In-network Data Estimation for Sensor-driven Scientific Applications [⋆]

Nanyan Jiang and Manish Parashar

Center for Autonomic Computing (CAC) &
The Applied Software Systems Laboratory (TASSL)
Department of Electrical and Computer Engineering
Rutgers University, Piscataway NJ 08855, USA
{nanyanj,parashar}@rutgers.edu

Abstract. Sensor networks employed by scientific applications often need to support localized collaboration of sensor nodes to perform in-network data processing. This includes new quantitative synthesis and hypothesis testing in near real time, as data streaming from distributed instruments, to transform raw data into high level domain-dependent information. This paper investigates in-network data processing mechanisms with dynamic data requirements in resource constrained heterogeneous sensor networks. Particularly, we explore how the temporal and spatial correlation of sensor measurements can be used to trade off between the complexity of coordination among sensor clusters and the savings that result from having fewer sensors involved in in-network processing, while maintaining an acceptable error threshold. Experimental results show that the proposed in-network mechanisms can facilitate the efficient usage of resources and satisfy data requirement in the presence of dynamics and uncertainty.

Key words: Sensor system programming, In-network data estimation, Sensor-driven scientific applications.

1 Introduction

Technical advances in sensing technologies are rapidly leading to a revolution in the type and level of instrumentation of natural and engineered systems, and is resulting in pervasive computational ecosystems that integrate computational systems with these physical systems through sensors and actuators. This in turn is enabling a new paradigm for monitoring, understanding, and managing natural and engineered systems – one that is information/data-driven and that

[⋆] The research presented in this paper is supported in part by National Science Foundation via grants numbers CNS 0723594, IIP 0758566, IIP 0733988, CNS 0305495, CNS 0426354, IIS 0430826 and ANI 0335244, and by Department of Energy via the grant number DE-FG02-06ER54857, and was conducted as part of the NSF Center for Autonomic Computing at Rutgers University.

opportunistically combines computations and real-time information to model, manage, control, adapt, and optimize.

Several scientific and engineering application domains, such as waste management [1], volcano monitoring [2], city-wide structural monitoring [3], and end-to-end soil monitoring system [4], are already experiencing this revolution in instrumentation. This instrumentation can also potentially support new paradigms for scientific investigations by enabling new levels of monitoring, understanding and near real-time control of these systems. However, enabling sensor-based dynamic data-driven applications presents several challenges, primarily due to the data volume and rates, the uncertainty in this data, and the need to characterize and manage this uncertainty. Furthermore, the required data needs to be assimilated and transported (often from remote sites over low bandwidth wide area networks) in near real-time so that it can be effectively integrated with computational models and analysis systems. As a result, data in most existing instrumented systems is used in a post-processing manner, where data acquisition is a separate offline process.

The overall goal of this research is to develop sensor system middleware and programming support that will enable distributed networks of sensors to function, not only as passive measurement devices, but as intelligent data processing instruments, capable of data quality assurance, statistical synthesis and hypotheses testing, as they stream data from the physical environment to the computational world [5]. This paper specifically investigates abstractions and mechanisms for in-network data processing that can effectively satisfy dynamic data requirements and quality of data and service constraints, as well as investigate tradeoffs between data quality, resource consumptions and performance. In this paper, we first present the *iZone* programming abstractions for implementing in-network data estimation mechanisms. We then explore optimizations that can use the spatial and temporal correlation in sensor measurements to reduce estimation costs and handle sensor dynamics, while bounding estimation errors. For example, an appropriate subset of sensors might be sufficient to satisfy the desired error bounds, while reducing the energy consumed. The optimized in-network data estimation mechanisms are evaluated using a simulator. The evaluations show that these mechanisms can enable more efficient usage of the constrained sensor resources while satisfying the applications requirements for data quality, in spite of sensor dynamics.

The rest of the paper is organized as follows. Section 2 describes the *iZone* programming abstraction and in-network data estimation mechanisms. Section 3 presents space, time and resource aware optimizations for in-network interpolation. Section 4 presents an experimental evaluation. Related work is discussed in Section 5. Finally, Section 6 presents conclusions.

2 In-network Data Estimation

Scientific applications often require data measurements at pre-defined grid points, which are often different from the locations of the raw data provided directly by the sensor network. As a result, a sensor-driven scientific/engineering application requires a virtual layer, where the logical representation of the state of the environment provided to the applications may be different from the raw measurements obtained from the sensor network. The *iZone* abstractions described in this section enables applications to specify such a virtual layer as well as implement the models (e.g., regression models, interpolation functions, etc.) that should be used to estimate data on the virtual layer from sensor readings.

2.1 The *iZone* abstraction

As mentioned above, there is often a mismatch between the discretization of the physical domain used by the application and the physical data measured by the sensor network and as a result, data from the sensors has to be processed before it can be coupled with simulations. The goal of the *iZone* abstraction is to support such an integration. It essentially abstracts away the details of the underlying measurement infrastructure and hides the irregularities in the sensor data by virtualizing the sensor field. The result is a consistent representation over time and space to match what is used by the simulations.

The *iZone* itself is thus a representation of the neighborhood that is used to compute a grid point, and can be specified using a range of coordinates, a function, etc. The *iZone* abstraction enables the implementation of the estimation functions. For example, interpolation algorithms, such as regressions, inverse distance weighing (IDW), and kriging, require the definition of an interpolation zone, an *iZone*, which defines the neighborhood around the grid point to be estimated, and such neighborhood is then used to compute that interpolation point. Note that for several interpolation algorithms, this zone may change on the fly based on the constraints provided by the application. The *iZone* abstraction also provides operators for obtaining sensor measurements corresponding to the region of interest as well as for defining in-network processing operators to compute a desired grid point from sensor values of this region. The semantics of operators of *discover*, *expand*, *shrink*, *get*, *put* and *aggregate* are list in Table 1.

Table 1. The *iZone* operators

Operator	Semantics
discover	Discover sensors within an *iZone*
expand	Expand an *iZone* by adding additional sensors
shrink	Shrink an *iZone* by removing sensors
get	Collect data from sensor(s) in the *iZone*
put	Send data to sensor(s) in the *iZone*
aggregate	Aggregate sensor data using reduction operators, such as max, min, weighted_avg, etc.

Once an *iZone* is defined, computing the data value at a grid point consists of (1) identifying a master node that coordinates the estimation process, which could be the sensor node that is closest to the grid point and has the required capabilities and resources, (2) discovering the sensors in the *iZone* that will be used in the estimation, (3) planning the in-network estimation strategy based on desired cost/accuracy/energy tradeoffs, and (4) performing the estimation and returning the computed data value at the desired grid point.

3 STaR: Space, Time and Resource Aware Optimization

This section explores temporal and spatial correlations in the sensor measurements to reduce costs and handle sensor dynamics, while bounding estimation errors for a given *iZone*. For example, a subset of the sensors in an *iZone* may be sufficient to satisfy the desired error bounds while reducing the energy consumed. Further, temporal regression models can be used to handle transient data unavailability, which may otherwise lead to a significant increase in costs and energy consumption [5].

3.1 Saving Energy using *iSets*

Typically, for a densely deployed sensor network, a subset of sensors across an *iZone* may be sufficient to meet the quality requirements for in-network data estimation. In this case, sensors in the *iZone* are divided into multiple interpolation sets, *iSets*, each of which can be used to estimate the data points while still satisfying the error bounds, and reducing costs and energy consumed. The *iSets* are generated and maintained at runtime to balance cost and energy as well as to tolerate failures.

The problem of generating the *iSets* can be formalized as follows: assume that an *iZone* Z is divided into m exclusive subsets $\{S_1, S_2, ..., S_m\}$ (such that $S_i \cap S_j = \emptyset$ and $S_1 \cup S_2, ..., \cup S_m = Z$), and each subset k ($k = 1, 2, ...m$) contains N_{a_k} number of sensors. The objective is to find "best" collection of *iSets* that satisfies data quality requirements.

The *iSets* should satisfy three requirements: (1) the interpolation error for *EACH iSet* should be less than the specified error tolerance; (2) the average number of sensor measurements for each *iSet* should be minimized in order to reduce the energy consumed; (3) the average aggregated error (i.e., $1/m \sum_k err(S_k)$) should be minimized in order to achieve best data quality whenever possible. In addition to these basic requirements, further constraints may be added to satisfy additional resource consumption and scheduling requirements. For example, the sizes of the *iSets* should be similar to make resource consumption more balanced and the scheduling easier. Similarly, the variance of interpolation errors across the *iSets* should be as small as possible.

The *iZone* is thus divided into m *iSets*, only one *iSet* of which needs to be active at a time. These *iSets* can now be scheduled in a round-robin fashion. Note that, as the number of *iSets* increases (i.e., the average size of *iSets* decreases), the efficiency of the approach increases as well. The generation and maintenance of *iSets* is illustrated in Figure 1 and is described below.

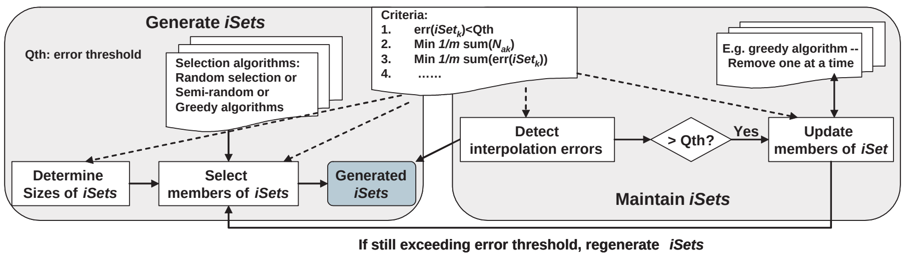
Generate iSets
Qth: error threshold
Selection algorithms: Random selection or Semi-random or Greedy algorithms
Criteria:
1. err(iSet_k)<Qth
2. Min 1/m sum(N_ak)
3. Min 1/m sum(err(iSet_k))
4. ……
E.g. greedy algorithm -- Remove one at a time
Determine Sizes of iSets
Select members of iSets
Generated iSets
Detect interpolation errors
> Qth?
Yes
Update members of iSet
Maintain iSets
If still exceeding error threshold, regenerate iSets

Fig. 1. An overview of generating and maintaining *iSets*.

3.2 Generating *iSets*

Determining the sizes of *iSets*: The appropriate size of the *iSets* used for interpolation is determined based on the specifications provided by the application, and computed (offline or online) using a stochastic approach as follows. The size of an *iSet* is first initialized to 3 (i.e., $k = 3$). The sensors used for interpolation tests are randomly selected, and the number of tests is set to some reasonable number (i.e., $N_{tr} = 50$). If the interpolation error is within the error tolerance threshold Q_{th}, the successful interpolation counter *succ* is incremented. If the success rate (i.e., $succ/N_{tr}$) is greater than a specified percentage θ, the algorithm terminates and the current size of the *iSet* is returned as the desired *iSet* size. If the success rate is less than θ after all the tests are completed, the size of *iSet* is incremented and the procedure is repeated until the size equals the size of the *iZone*. Note that this is only done once.

Selecting members of the *iSets*: Once the size of an *iSet* is determined, members of each *iSets* are selected so as to satisfy the criteria discussed earlier. An straightforward approach is to use an exhaustive search to find the optimal collection of *iSets*. This approach is obviously expensive for reasonably sized *iZones*, and as a result, we propose approximate algorithms, random, semi-random and greedy algorithms, to find near optimal *iSets*, as described below.

Random algorithm Given N sensors in an *iZone*, this algorithm randomly generates m mutually exclusive *iSets*, each approximately of size k ($N_{a_k} \approx k$), and $\sum_{k=1}^{m} N_{a_k} = N$. The interpolation error of each *iSet* is then evaluated. If the error for every *iSet* is below the error threshold, Q_{th}, the aggregated error across all the *iSets* is computed and saved. This process is repeated several times. The collection of *iSets* that leads to the smallest aggregated error is finally selected as the initial collection of *iSets*. The number of trials N_{tr} may be explicitly specified or computed based on observed (or historical) data. The actual value depends on the characteristics of sensor data. For example, for the dataset used in the experiments in the paper (see Section 4), a suitable value is between 50 and 100.

Semi-random algorithm This algorithm is based on the heuristic that if the sensors in an *iSet* is more uniformly distributed across the *iZone*, the estimation has higher chance to be more accurate. This algorithm attempts to assign neighboring nodes to different *iSets*. This is done using locality preserved space filling curves [6] as the indexing mechanism. First, each sensor is indexed using the Hilbert space filling curve (SFC) based on their locations. Within a given *iZone*, sensors with neighboring identifiers are then virtually grouped based on their SFC indices, so that the size of each virtual group is equal to the number of *iSets* required. For example, if m *iSets* are needed, each group would have m members. The *iSets* are now constructed by selecting one sensors from each of the virtual groups. Note that the selection of sensors from each of the virtual groups is random.

Greedy algorithms Three of greedy algorithms are also devised to select appropriate sensors for the *iSets*. These algorithms are described below.

Greedy Algorithm 1 – "Remove one node at a time": In this algorithm, we start with one *iSet* containing all sensors in the *iZone*. Sensors are then removed one at a time. The removed sensor is the one that leads to minimal interpolation error at each step, until desired *iSet* size is achieved. That is, given the *iSet* k, sensor node j such that

$$j = \arg\min_{i \in S_k} err(S_k - i)$$

is removed from the *iSet*. We then use a random algorithm to select the appropriate members of the last two *iSets*. This is because, if all *iSets* were generated using "remove one at a time", the last *iSet* would come up together with the $(m-1)$th one without any chance to make any local optimal selection to minimize estimation errors as other *iSets*, and thus usually cannot meet quality requirements. As a result, the last two *iSets* are chosen with another methods, such as random algorithm or the second greedy algorithm.

Greedy Algorithm 2 – "Add one node at a time": The second algorithm starts with a single sensor node in the initial *iSet*, and add one node at a time while maintaining interpolation error constraints. That is, given the *iSet* k, node j, which has not been assigned to any *iSet* and minimizes the interpolation error, such as

$$j = \arg\min_{i \in Z \setminus S_k} err(S_k \cup i),$$

is added to the *iSet*. This process is repeated until all nodes are assigned to *iSets*.

Greedy Algorithm 3: This algorithm uses the heuristic that nodes which are far from each other are less correlated and as a result are good candidates to add into the existing *iSets*. The idea is to select sensors that far from the last selected sensor. To implement this algorithm, we used the SFC-based indexing method described as part of the semi-random algorithm. Sensor nodes are first indexed using the Hilbert SFC. Virtual groups with sizes equal to number of *iSets* are then formed base on their SFC indices.

The algorithm is initialized by assigning sensors of one virtual group to each *iSet*. Next, sensors are permutated from one of remaining virtual groups, and are mapped to each of m *iSets*. The permutation leading to the least aggregated interpolation error is added to each of them respectively at a time. This step is repeated until all the sensors are assigned to the *iSets*. As stated with the heuristic, the sequence of which virtual group to be added to the *iSets* has the impact on the accuracy of estimations using the *iSets*. As a result, a preprocessing step is used to find such good sequence(s) either online or using historical data. Note that this only needs to be done once.

Once sensors are assigned to *iSets*, interpolation are performed. Next, we describe how to maintain *iSets* when the underlying system changes at runtime.

3.3 Maintaining *iSets* at runtime

Due to the dynamics of underlying physical environment and the sensor system, currently valid *iSets* may not satisfy data quality requirements in the future. As a result, mechanisms are needed to maintain *iSets* to ensure that they continue to meet data quality requirements. In this section, we describe how to maintain *iSets*. We also assume that the sensor network is clustered to construct a two level self-organizing overlay of sensors, in which cluster heads perform coordination of the *iZone*.

Our approach is as follows. First, interpolation errors are tracked by using localized error models at each individual cluster. The error models are localized so that a violation of error thresholds can be detected locally without communicating with other clusters. When a threshold violation is detected, a greedy algorithm is used to update the involved *iSet* to improve estimation quality whenever possible.

Generating models for interpolation errors: It is noted that interpolation errors are often correlated with relevant sensor measurements. As a result, regression models can be used to describe the relationship $e(t) = f(v_k(t))$ between interpolation errors $e(t)$ and the current measurement $v_k(t)$ of sensor k. A combination of offline and online estimation methods can be used to learn such a relationship, in which the coarse trends of error models are learned using offline methods using historical data, while specific local parameters can be learned at runtime. For example, an offline study may suggest that a regression model with degree one, i.e., $a_1 v_k + b_1$, should be used. The model parameters a_1 and b_1 are estimated using previous values of sensor k and the corresponding interpolation errors at runtime at individual cluster heads. These models can then be used to estimate interpolation errors using measurement of sensor k from local cluster.

Maintaining *iSets* using a greedy algorithm: When an *iSet* only temporarily exceeds error thresholds, the greedy "remove one at a time" algorithm can be used at each cluster to temporarily remove sensor measurements from that *iSet*. Each cluster makes recommendation of which node(s) to remove, and the recommendation that provides the least interpolation error is enforced. Note that if the interpolation error still exceeds error threshold, the *iSets* needs to be regenerated.

Transient unavailability using temporal estimation: Since unavailability of scheduled sensors requires re-collection of raw data and thus resulting in expensive communication and energy consumption, temporal models are used to estimate the missing sensor measurement. The idea is to use the fact that neighboring sensor nodes would experience similar changes. As a result, samples from neighboring sensors can be used to facilitate the estimation of temporal model parameters, such as degree of regression model, length of time-series. Note that these parameters would change as the underlying physical characteristics vary. The actual coefficients of temporal model are determined based on previous values of the missing sensor data. The evaluations of these optimized in-network mechanisms are presented next in Section 4.

4 Experimental Evaluation

In this section a simulator is used to evaluate the performance of in-network data estimation mechanisms. The simulator implements the space, time, and resource aware optimization mechanisms for realistic scenarios. The scenarios in the experiments are driven by a real-world sensor dataset obtained from an instrumented oil field with 500 to 2000 sensors, and consisting of pressure measurements sampled 96 time per day. A two-tiered overlay with about 80 clusters is initialized. More powerful nodes are elected as cluster heads and also perform the in-network estimations.

In each experiment, about 500 instances of in-network interpolations are performed on pressure values obtained from simulated sensor nodes. Communication costs are evaluated with and without optimizations. The accuracy and costs are also evaluated in the presence of dynamics of physical environments and sensor systems. Finally, the cost of generating *iSets* is examined using the random, semi-random and greedy algorithms. The primary metrics used in the evaluation are communication cost, measured in terms of number of messages transmitted within the network, and accuracy, measured in terms of relative or absolute interpolation errors.

4.1 Communication costs

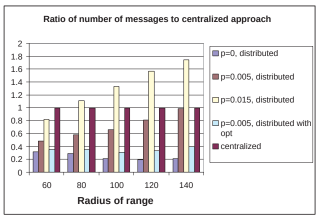

Fig. 2. Communication cost in the presence of sensor dynamics.

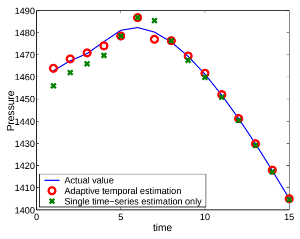

Fig. 3. Adaptive temporal interpolation.

The current *iZone* prototype implements a distributed in-network mechanism, in which parameters corresponding to the estimation model are first computed at the cluster heads. The cluster heads coordinate the estimation process, and distribute those parameters to the selected *iZone* sensors. A decentralized energy-efficient aggregation scheme is then used to estimate the data. To simulate transient unavailability of sensors, each sensors are given the same unavailability rates, which are the frequencies that scheduled sensors are not available at the time of interpolation. The communication costs are normalized to the cost of the baseline centralized approach, where a coordinator sensor collects raw measurement from selected *iZone* sensors and does the estimation. As plotted in Figure 2, the distributed approach performs best when the sensor system is static. With a small unavailability rate of 0.5%, the communication cost increases by over 50% for an *iZone* radius of 60, and over 4 times for a radius of 140. The cost also increases as the unavailability rate increases.

Effectiveness of temporal estimation: In this experiment, the performance of using temporal estimation for temporarily unavailable sensor data is evaluated. Model parameters such as the order of regression functions, and the length of historical data used for the estimation, are chosen at runtime. The estimation parameters varies over time. For example, the length of used historical data is changed from 8 to 6 after interpolation time 6, and the degree of regression models is changed from 3 to 2 after interpolation time 5. The adaptive temporal estimation using spatial-temporal information from neighboring nodes (i.e., circles in Figure 3) is much closer to the actual values (i.e., solid line) than that using only historical measurements from sensors with temporally unavailable data (i.e., crosses in Figure 3). This is because the spatial-temporal models of neighboring nodes can better catch the changes of the underlying physical characteristics than that only using the model from temporally unavailable sensor. Furthermore, as plotted in Figure 2, by using adaptive temporal estimations for temporarily unavailable sensor measurements when possible, the communication cost is reduced by about 25% for a radius of 60, and about 60% for a radius of 140.

4.2 Effectiveness of generating *iSets*

In these experiments, the effectiveness of using random, semi-random and greedy algorithms to generate *iSets* for in-network interpolation tasks are investigated, and three *iSets* are formed within the given *iZones* for this set of experiments.

Effectiveness of random-based algorithms The histograms of average interpolation errors for each of three generated *iSets* are plotted in Figure 4 using random and semi-random algorithms respectively. The probability of smaller average interpolation errors using the semi-random algorithm is much higher than that using the random algorithm. For example, for an average interpolation error less than 0.2%, the semi-random algorithms (e.g., with probability about 32%) have higher probability to generate *iSets* meeting quality requirements than that using the random algorithms (e.g., with probability about 12%).

In Figure 5, the standard deviation of interpolation errors is examined for the two algorithms. The random algorithm gives much larger variation than the semi-random algorithm. For the random-based algorithms, within the same range of interpolation error (i.e., 0.62%), the standard deviation of random method is still much larger than that of semi-random algorithm. This tells us that the semi-random algorithm is generally more effective in finding the collections of *iSets* having both small interpolation errors and a small variation of such errors.

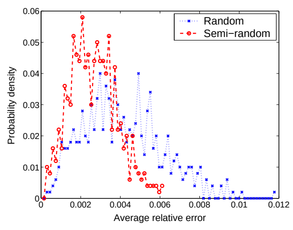

Fig. 4. Histograms of interpolation errors of generated *iSets*

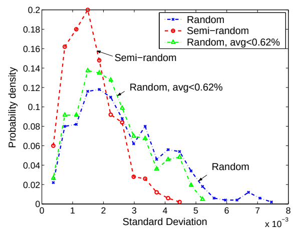

Fig. 5. Histograms of variation of interpolation errors of generated *iSets*

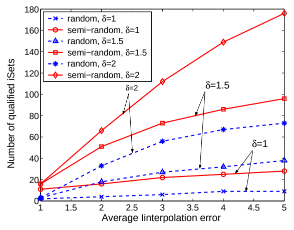

Fig. 6. Effectiveness of random-based algorithm

In Figure 6, the number of collections of *iSets* is counted in terms of the absolute average interpolation errors and deviations (i.e., δ) among *iSets*. More candidates meeting those requirements are available using semi-random algorithm than that using random algorithm. For example, with small variance 1 (i.e., $\delta = 1$), the available number of collections using semi-random algorithm is five more times than that using random algorithm, which indicates the effectiveness using the semi-random algorithm especially with higher quality requirements.

Effectiveness of greedy algorithms First the three greedy algorithms, as well as random and semi-random algorithms are compared in terms of interpolation errors. Three *iSets* are generated in this example. As shown in Figure 7, the first greedy algorithm, "remove one at a time", behaves well for most of the generated *iSets* except the last one. For this algorithm, the last *iSet* exhibits high error since it has no chance to exploit local optimization. The last two *iSets* are thus chosen using other algorithms, such as random algorithms or "add one at a time" greedy algorithm. The second algorithm, "add one at a time" may give relative balanced results, however, the overall error rates could be higher than that of random and semi-random algorithms. The third greedy algorithm gives good accuracy performance comparing to random and semi-random algorithms. The tradeoff is that it needs pre-processing to find good sequence of the next explored sensors. However, the pre-processing could be performed offline and its cost is much less than that of random and semi-random algorithms.

4.3 Tradeoffs between accuracy and energy consumption

In this section, the tradeoff of accuracy and energy consumption is examined. The energy consumption is normalized to one when all sensors in an *iZone* are active at each time. As shown in Figure 8, as the sizes of *iSets* becomes smaller, less energy is consumed and the interpolation errors become greater. For example,

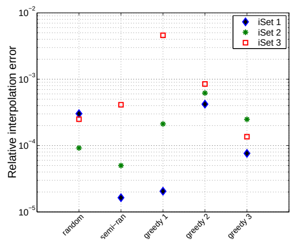

Fig. 7. Interpolation errors of algorithms generating *iSets*

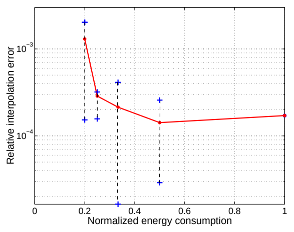

Fig. 8. Tradeoff between interpolation error vs. energy consumption

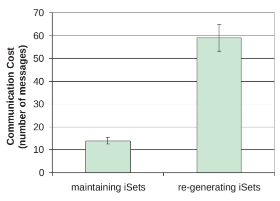

Fig. 9. Cost of maintaining *iSets*

when half of the nodes are active, approximately half of the energy can be saved, and the range of interpolation errors of selected *iSets* are greater. In addition, the maximum error is slightly greater than when using all nodes. When only one third or one fourth of nodes are active, the maximum error is similar to that of using only half of all nodes. This means that only a portion of active sensors may be able to meet accuracy requirements while saving additional energy. However, when only one fifth of nodes are active, the errors (both maximum and average) become much larger and not meet application requirements anymore.

4.4 Cost of maintaining *iSets*

In this experiment, the communication cost of maintaining *iSets* at local cluster heads is examined. The cost primarily consists of exchanging information, such as calibration information, identifiers of temporarily removed sensors between cluster heads. As shown in Figure 9, the cost of maintaining *iSets* at cluster head introduces much less communication overhead than regenerating *iSets*. Furthermore, the original error before removing a sensor is .06069% (threshold is .06%), and after removing one of them using greedy algorithm, the result becomes .00922%, which is far lower than the threshold. This method is quite effective when the change of physical phenomenon is just temporary. After this, the original *iSet* can be used again with the error rate lowered to .04681%. As a result, this method significantly reduced the frequency to re-generate the whole *iSets* in the *iZone*.

5 Related Work

There are some recent research efforts [7, 8] that use the concept of virtual sensors to support sensor applications. VNLayer [7] provides abstraction layers that mask uncertainty of underlying sensor networks through consistency management. Virtual sensor [8] provides a virtual sensor model and application APIs to support heterogeneous aggregation and hierarchical specifications. However, the underlying concepts and implementation of the system described in this paper is quite different from these approaches in that it uses virtualization to address the mismatch between the instrumentation of the physical domain and its discretization in the computational model, rather than to create, for example, a virtual sensor for a derived data type.

Data aggregation is an essential functionality in sensor networks, and has been addressed by a number of research efforts [9, 10]. Optimizations techniques such as aggregation trees are used to resolve queries efficiently. Homogeneous aggregation operations are supported. The approach presented in this paper supports used-defined functions using in-network coordination and optimization mechanisms. The sensor selection schemes are also closely related to our work. The sensor selection approach described in [11] uses approximation algorithms to select near-optimal subsets of k sensors that minimize the worst-case prediction error. Entropy-based approaches [12] are used for the sensor selection problems of target tracking and localization applications. The goal of our proposed algorithms is to find a *"best" collection* of subsets, *all* of which satisfy the error tolerance rate (and minimize aggregated errors), while saving and balancing the energy consumptions among sensors in the long run.

6 Conclusion

This paper investigated abstractions and mechanisms for in-network data processing that can effectively satisfy dynamic data requirements and quality of data and service constraints, as well as investigate tradeoffs between data quality, resource consumptions and performance. Specifically, the proposed mechanisms (i) allow flexibility in the specification of relevant subsets of a sensor network with *iZones* and *iSets*; (ii) explore space, time and resource aware optimizations that utilize the spatial and temporal correlation among sensor measurements to reduce costs while bounding estimations errors; (iii) are robust with respect to network dynamics; and (iv) provide a virtualization of the physical sensor grid to match the representation of the physical domain used by the models, and can dynamically discover and access sensor data independent of any change to the sensor network itself. Experimental results show that the proposed in-network mechanisms can facilitate the efficient usage of constraint resources and satisfy data requirement in the presence of dynamics and uncertainty.

References

1. Parashar, M., Matossian, V., Klie, H., Thomas, S.G., Wheeler, M.F., Kurc, T., Saltz, J., Versteeg, R.: Towards dynamic data-driven management of the ruby golch waste repository. Proceedings of the Workshop on Distributed Data Driven Applications and Systems, International Conference on Computational Science (ICCS) (2006)
2. Werner-Allen, G., Lorincz, K., Ruiz, M., Marcillo, O., Johnson, J., Lees, J., Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network. Second European Workshop on Wireless Sensor Networks (2005)

3. Kottapalli, V.A., Kiremidjiana, A.S., Lyncha, J.P., Carryerb, E., Kennyb, T.W., Lawa, K.H., Lei, Y.: Two-tiered wireless sensor network architecture for structural health monitoring. SPIEs 10th Annual International Symposium on Smart Structures and Materials (2003)
4. Szlavecz, K., Terzis, A., Musaloiu-E., R., Cogan, J., Small, S., Ozer, S., Burns, R., Gray, J., Szalay, A.S.: Life under your feet: An end-to-end soil ecology sensor network, database, web server, and analysis service. MSR-TR-2006-90 (2006)
5. Jiang, N., Parashar, M.: Programming support for sensor-based scientific applications. Proceedings of the Next Generation Software (NGS) Workshop in conjunction with the 22nd IEEE International Parallel and Distributed Processing Symposium (IPDPS) (2008)
6. Sagan, H.: Space-Filling Curve. Springer Verlag (May 1995)
7. Brown, M., Gilbert, S., Lynch, N., Newport, C., Nolte, T., Spindel, M.: The virtual node layer: A programming abstraction for wireless sensor networks. ACM SIGBED Review **4**(3) (2007) 7–12
8. Kabadayi, S., Pridgen, A., Julien, C.: Virtual sensors: abstracting data from physical sensors. Proceedings of the 2006 International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM) (2006) 587 – 592
9. Yao, Y., Gehrke, J.E.: The cougar approach to in-network query processing in sensor networks. Sigmod Record **31**(3) (September 2002)
10. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: TAG: a Tiny AGgregation service for Ad-Hoc sensor networks. in Proceedins of the USENIX Symposium on Operating Systems Design and Implementation (2002)
11. Das, A., Kempe, D.: Sensor selection for minimizing worst-case prediction error. International Conference on Information Processing in Sensor Networks (IPSN'08) (April 2008)
12. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration for tracking applications. IEEE Signal Processing Magazine (2002)