Tracking Dynamics Using Sensor Networks: Some Recurring Themes

Krithi Ramamritham

Dept of Computer Science & Engineering
Indian Institute of Technology Bombay

Abstract. Much of the data consumed today is dynamic, typically gathered from distributed sources including sensors, and used in real-time monitoring and decision making applications. Large scale sensor networks are being deployed for applications such as detecting leakage of hazardous material, tracking forest fires or environmental monitoring. Many of these "natural" phenomena require estimation of their future states, based on the observed dynamics. Strategically deployed sensors can operate unattended (minimizing risk to human life) and provide the ability to continuously monitor the phenomena and help respond to the changes in a timely manner. In this paper, we show that in-network aggregation, in-network prediction, and asynchronous information dissemination form sound building blocks for addressing the challenges in developing low overhead solutions to monitor changes without requiring prior knowledge about the (dynamics of) the phenomena being monitored.

1 Introduction

More and more of the information we consume is dynamic and comes from sensors. Networks made up of sensor nodes are being increasingly deployed for observing continuously changing data. Numerous interconnected sensors are being used to build

- smart buildings that make efficient use of energy,
- smart vehicles that improve efficiency, assist drivers and help in navigation,
- robots to monitor water bodies, for better pollution control, and
- systems to track weather patterns.

These *cyber-physical systems* blend sensing, actuation, computation, networking, and physical processes. In these systems, there is a need to process enormous amounts of changing, time-sensitive data and to continuously update query results (energy) efficiently, adaptably, i.e., continuously evolve to changing conditions, and resiliently – to message/sensor losses.

Continuous queries are appropriate for such monitoring applications since a user is updated as and when source data changes. But this continuous updating is a challenging requirement because sensing devices are often battery powered, making energy efficiency in processing sensor data and updating users

V. Garg, R. Wattenhofer, and K. Kothapalli (Eds.): ICDCN 2009, LNCS 5408, pp. 1–7, 2009.

a primary design consideration; since energy expended for communication is significantly higher than that for local computations, data dissemination and query-processing techniques should minimize the amount of communication so as to increase the lifetime of the network.

This paper outlines some simple techniques to address these challenges in the context of aggregate query processing. We show that in-network aggregation, in-network prediction, and asynchronous information dissemination form sound building blocks for addressing the challenges in developing low overhead solutions to track changes without requiring prior knowledge about the (dynamics of) the phenomena.

2 Data Dissemination in Sensor Networks

A canonical sensor network application involves sensors wirelessly interacting through multi-hop protocols with their neighbours. Queries are posed typically at a base station and hence results are also expected to be produced at the base station. A very simple model for processing sensor data is to push all data to the base station and compute at base station. This simple technique floods the data through the network. It strains sensor nodes near the base, draining their batteries and disconnecting network. An alternative is to pull only the relevant data to the base station and compute at the base station. Here the query needs to be flooded. But, it begs the question: "which of the (new) sensor data is relevant to a given query?".

Only new data that is required to meet user specified accuracy or coherency requirements, e.g., 2 degree temperature, 10m distance, etc. need be deemed to be relevant. This exploitation of user specified coherency requirements reduces the number of updates in sensor networks.

This implies that we require a query to request the value of an aggregate with an associated coherency, c. This denotes the accuracy of the results delivered to the query node relative to that at the sources, and thus, constitutes the user-specified requirement. For example, a query injected for building monitoring is: "Report the average temperature of the southern wall of the building whenever it changes by more than 2 degrees". Thus, any change in the average temperature value that is within two degrees of what the query node knows need not be reported and the current value known to the query node is considered accurate enough. The resulting reduction in the number of update messages improves both fidelity of query results (by reducing the probability of message losses due to collisions) and lifetime (by reducing communication-related energy needs).

Essentially, we relax strong coherency by adopting the notion of Δ_v - coherency: The difference in the data values at the source and the base bounded is by Δ_v at all times. For example, if we are only interested in temperature changes larger than 2 degrees, $\Delta_v = 2$.

3 Routing Relevant Data to Base

Typically, an overlay network is constructed to route messages from a set of source nodes to the base station. Many ways to realize the overlay tree have been proposed in the literature. Even those which incur nontrivial overheads can be justified since the overlay tree construction costs can be amortized over the monitoring period, that is the period over which the continuous query will execute. Tree construction, i.e., choosing the node to which a node with an update sends the update can depend on how many hops away it is from the source, the length of expected time that the node can serve, etc.

Also, to ensure that single node failures do not lead to failure of the network as a whole, there is a need to adjust the overlay tree after such failures. In [3] we propose a tree construction algorithm that has the following features:

1. It does **take into account the coherency requirements associated with the query, the remaining energy at the sensors, and the communication and the message processing delays**, thereby contributing to higher lifetime and fidelity of query results.
2. It is able to exploit the presence of common sources across multiple queries. This leads to further increase in fidelity and lifetime.
3. It incorporates optimizations to efficiently handle complex aggregate queries with `group by` clauses.
4. Upon the death of a node, the dissemination tree can be locally adjusted allowing query results to be provided. This increases lifetime.

4 Asynchronous In-Network Aggregation and Prediction

For many types of aggregate queries, the number of message transmissions can be reduced significantly by computing partial aggregates wherever possible while the messages are being routed towards the query node. Overlay network allows in-network partial aggregation. For example, consider computing the max of a set of sensed values. When each node in the overlay tree hears from its children, it can compute the max of the received values and send it to its parent. This technique called in-network aggregation has been exploited to increase the lifetime of the sensor network. The nodes at which this is done are called aggregator nodes. Existing approaches to answering coherency based aggregate queries perform in-network aggregation by synchronizing transmissions of nodes level-by-level on an aggregation tree [5]. Any message that is received by an aggregator node is delayed for a certain amount of time before it can be propagated up the tree. This leads to a definite loss in fidelity. Moreover, these approaches do not address the issues of energy efficiency and timeliness of query results in their tree construction mechanisms. In [3] we present an asynchronous prediction-based approach for answering aggregate queries. In it, each node in the tree computes a partial aggregate of the values sensed by the source nodes which belong to the subtree rooted at that node. The computed value of the partial aggregate

is pushed by a node to its parent node in the aggregation tree. It incorporates several novel ingredients:

It makes use of asynchronous in-network aggregation wherein an aggregator node computes a partial aggregate *asynchronously*, i.e., whenever an update that may affect the current partial aggregate is received from one of its serving nodes in the aggregation tree. Thus, every received message that is required to be sent to the query node should be pushed up the tree as soon as possible, i.e., **change in values sensed at the sources must be propagated to the query node as soon as possible**. In contrast, existing approaches compute aggregates *synchronously*, often delaying propagation of the effect of received partial aggregates.

Suppose node B receives a partial aggregate from node E. In order to compute a new partial aggregate, B needs the current value of the partial aggregate computed by node F. Since this value is unknown to B, what value should B use for the current value at node F? When an aggregator node receives a partial aggregate from a serving node and computes a new partial aggregate, what values should it assume for all other serving nodes? In our approach, **the aggregator node predicts the missing values from the previously received values using a computationally efficient prediction mechanism**. This *prediction-based* approach is in contrast to existing *last-value-based* approaches that use the last received values for this purpose.

If each partial aggregate computed as above were to be disseminated towards the source, would it not lead to significant energy consumption? This is where the idea of *in-network prediction* proves to be useful again. When an aggregator node computes a partial aggregate asynchronously, it also calculates the value of the partial aggregate as would be predicted by the receiving node. **If the difference between the previous value and the new value is within a specified fraction of the coherency associated with the partial aggregate (derived from user specified coherency on the aggregate), it does not send the computed value**, thus saving energy in transmissions.

The latter idea of *Prediction based In-network Filtering* implies that a child node F need not send a new partial aggregate to B if (a) if it is not *too different* from the previously sent value, or (b) B's new prediction will not be *too different* from prediction based on previous values received by B from F.

Let us make the notion of *too different* somewhat more concrete. Let us denote by c, the user specified coherency on the aggregate query and by c', the coherency associated with the partial aggregate. c and c' are related as:

$$c' = c \times \beta$$

where β is a real number in the interval $[0, 1]$. To understand β, consider an aggregation tree. In an aggregation tree, the number of messages exchanged between nodes increases as we traverse from the leaf nodes in the tree towards the root. Due to higher number of messages near the source nodes, the possibility of messages being lost due to collisions is higher. We use coherency c to suppress the number of messages exchanged between nodes. β effectively controls the number of messages suppressed and ensured that the query node receives results at the desired accuracy.

In order to ensure accurate prediction of partial aggregates, we use the notion of *NoActivityThreshold* to guard against loss of messages (and thus loss of model parameters and partial aggregate values) due to collisions. *NoActivityThreshold* at a node for its dependent node is defined as the amount of time for which a node waits before pushing the value of its partial aggregate to the dependent, i.e. if a node has not pushed the value of the partial aggregate to its dependent for a duration greater than *NoActivityThreshold*, the node pushes the value of the partial aggregate.

We would like to point out that our asynchronous approach is in direct contrast with the epoch-based synchronization schemes, for example, TAG [4] and TiNA [5]. In that approach each epoch is divided into time slots and all serving nodes at a given level in the tree are allowed to transmit within a particular time slot. The dependents of these nodes listen during this time slot. At the end of this time slot, each dependent computes a partial aggregate of the data received from its serving nodes. During the next time slot, the dependents transmit the partial aggregates to their dependents. In synchronous computation methods, the duration of the time slot for which the message is withheld at each aggregator node decides the amount of delay for each value that is generated at a source to reach the query node. This delay can lead to loss in fidelity, unacceptable for scenarios that require online decision making. Using an asynchronous approach minimizes this delay, thus providing the potential to deliver higher fidelity. In terms of lifetime, with the asynchronous approach it may seem that computation of an aggregate on receiving a message from any serving node, and a subsequent push to the dependent, if required, may lead to unnecessary transmissions and thus a decrease in lifetime. Our approach of using in-network filtering and in-network prediction for energy efficient aggregation ensures that this is not the case.

In [3], we show that for simple aggregates, for improved query lifetime and correctness, overlay tree construction algorithm should be (remaining) energy-aware, and that we should exploit in-network processing ability. The latter translates to exploiting coherency requirement, using asynchronous prediction based aggregation, and using prediction based In-network filtering. Experimental results demonstrate that (a) the resulting scheme has only one fifteenth of the fidelity loss along with a 40% improvement in lifetime compared to a synchronous last-value-based aggregate computation method; (b) simple approaches to predicting partial aggregates work well for real-world phenomena.

5 Aggregations While Tracking Dynamic Natural Phenomena

As was mentioned in the introduction, time critical data sensed in-situ or remotely – from many mobile/stationary nodes have to be continuously aggregated to track phenomena in the physical world, e.g., movement of oil slicks, gas plumes, etc. Here we briefly show that asynchronous in-network filtering, prediction & aggregation are themes that are useful even in more complex aggregation scenarios.

Consider tracking a dynamic boundary. A dynamic boundary has mainly two types of variations: spatial and temporal. So, the effective tracking of dynamic boundaries requires handling both of these variations. In [2] we describe an algorithm for dynamic boundary tracking which combines a spatial estimation technique and a temporal estimation technique to effectively track a dynamic boundary using static range sensors that measure the distance to the boundary from their current location. The first step is to estimate the boundary at a location x using a spatial estimation technique. It uses spatial correlations among (error-prone sensor) observations at a given time by sensors within a small neighbourhood of x. Aggregator nodes in the overlay network perform aggregation operations on sensor observations to estimate a number of boundary points. Partial information of the boundary from aggregator nodes is then sent to the base station where the final estimate of the boundary at x is computed. In order to exploit in-network aggregation possibility, the aggregation is done using kernel smoothing that is amenable to being "broken-up" into subcomputations that can be done within the network, by the aggregator nodes. A similar approach is used to perform in-network subcomputations needed to determine the confidence interval associated with the boundary estimated at x. A confidence band is estimated from multiple boundary points around the entire boundary using an interpolation scheme executed by the base station.

The second component of the overall approach is a temporal estimation technique which ensures that the estimates are updated whenever due to changes in the boundary the confidence band does not cover the boundary with a desired accuracy. We use a Kalman Filter based technique to predict future boundary locations based on its model of the boundary dynamics. Once the boundary has moved by more than a certain threshold, the spatial estimation technique is invoked to get an accurate estimate of the boundary. As a result, boundary estimates are updated based on only the local dynamics of the boundary and partial estimates track changes in sections of the boundary. Both of these lead to reduction in communication overhead for accurate boundary estimation. Effectiveness with respect to tracking efficiency and correctness have been demonstrated on real contours [2].

Our current work involves applying the building blocks discussed in this paper for handling aggregations of sensor observations when using mobile in-situ sensors [1].

6 Conclusions

Given a sensor network and aggregate queries over the values sensed by subsets of nodes in the network, how do we ensure that *high quality results* are served for the *maximum possible time*? The issues underlying this question relate to the *fidelity* of query results and *lifetime* of the network. To maximize both, we propose a novel technique called *asynchronous in-network prediction* incorporating two computationally efficient methods for in-network prediction of partial aggregate values. These values are propagated via a tree whose construction is cognizant

of (a) the coherency requirements associated with the queries, (b) the remaining energy at the sensors, and (c) the communication and message processing delays. Finally, we exploit *in-network filtering* and *in-network aggregation* to reduce the energy consumption of the nodes in the network. Experimental results over real world data used to track dynamic physical phenomena support our claim that for aggregate queries with associated coherency requirements, a prediction based asynchronous scheme provides higher quality results for a longer amount of time than a synchronous scheme.

References

1. Srinivasan, S., Ramamritham, K., Kulkarni, P.: ACE in the Hole: Adaptive Contour Estimation Using Collaborating Mobile Sensors. In: IPSN: ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN 2008) (April 2008)
2. Duttagupta, S., Ramamritham, K., Kulkarni, P., Moudgalya, K.: Tracking Dynamic Boundary Fronts using Range Sensors. In: Verdone, R. (ed.) EWSN 2008. LNCS, vol. 4913, pp. 125–140. Springer, Heidelberg (2008)
3. Edara, P., Limaye, A., Ramamritham, K.: Asynchronous In-network Prediction: Efficient Aggregation in Sensor Networks. ACM Transactions on Sensor Networks (November 2008)
4. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a Tiny AGgre- gation service for ad-hoc sensor networks. SIGOPS Oper. Syst. Rev. 36 (2002)
5. Sharaf, M.A., Beaver, J., Labrinidis, A., Chrysanthis, P.K.: TiNA: A Scheme for Temporal Coherency-Aware in-Network Aggregation. In: Third International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE) (2003)