field.** With the aid of *PDT* techniques, an application may ask for "What is going on in a sensor field?" instead of asking "What are the sensor readings?" *PDT* techniques describe the underlying sensor field using a higher level of knowledge (e.g., report a fire alarm instead of a bunch of high temperature readings).
2. **Phenomenon-guided data acquisition.** Data acquisition can be guided by detected phenomena in the sense that we reduce the sampling rate of *non-interesting* sensors (i.e., sensors that do not contribute to any phenomena). Also, we reduce the sampling rate of sensors that are (and will remain) involved in a phenomenon. Such sensors with persistent phenomena are temporarily turned off with the assumption that their phenomena will not disappear instantaneously. Sensors on the *boundaries* of a phenomenon tend to be more interesting and are likely to change their values quickly. We increase the sampling rate of boundary sensors such that we capture the possible change in their state as quickly as possible. Reducing the sampling rate of a sensor will result in a general reduction in the sensor's energy consumed in sampling, processing, and communication. Also, the processing load over the centralized DSMS (or the sink node of the sensor network) will be reduced.
3. **Data compression.** Voluminous sensor data can be compressed using *PDT* techniques. Instead of maintaining the readings of each individual sensor, we maintain phenomenon pairs (R, B), where R is the region that bounds a phenomenon with behavior B.
4. **Prediction.** Tracking a phenomenon movement and predicting its future trajectory foresees the next state of the sensor field. Based on the boundaries of a phenomenon and their trajectories, we can predict the movement of various phenomena in the space. Prediction of phenomenon-movement enables us to decide which sensors to turn on and off in order to conserve energy without losing useful information.
5. **Phenomenon-guided query processing.** Given a query and given a set of phenomena, query processing can be guided to regions with phenomena that satisfy the query predicates. Hence, the query space is reduced. All phenomena in the space are maintained and their contributing sensors are indexed. Then, a user query is mapped to a set of system-detected phenomena. Regions that are covered by this set of phenomena are processed to answer the query.

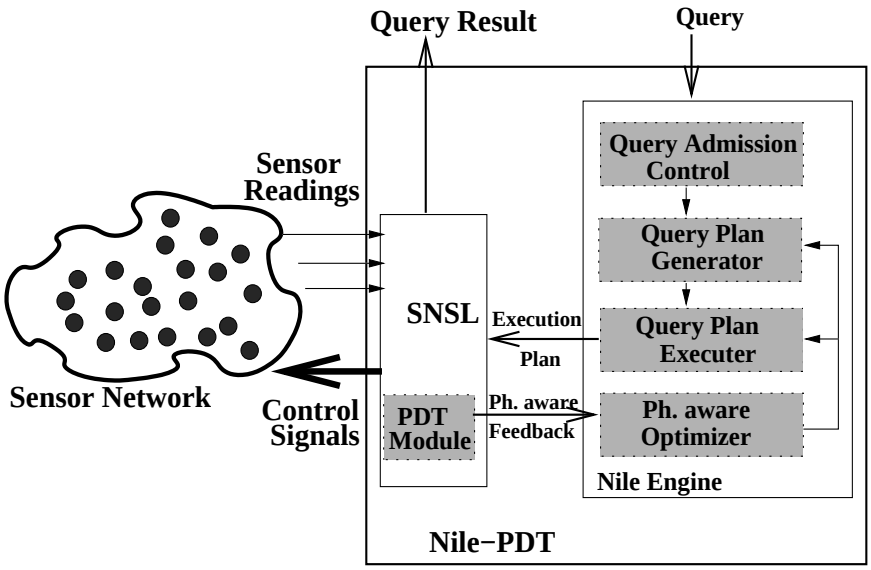

Fig. 1. Nile-PDT architecture.

1.2 Applications

Several applications benefit from the detection and tracking of various phenomena in the sensor field. Examples of these applications include:

1. Tracing pollutants in the environment, e.g., oil spills, or gas leakage.
2. Reporting the excessive purchase of an item at different branches of a retail store.
3. Detecting computer worms that strike various computer sub-networks.

Notice that a phenomenon may or may not have spatial properties. The phenomenon in the first example has spatial properties, where an oil spill is a contiguous portion of the ocean surface. If a phenomenon has spatial properties, it is referred to by the term *cloud*. Retail store applications may not have the notion of spatial attributes, where retail stores can be spread arbitrary over the region. In the third application, the notion of spatial distance is relative to the network connectivity. Also, to generalize the concept of phenomena, a sensor may be a physical device that acquires readings from the environment, (e.g., temperature, light, humidity, or substance identifiers as in the first example) or a virtual sensor like the cashier machine that reads item identifiers as in the second example. A sensor may even be a piece of software that detects computer worms as in the third example.

This Ph.D. research is done in the context of *Nile* [14], a prototype data stream management system developed at Purdue University. We extend *Nile* with a *Sensor Network Support Layer* (*SNSL*) where a Phenomenon Detection

and Tracking (*PDT*) module is placed. *PDT* monitors phenomena as they propagate in the sensor field and returns feedback to a phenomenon-aware query optimizer which, in turn, controls the generation and execution of query plans. Figure 1 illustrates the architecture of *Nile-PDT*.

2 Research Plan

In this section, we identify the major challenges in the design and the implementation of phenomenon-aware systems. In particular, we address the following challenges:

1. **The phenomenon-extraction challenge,** where we separate sensors that participate in a phenomenon from sensors that participate in no phenomena.
2. **The sensor-network processing requirements challenge,** where we develop algorithms that comply with the requirements of large-scale dynamically-configured sensor-networks with distributed processing capabilities.
3. **The similarity-notion challenge,** where we define various notions of similarity among sensors' behavior and develop techniques that comply with these notions.
4. **The phenomenon-interpretation challenge,** where we develop a query optimizer that makes use of knowledge about detected phenomena to answer user queries.

In the remainder of this section, we devote a subsection to explore each challenge, emphasize its associated research tasks, and propose preliminary ideas.

2.1 The phenomenon-extraction challenge

As a first step towards phenomenon detection, we propose a concrete definition of a phenomenon. Two parameters control the *phenomenon* definition, the *strength* (α) and the *time span* (w). The *strength* of a phenomenon indicates that a certain phenomenon should occur at least α times to qualify as a phenomenon. (This measure is similar to the notion of support in mining association rules, e.g., see [1].) Reading a value less than α times is considered noise, e.g., impurities that affect the sensor readings. The time span w limits how far a sensor can be lagging in reporting a phenomenon. w can be viewed as a time-tolerant parameter, given the common delays in a sensor network. (This measure is similar to the notion of gaps in mining generalized sequential patterns [23].) In the light of these two parameters, a phenomenon can be defined as follows:

Definition 1. *In a sensor network SN, a phenomenon P takes place only when a set of sensors $S \subset SN$ report similar reading values more than α times within a time window w.*

In [5], we simplify the definition by considering the discrete case of the phenomenon where the notion of similarity reduces to equality. (In Section 2.3, we consider other notions of similarity.) The process of *phenomenon detection* and *tracking* (*PDT*) is divided into three phases:

1. The *joining* that applies an in-memory *multi-way* join over the entire sensor network to detect sensors with the same value within a time frame of length w from each other.
2. The *candidate selection* phase that enforces the (α) and (w) constraints on join pairs to report phenomenon candidate members.
3. The *grouping/output* phase that groups phenomenon candidate members and investigates the application semantics to form and report phenomena to the user.

2.2 The sensor-network processing requirements challenge

To implement a phenomenon-aware sensor database system, we need to shift the phenomenon detection phases to the sensor-network level. Distributed algorithms need to replace the centralized ones. We address five major challenges in sensor network processing:

1. **Scalability,** to face the excessive deployment of sensors in the space.
2. **Adaptivity,** to handle the gradual/abrupt appearance/disappearance of phenomena.
3. **Distributed processing,** to relieve the centralized system from possible congestions and to reduce the communication cost by filtering irrelevant data as early as possible.
4. **Dynamic configuration,** where sensors may be added or removed from the network based on the network conditions, the sensor's lifetime, and the availability of additional sensors.
5. **Limited energy,** to reduce the frequency of battery replacement in environments where the existence of a human being is either tough or dangerous, e.g., habitat monitoring [25].

To address the above challenges, we place a new operator (the *SNJoin* operator [3]) at the core of the *PDT* module. *SNJoin* is a distributed multi-way join operator that is specially designed for large-scale dynamically-configured sensor networks.

2.3 The similarity-notion challenge

In this section, we generalize our work to include continuous phenomena. Continuous phenomena are generated by sensors whose values are drawn from continuous ranges. The major challenge in continuous phenomena comes from the fact that similarity among sensors' behavior does not necessarily mean equality. Instead, various notions of similarity need to be explored. We plan to examine the following notions of similarity:

1. **Similar values,** where similarity is assessed based on a distance function "dist". Two values v_1 and v_2 are considered similar if $dist(v_1, v_2) < D$.
2. **Similar behavior,** where we extract summaries from the sensor data (e.g., histograms, count sketches, or user-defined summaries) that capture the sensors' behavior over a window of time. Similarity is assessed based on the distance between the summaries.
3. **Similar trend,** where the increase/decrease in one sensor readings implies the increase/decrease of another sensor's readings. Generally, the change in the readings of one sensor is related to the change in the other sensor's readings by a function f (i.e., $\triangle v_1 = f(\triangle v_2)$). For example, the increase in the readings of smoke detectors is usually accompanied by an increase in the readings of temperature sensors.

We investigate two approaches to handle continuous phenomena: First, as a preprocessing phase, we group sensor readings into clusters, represent each reading by its cluster identifier, and apply discrete PDT techniques over cluster identifiers. Second, we replace the equality join by a similarity join, where the similarity function is a user-defined function that is provided as part of the query syntax. Initial implementation of PDT using similarity join is conducted in [2].

2.4 The phenomenon-interpretation challenge

A phenomenon-aware system implies that the phenomenon detection and tracking process is always running in the background to detect new phenomena and to track the propagation of already-detected phenomena. Based on the understanding of surrounding phenomena, phenomenon-aware systems answer user queries. The ultimate goal of our research is to build a sensor database system that optimizes user queries on a *"phenomenon detection guides query processing"* basis. We view phenomenon-aware query optimization as a rich area of research where phenomenon understanding alters the construction/execution of query plans. We explore phenomenon-aware query optimization along two directions:

1. Increasing the sampling rate of sensors that contribute to phenomena associated with active queries.
2. Controlling the join probing sequence such that a reading coming from one sensor probes only sensors where a match is likely to be found. The join probing sequence is tuned to favor the joins that affect the appearance or the disappearance of a phenomenon.

3 Experiments

As a proof of concept, we conduct an experimental study to show the performance gains a phenomenon-aware optimizer may achieve. We generate a set of 2000 sensors using the Nile-PDT simulator [2]. Each sensor generates a stream of

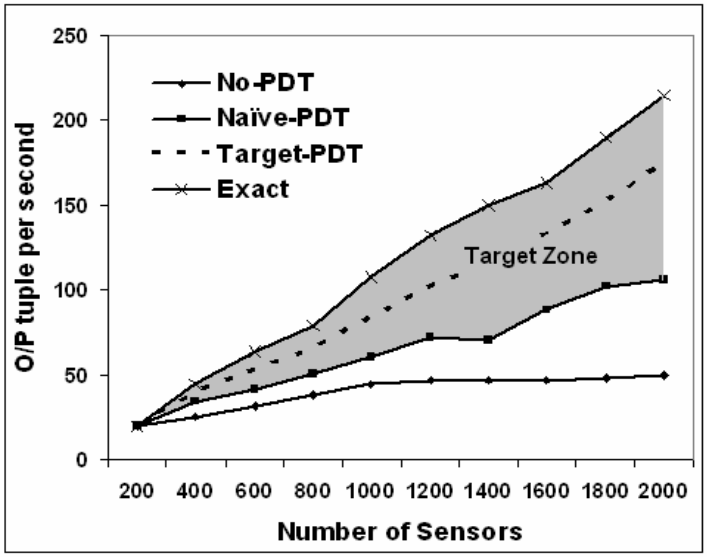

Fig. 2. Performance of Nile-PDT.

readings at a rate of 1 reading per second. We detect and track discrete phenomena as discussed in [5]. Detected phenomena are fed into a naive phenomenon-aware query optimizer. The naive optimizer searches the list of detected phenomena, determines interesting phenomenon regions (i.e., phenomenon regions that satisfy the query predicates), and deploys the query only over regions of its interesting phenomena. We process a set of 100 selection query with predicate selectivities that range from 1% to 20% of the whole space.

Two experiments are conducted. First, we measure the energy saved (in terms of the number of transmitted messages) using a phenomenon-guided data acquisition. Sensors that contribute to phenomena are sampled more frequently than sensors that contribute to no phenomena. The number of transmitted sensor readings is reduced by up to 65%. Second, we measure the accuracy of the query result in terms of the average number of output tuples. Figure 2 illustrates the performance of the query processor in the following cases:

1. **No-PDT,** where no PDT-capabilities are used.
2. **Exact,** where infinite resources are given to calculate the answer.
3. **Naive-PDT,** where a naive PDT phenomenon-aware optimizer is used.
4. **Target-PDT**, which is an imaginary curve that reflects where we expect the performance of a non-naive optimizer will fall.

The Target-PDT curve is somewhere between the *naive-PDT* and the *Exact* curves (i.e., the target zone in the figure).

4 Related Work

Sensors are devices that are capable of sampling, processing, and transmitting readings from the surrounding environment. Various techniques have been proposed to handle the *sampling* [4, 6, 10, 20], *processing* [9, 29], and *transmission* [8, 16, 18] tasks. In this section, we overview other techniques that analyze sensor data to track objects and/or regions as they move in the space. We also highlight the join operation over data streams due to its important role in the process of phenomenon detection and tracking.

To reduce the overall power consumption of the sensor network while object tracking, [28] proposes a prediction-based strategy that focuses on regions where the moving object is likely to appear. In [30], the tree-like communication structure of the sensor network is reconfigured dynamically to reduce the number of hops between a sensor and the sink node as the sensor comes closer to the moving object. Instead of tracking a single object, [2, 5] provide a framework to detect and track phenomena in a sensor field once a region of sensors exhibit a common behavior. The work in [22] investigates how to detect boundaries that separate homogeneous regions. In [15], continuous regions with similar values are grouped into homogeneous regions called *isobars*.

The join operation detects similarities in value among sensors along the trajectory of a moving object or among sensors in the same phenomenon region. For example, [12] tracks moving objects in a sensor field through a window join algorithm (*w-join*). The join operation has been studied thoroughly in the literature, e.g., [11–13]. Symmetric Hash Join [27] is the first algorithm that takes care of the infiniteness of the data source. XJoin [26] provides disk management to store overflowing tuples on disk for later processing. An asymmetric window join over two data streams with different arrival rates is discussed in [17]. The Hash-Merge Join (*HMJ*) [21] is a recent non-blocking join algorithm that produces early join results. In our work, we propose the *SNJoin* operator [3], a multi-way join operator that is specially designed for large-scale dynamically-configured sensor networks.

5 Conclusions

In this paper, we proposed a framework for phenomenon-aware sensor database systems. We provided a concrete definition for the phenomenon and explored various notions of similarity among sensors' behavior. In a phenomenon-aware sensor database system, the knowledge gained through detected phenomena guides query processing to regions of interest in the sensor field. The proposed research plan has four phases. The first phase is concerned with detecting and tracking discrete phenomena (i.e., the notion of similarity reduces to equality) in a centralized data stream management system. The second phase pushes the detection and tracking of phenomena to the sensor-network level in a distributed-processing fashion. The third phase addresses various notions of similarity among sensors' behavior and generalizes the phenomenon concept to include continuous

phenomena. The fourth phase achieves, through a phenomenon-aware optimizer, the ultimate goal of answering user queries efficiently based on the knowledge about phenomena in the space.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of VLDB*, 1994.
2. M. H. Ali, W. G. Aref, R. Bose, A. K. Elmagarmid, A. Helal, I. Kamel, and M. F. Mokbel. Nile-pdt: A phenomena detection and tracking framework for data stream management systems. In *Proc. of VLDB*, 2005.
3. M. H. Ali, W. G. Aref, and I. Kamel. Multi-way joins for sensor-network databases. Technical Report CSD-05-21, Department of Computer Science, Purdue University, 2005.
4. M. H. Ali, W. G. Aref, and C. Nita-Rotaru. Spass: Scalable and energy-efficient data acquisition in sensor databases. In *Proc. of the International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE)*, 2005.
5. M. H. Ali, M. F. Mokbel, W. G. Aref, and I. Kamel. Detection and tracking of discrete phenomena in sensor-network databases. In *Proc. of SSDBM*, 2005.
6. B. Babcoc, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. of the Annual ACM-SIAM Symp. on Discrete Algorithms*, 2002.
7. P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In *Proc. of MDM*, 2001.
8. A. Cerpa and D. Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proc. of INFOCOM*, 2002.
9. J. Considine, F. Li, G. Kollios, and J. W. Byers. Approximate aggregation techniques for sensor databases. In *Proc. of ICDE*, 2004.
10. A. Deshpande, C. Guestrin, S. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *Proc. of VLDB*, 2004.
11. L. Golab and M. T. Ozsu. Processing sliding window multi-joins in continuous queries over data streams. In *Proc. of VLDB*, 2003.
12. M. A. Hammad, W. G. Aref, and A. K. Elmagarmid. Stream window join: Tracking moving objects in sensor-network databases. In *Proc. of SSDBM*, 2003.
13. M. A. Hammad, M. Franklin, W. G. Aref, and A. K. Elmagarmid. Scheduling for shared window joins over data streams. In *Proc. of VLDB*, 2003.
14. M. A. Hammad, M. F. Mokbel, M. H. Ali, W. G. Aref, A. C. Catlin, A. K. Elmagarmid, M. Eltabakh, M. G. Elfeky, T. Ghanem, R. Gwadera, I. F. Ilyas, M. Marzouk, and X. Xiong. Nile: A query processing engine for data streams. In *Proc. of ICDE*, 2004.
15. J. M. Hellerstein, W. Hong, S. Madden, and K. Stanek. Beyond average: Toward sophisticated sensing with queries. In *Proc. of IPSN*, 2003.
16. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of MOBICOM*, 2000.
17. J. Kang, J. F. Naughton, and S. D. Viglas. Evaluating window joins over unbounded streams. In *Proc. of ICDE*, 2003.
18. J. Kulik, W. R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *ACM Wireless Networks*, 8(2-3):169–185, 2002.

19. S. Madden and M. Franklin. Fjording the stream: An architecture for queries over streaming sensor data. In *Proc. of ICDE*, 2002.
20. S. Madden, M. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *Proc. of SIGMOD*, 2003.
21. M. Mokbel, M. Lu, and W. Aref. Hash-merge join: A non-blocking join algorithm for producing fast and early join results. In *Proc. of ICDE*, 2004.
22. R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *Proc. of IPSN*, 2003.
23. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of EDBT*, 1996.
24. S. Srinivasan, H. Latchman, J. Shea, T. Wong, and J. McNair. Airborne traffic surveillance systems: video surveillance of highway traffic. In *the 2nd ACM international workshop on Video surveillance & sensor networks*, 2004.
25. R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin. Habitat monitoring with sensor networks. *Communications of ACM*, 47(6):34–40, 2004.
26. T. Urhan and M. Franklin. Xjoin: A reactively-scheduled pipelined join operator. *IEEE Data Eng. Bull.*, 23(2):27–33, 2000.
27. A. N. Wilschut and E. M. G. Apers. Pipelining in query execution. In *Proc. of the International Conference on Databases, Parallel Architectures and their Applications*, 1991.
28. Y. Xu, J. Winter, and W.-C. Lee. Prediction-based strategies for energy saving in object tracking sensor networks. In *Proc. of MDM*, 2004.
29. Y. Yao and J. Gehrke. Query processing in sensor networks. In *Proc. of CIDR*, 2003.
30. W. Zhang and G. Cao. Optimizing tree reconfiguration for mobile target tracking in sensor networks. In *Proc. of INFOCOM*, 2004.