Data Quality and Query Cost in Wireless Sensor Networks

David Yates
Computer Information Systems Dept.
Bentley College
Waltham, Massachusetts 02452
Email: dyates@bentley.edu

Erich Nahum
IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, New York 10532
Email: nahum@watson.ibm.com

Jim Kurose
and Prashant Shenoy
Dept. of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003

***Abstract*— This research is motivated by emerging, real-world wireless sensor network applications for monitoring and control. We examine the benefits and costs of caching data for such applications. We propose and evaluate several approaches to querying for, and then caching data in a sensor field data server. We show that for some application requirements (i.e., when delay drives data quality), policies that emulate cache hits by computing and returning approximate values for sensor data yield a simultaneous quality improvement and cost savings. This win-win is because when system delay is sufficiently important, the benefit to both query cost and data quality achieved by using approximate values outweighs the negative impact on quality due to the approximation. In contrast, when data accuracy drives quality, a linear trade-off between query cost and data quality emerges. We also identify caching and lookup policies for which the sensor field query rate is bounded when servicing an arbitrary workload of user queries. This upper bound is achieved by having multiple user queries share the cost of a sensor field query. Finally, we demonstrate that our results are robust to the manner in which the environment being monitored changes using two different sensor field models.**

I. Introduction

Over the next several years, wireless sensor networks will enable many new sensing applications, ranging from environmental and infrastructure monitoring to commercial and industrial sensing. There are many performance metrics of interest for sensor networks. We focus on two that are common to the vast majority of sensor networks:

1) The *accuracy* of the data acquired by the application from the sensor network; and
2) the total *system end-to-end delay* incurred in the sequence of operations needed for an application to obtain sensor data.

Although almost all sensor network applications have performance requirements that include accuracy and system delay, their relative importance may differ between applications. We therefore define the *quality* of the data provided to sensor network applications to be a combination of accuracy and delay. As in most systems, improved quality usually comes at some *cost*. For current wireless sensor networks, the most important component of cost typically is the energy consumed in providing the requested data. In turn this is dominated by the energy required to transport messages through the sensor field. This cost versus quality trade-off has recently been an active area of research [1], [2], [3], [4], [5].

To perform our research, we construct a sensor network model. We then develop novel policies for caching sensor network data values in the gateway server, and then retrieving these values via cache lookups. We also propose a new objective function for data quality that combines accuracy and delay. Finally, we use our sensor network model to assess the impact of several factors on query cost and data quality performance:

- Our caching and lookup policies; the
- relative importance of data accuracy and system end-to-end delay; and the
- manner in which the sensed data values in the environment change.

This assessment evaluates seven different caching and lookup policies by implementing them in a simulator based on CSIM 19 [6], [7].

Almost all sensor network deployments have three main components:

1) One or more sensor fields consisting of sensor field nodes that communicate with one or more base stations;
2) One or more data servers (or gateways) that accept requests for sensor data and generate replies for these requests; and
3) Monitoring and control centers that are connected to the appropriate sensor data servers via a backbone network.

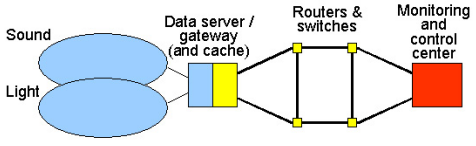

Fig. 1. Sensor Network Deployment Example.

Figure 1 shows an example of such a deployment with two sensor fields, one data server, and one monitoring and control center. If the data server shown in this figure is augmented with storage, it can store and cache sensor field values that are carried in query replies.

The caching approaches we propose are designed to be general since they make no assumptions about whether the overall sensor network architecture uses a structured or unstructured data model. In other words, our approaches are independent of the database model for the sensor network. The database could implement a structured schema that extends a standard like the Structured Query Language (SQL). The TinyDB and Cougar systems both advocate this approach [8], [9], [10]. However, the schema could also be modified while the system is running (e.g., as in IrisNet [11]). The database model might also expose a low-level interface to the application. Directed Diffusion [12] does this by allowing applications to process attributes or attribute-value pairs directly.

A. Data Acquisition and Caching in Sensor Networks

Consider the impact of adding a cache to the data server or gateway in Figure 1. Figure 2 shows such a system in which a cache is added to the internal architecture of the server, on the "border" between the sensor field(s) and the backbone network. There are two possible data paths that

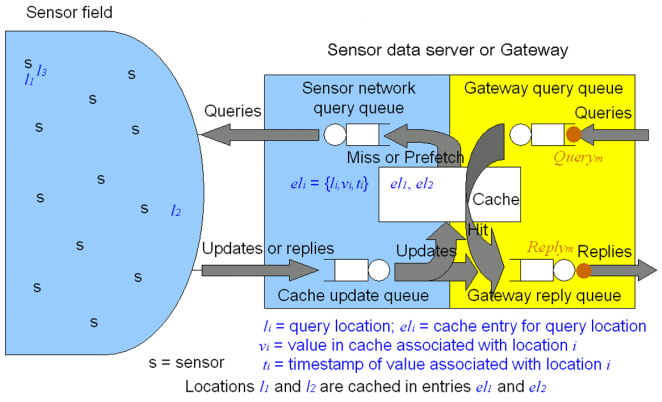

Fig. 2. Sensor Network Data Server or Gateway with a Cache.

can be traversed in response to a query from the backbone network:

- For a cache miss, a query is sent to the sensor field by the gateway, incurring a cost. To update the cache, each sensor data value v_i is copied into a cache entry. A cache entry, el_i, associates with location l_i, the most recent value observed at this location, and its timestamp into the tuple $\langle l_i, v_i, t_i \rangle$. We say that the *system delay*, S_d, is the time between an application query arriving at the point labeled *Query*$_m$ in Figure 2 and the corresponding reply departing from *Reply*$_m$. The *value deviation*, D_v, is the unsigned difference between the data value in *Reply*$_m$ and the true value at l_i when *Reply*$_m$ leaves the gateway reply queue.
- The data path for a cache hit is much shorter than for a cache miss. For example, if the cache is indexed by location, and a cache entry is present for a location l_i specified in a query, a reply can be generated using only the information in the tuple that corresponds to l_i. Since the processing required to perform this cache lookup and generate a reply is relatively small, we assume that the system delay for a cache hit (S_d) and its associated cost are both zero. We also determine the value deviation for cache hits (D_v) in the same way as for cache misses.

We exploit spatial locality within sensor field data in the cache. Specifically, some caching and lookup policies allow cache "hits" in which the value at location l_i is approximated based on values $v_{i'}$ from neighboring location(s) $\{l_{i'} \in N(l_i)\}$. (Here $N(l_i)$ denotes the neighborhood of location l_i.) We develop and describe three such policies that implement what we call *approximate* lookups and queries. We compare these approximate policies with four *precise* lookup and query policies that only use information associated with location l_i to process queries that reference location l_i.

II. Cost and Quality in Sensor Networks

A. Caching and Lookup Policies

Our caching and lookup policies are designed to explore alternative techniques for increasing the effective cache hit ratio, and thus conserving system resources.

All of the caching and lookup policies we propose and evaluate incorporate an age threshold parameter T that specifies how long each entry is stored in the cache. We now describe all seven of our caching and lookup policies. *All hits*, *all misses*, *simple lookups* and *piggybacked queries* implement precise lookups and queries. On the other hand, *greedy age lookups*, *greedy distance lookups*, and *median-of-3 lookups* implement approximate lookups and queries.

- **All hits** (age threshold parameter $T = \infty$): In this policy cache entries are loaded into the cache but are never deleted, updated, or replaced.
- **All misses** (age parameter $T = 0$): In this policy entries are not stored in the cache.
- **Simple lookups** (T): This caching policy results in a cache hit or a cache miss based on a lookup at the location specified in the user query. Once an entry is loaded into the cache, it is stored for T seconds and then deleted.
- **Piggybacked queries** (T): A cache hit or miss is determined only by a lookup at the location specified in the user query. If a query has already been issued to fill the cache at a particular location, subsequent queries block in a queue behind the original query and leverage the pending reply to fulfill multiple queries.
- **Greedy age lookups** (T): A cache hit or miss is determined by a lookup first at the location specified in the query, and second by lookups at all neighboring locations. If there is more than one neighboring cache entry, the freshest (newest) cache entry is selected. As for piggybacked queries, if a query has already been issued to fill the cache at any of these locations, subsequent queries block in a queue behind the original query and leverage the pending reply to fulfill multiple queries.
- **Greedy distance lookups** (T): A cache hit or miss is determined by a lookup first at the location specified

in the query, and second by lookups at neighboring locations. If there is more than one neighboring cache entry, the nearest cache entry is selected.

- **Median-of-3 lookups** (T). A cache hit or miss is determined by a lookup first at the location specified in the query, and second by lookups at all neighboring locations. If there are at least three neighboring cache entries, the median of three randomly selected entries is selected as the value returned with a cache hit. If there are one or two neighboring cache entries, a randomly selected entry provides a cache hit. Otherwise, the query is treated as a miss.

By implementing blocking behind pending sensor field queries, four of these seven policies have an upper bound on the sensor field query rate, R_f. Specifically,

$$\max(R_f) \quad = \quad \frac{|\mathbf{N}|}{T}. \tag{1}$$

The four policies are piggybacked queries, median-of-3 lookups, and the two approximate greedy policies. In Equation (1), $|\mathbf{N}|$ is the number of distinct locations that can be specified in queries for sensor data.

B. Sensor Network Data Quality and Query Cost

We normalize sensor network data quality in order to compare quality measurements from different sensor networks, as well as for different system parameters (e.g., number of sensors, distance between sensors, etc.) We define data quality to be a linear combination of normalized *system delay* and normalized *value deviation* using a parameter A, which is the relative importance of delay when compared with value deviation. The expression that defines quality, denoted Q_n, is:

$$Q_n \quad = \quad A \, \frac{1}{(1+e^{-b})} + (1-A) \, \frac{1}{(1+e^{-c})} \tag{2}$$

where $-b$ and $-c$ are the exponents used to perform *softmax normalization* on delays and value deviations, and $0 \leq A \leq 1$. The exponents in Equation (2) are the z scores of their respective values and are therefore defined as follows:

$$-b \quad = \quad -\frac{S_d - \text{mean}(S_d)}{\text{stddev}(S_d)}, \text{ and} \tag{3}$$

$$-c \quad = \quad -\frac{D_v - \text{mean}(D_v)}{\text{stddev}(D_v)}. \tag{4}$$

Since small values of system delay (S_d) and value deviation (D_v) are both desirable, smaller values of Q_n, e.g., $0 < Q_n \ll 0.5$ imply better data quality, and larger values of Q_n correspond to worse quality. Softmax normalization yields transformed values that lie in the range $[0, 1]$. Because of this property, and because of our definition of A, $0 \leq Q_n \leq 1$. This type of normalization has been used by others in neural networks; data mining for pattern recognition; and data classification [17], [18], [19], [20].

We use two different sensor field models in our research in order to generalize our results. The first model uses correlated random variables to simulate how the environment changes for 1000 sensor locations. This model gives us the flexibility to vary how the environment changes. The second model uses real-world trace data to drive how the environment changes. This trace data was taken from 54 light, temperature, and humidity sensors deployed in the Intel Berkeley Research lab over a five-week period [21].

C. Simulated Changes to the Environment

For the simulated changes to environment, the sensor field is a 3-dimensional field with rectangular planes on six faces. There is an 8-unit spacing between 10 sensors in the X-dimension, a 6-unit spacing for 10 sensors in the Y-dimension, and a 4-unit spacing for 10 sensors in the Z-dimension. Four base stations are placed on the X-Y plane. These four base stations are then connected to the gateway server that has the common cache. The sensors always communicate with their closest base station, and the properties of each one-way communication to and from location l are as follows:

$$\mathit{Cost}_l \quad = \quad p \, r_{b'}^2 \quad | \quad \min(\mathit{Cost}_l) = 1 \text{ unit} \tag{5}$$

where $r_{b'}$ is the distance between location l and its nearest base station b', and p is the normalization constant for the set of costs. In addition,

$$\mathit{Delay}_l \quad = \quad q \, r_{b'} \quad | \quad \max(\mathit{Delay}_l) = 1 \text{ second} \tag{6}$$

where q is the normalization constant for the set of delays. We assume that all four base stations communicate with the gateway server containing the cache at zero cost, with zero delay, and using infinite bandwidth. Finally, each base station is connected to the sensor field with an access link with a capacity of 25 queries per second.

D. Trace-driven Changes to the Environment

For the trace-driven changes to the environment, our second sensor field model has more than an order of magnitude fewer locations (54 instead of 1000). The sensors are arranged in a 2-dimensional field at the numbered locations in Figure 3, which is taken from [21]. Each entry in the trace is from a Mica2Dot sensor, which senses humidity, temperature, light, and battery voltage. The trace contains over 2.2 million entries taken over more than five weeks in early 2004. This means that one location reads and records new sensor field values

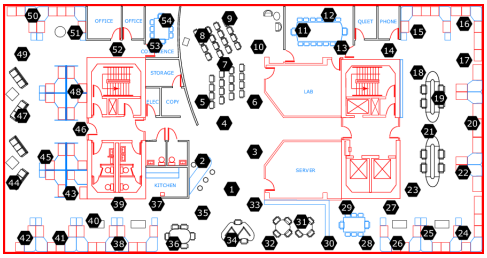

Fig. 3. Sensor Field at Intel Berkeley Research Lab.

an average of about once every 1.33 seconds. We wanted to use the most dynamically changing of the sensor field values

in our model to maximize the error in query accuracy. We therefore chose the value with the largest average difference between samples. This was light intensity, which is reported in Lux.

Again, we assume that sensors always communicate with their closest base station, and the cost and delay of each one-way communication are given by Equations (5) and (6), respectively.

E. Query Workload Model

We use a query workload model that is well suited for sensor network applications that include monitoring and control functions. Many of these applications have a workload that includes a periodic arrival process of queries as well as a random arrival process. There are examples of query workloads that capture both of these components in the literature, e.g., [12], [13], [15]. On the other hand, other researchers assume that queries either have exclusively periodic interarrival times [9], [10], [14] or random (usually exponential) interarrival times [8], [16]. We assume that the query workload for our applications consists of the superposition of two query processes: a polling component that slowly scans the sensor field at a fixed rate, and a random component that consists of queries to different locations in the sensor field. Within this random component it is equally likely that each location in the sensor field will be sampled. This workload model is similar to models used by others in [12], [13], [15]. Specifically, our query workload is characterized by two parameters:

- τ = the period of the polling component of the query workload ($\tau > 0$); and
- λ = the average query arrival rate of a process that represents the random component of our workload.

For simulated changes to the environment, λ and τ are fixed: $\lambda = 81$ queries per second is used as the rate parameter to generate queries with exponentially distributed interarrival times with mean $1/\lambda$. The parameter τ is set to $111.1\overline{1}$ seconds so that the arrival rate for polling queries is 9 queries per second. When $\lambda = 81$ and $\tau = 111.1\overline{1}$, the aggregate arrival rate for queries is $81 + 9 = 90$ queries per second. Since the total capacity of the sensor field access links is $4 \times 25 = 100$ queries per second, their average link utilization is 0.90 for "all miss" runs, and less for runs that include some cache hits.

For trace-driven changes to the environment, $\lambda = 0.81$ queries per second and $\tau = 600$ seconds. This makes the average arrival rate for queries two orders of magnitude less than query rate for simulated changes, namely 0.9 queries per second. The average link utilization is also 0.90 for "all miss" runs.

III. Discussion of Results

We wanted our simulated results to capture the fact that sensor field readings are correlated in both space and time. In our sensor field model, at time $t+1$, the value at each location l is drawn from a normal distribution with mean

$$\mu_{l,t+1} = \frac{1}{3}\mu + \frac{1}{3}\mu_{l,t} + \frac{1}{3}\mu_{N(l),t}.$$

The long-term mean of this distribution is $\mu = 0$. The standard deviation $\sigma = 0.407514$, and the tails are truncated at minimum / maximum values of $\sigma - 6\mu$ / $\sigma + 6\mu$. This standard deviation is the same as the standard deviation of the system end-to-end delays during a set of 20 runs without a cache for our 1000-node sensor network model. $N(l)$ denotes the neighbors of location l, and each neighboring location l' of l contributes to $\mu_{N(l),t}$ in proportion to $\mu_{l',t}/r_{l'}$, where $r_{l'}$ is the distance between locations l and l'. This model for a changing environment is based on the model for correlated sensor network data developed by Jindal and Psounis [22].

Each data value presented in our results is derived by averaging 20 simulation runs initialized with different seeds. Additional details of our experimental methodology are described in [23].

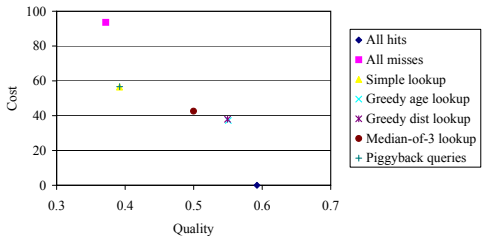

Fig. 4. Cost vs. Quality for $A = 0.1$ and Correlated changes over 1000 locations.

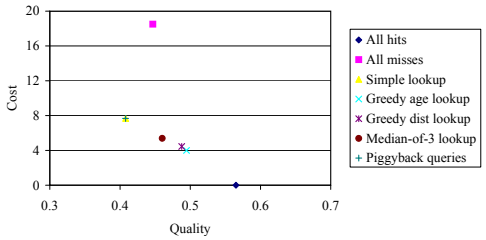

Fig. 5. Cost vs. Quality for $A = 0.1$ and Trace-driven changes over 54 locations.

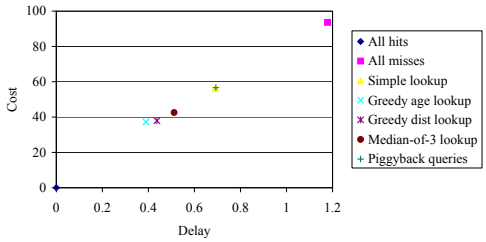

Fig. 6. Cost vs. Delay for $A = 0.1$ and Correlated changes over 1000 locations.

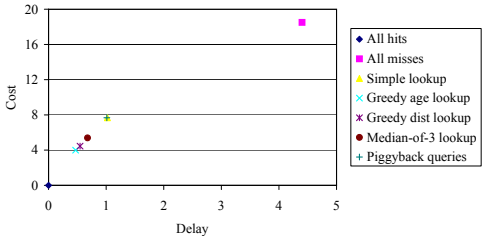

Fig. 7. Cost vs. Delay for $A = 0.1$ and Trace-driven changes over 54 locations.

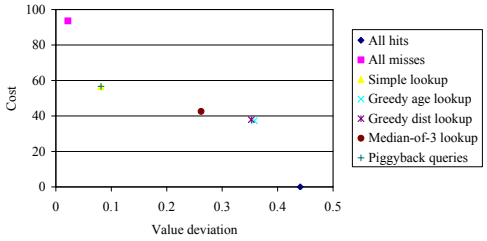

Fig. 8. Cost vs. Value deviation for $A = 0.1$ and Correlated changes over 1000 locations.

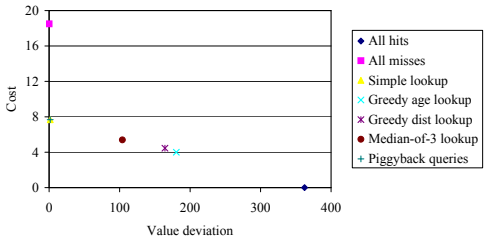

Fig. 9. Cost vs. Value deviation for $A = 0.1$ and Trace-driven changes over 54 locations.

The odd-numbered figures, Figures 5, 7, 9 and 11, show results for light intensity in Lux measured over time in the Intel Berkeley lab data set. These results are for $T = 90$ seconds, and 0.9 queries per second. Note that because of Equation (1), the maximum sensor field query rate, $\max(R_f)$, is reduced to $54/90 = 0.6$ queries per second. The even-numbered figures, Figures 4, 6, 8 and 10, are for correlated changes to the environment with both the age parameter, T, and the average query rate scaled for the more rapidly changing environment. Specifically, $T = 8.8\overline{8}$ seconds and the average user query rate is 90 queries per second. Because of Equation (1), the maximum sensor field query rate $\max(R_f) = 1000/T = 112.5$ queries per second.

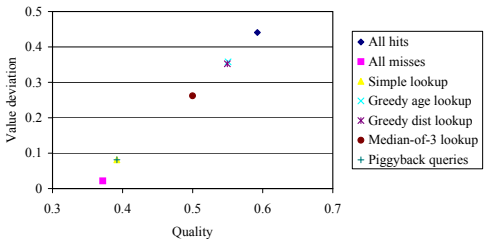

Fig. 10. Value deviation vs. Quality for $A = 0.1$ and Correlated changes over 1000 locations.

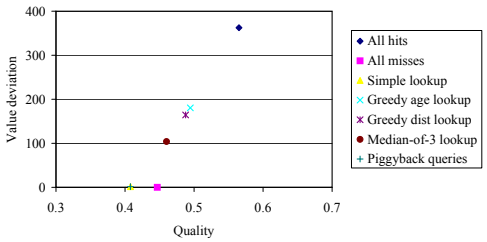

Fig. 11. Value deviation vs. Quality for $A = 0.1$ and Trace-driven changes over 54 locations.

We can draw two main conclusions from our experiments using the correlated and trace-driven models for how the environment changes. Results from these experiments appear in Figures 4 through 13.

1. There is a cost vs. quality trade-off for some data quality requirements but not others. For example, consider the results shown in Figures 4 and 5. Figure 5 shows cost versus quality for all seven caching and lookup policies, where $A = 0.1$ and the values at each location are changed according to the lab trace [21]. At the smallest cost, we have a 100% cache hit ratio (labeled "All hits") that provides a quality of below 0.6 for zero cost. For the largest cost, we see that a 0% cache hit ratio (labeled "All misses") provides the third-best quality at a cost of approximately 19 units. Recall that for quality, smaller values indicate better quality. The remaining five caching and lookup policies provide a linear trade-off between cost and quality. Figure 4 also shows a trade-off between cost and quality for the same value of A and the same seven caching and lookup policies, but with changes to the environment now modeled by a series of values correlated in space and time. There are two observations worth noting when comparing these first two figures. First, the cost values in Figure 5 are less than in Figure 4 because the distances within the sensor field are smaller. Second, the trends are similar between these two figures, with the exception of the increase in quality of the "all misses" policy between Figure 4 and Figure 5. This worse "all misses" quality is due entirely to an increase in the normalized delay term in the left hand side of Equation (2). This can be verified by comparing the relative differences in delays between the policies, shown on the horizontal axes in Figures 6 and 7.

We now examine system configurations for which delay is the more important component of quality in more detail. Figures 12 and 13 show such configurations for a value of $A = 0.9$. The most remarkable result in these figures is that there is no trade-off between cost and quality when we significantly prioritize delay over value deviation. The two greedy caching and lookup policies have the best cost performance and the best quality performance for both models of changing the environment in both Figures 12 and 13. Even though the "all hits" policy has the best absolute performance in these figures, we don't consider this a practical policy since it never updates the cache.

In studying Figures 4 through 13 it is interesting to understand which system variables depend on which system parameters. For example, cost, delay, and hit ratio values in these simulation results each depend on the following three variables:

- The caching and lookup policies themselves (including the value of T); the
- physical configuration of the sensor field; and the
- query arrival process.

Thus, a cost vs. delay or a cost vs. hit ratio graph is the same for different experiments in which these three variables are held constant. To see how cost and delay both increase with lower cache hit ratios, Table I shows the cache hit ratio for each of the (cost, delay) points in Figure 6. Similarly, Table II

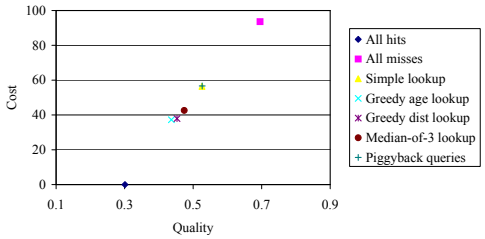

Fig. 12. Cost vs. Quality for $A = 0.9$ and Correlated changes over 1000 locations.

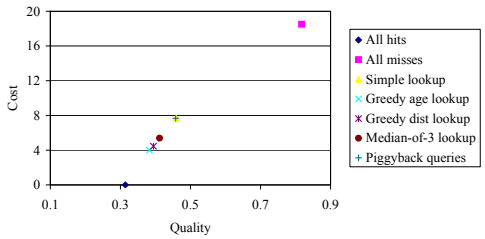

Fig. 13. Cost vs. Quality for $A = 0.9$ and Trace-driven changes over 54 locations.

shows the hit ratio for each of the (cost, delay) points in Figure 7.

Value deviation depends on the same parameters listed above, and additionally on the manner in which the environment changes. Thus, cost vs. value deviation graphs are the same when the policies, sensor field, query arrival process, and method for changing the environment are all identical. Figure 8 and Figure 9 show cost vs. value deviation results for correlated changes and trace-driven changes to the environment, respectively. The most interesting difference between the two figures is the overall increase the dispersion of the value deviations in Figure 9 when compared those in Figure 8. This is because the variation in sensor field values is much greater in the Intel Berkeley trace data than values that are drawn from our normal distribution with a time-dependent mean.

2. Different lookup policies perform best depending on whether delay or value deviation is most important to the application. If data quality is more important to the application than cost, and value deviation is more important than delay, simple lookups and piggybacked queries provide the best performance. This can be seen in Figures 4 and 5. In both of these figures, simple lookups and piggybacked queries yield the best quality, other than the "all misses" policy for correlated changes. When value deviation is most important, the expense of taking a cache miss (by not computing an approximate value from neighboring values for these two policies) is worthwhile, since value deviation is deemed most important. If query cost is at a premium compared with quality, using greedy age lookups or greedy distance lookups is preferred. These two policies have the most favorable cost performance in both sensor field models, other than the "all hits" case.

If delay is more important to quality than value deviation, Figures 12 and 13 show that performing greedy age lookups or doing greedy distance lookups yields the best performance. This is true regardless of whether cost or quality is more important to the application. We again assume that the "all hits" case is not useful to realistic applications. For these policies, getting the fast response time of a cache "hit" (which might be approximated from values at one or more neighboring locations) is worthwhile, since low delay is more important than a more accurate value.

TABLE I

HIT RATIOS, COSTS, AND DELAYS FOR $T = 8.8\overline{8}$, 90 QUERIES PER SECOND, AND CORRELATED CHANGES OVER 1000 LOCATIONS.

Policies	Hit ratio	Cost	Delay
All hits	1	0	0
All misses	0	94	1.18
Simple lookup	0.40	56	0.69
Greedy age lookup	0.62	37	0.39
Greedy distance lookup	0.60	38	0.44
Median-of-3 lookup	0.55	43	0.51
Piggyback queries	0.40	57	0.69

TABLE II

HIT RATIOS, COSTS, AND DELAYS FOR $T = 90$, 0.9 QUERIES PER SECOND, AND TRACE-DRIVEN CHANGES OVER 54 LOCATIONS.

Policies	Hit ratio	Cost	Delay
All hits	1	0	0
All misses	0	19	4.4
Simple lookup	0.59	7.7	1.0
Greedy age lookup	0.78	4.0	0.47
Greedy distance lookup	0.76	4.4	0.55
Median-of-3 lookup	0.71	5.4	0.68
Piggyback queries	0.59	7.7	1.0

The fact that different lookup policies perform best for different application requirements can be explained by examining the underlying delays and value deviations of the policies themselves. For example, consider the case where $A = 0.1$ and changes to the environment are driven by the lab traces. A value of $A = 0.1$ biases quality toward value deviation performance rather than delay performance. In this case, both value deviation and quality performance are best when using precise lookups and queries, as shown in Figure 11. Figure 5 therefore shows that the data quality supported by the simple lookup and piggyback query policies is superior to the data quality supported by the greedy and median-of-3 lookup policies.

Now consider the case where $A = 0.9$ and changes to the environment are again driven by the lab trace data. A value of $A = 0.9$ biases quality toward delay performance rather than value deviation performance. In this case, both delay and cost performance are best for approximate lookups and queries, as shown in Table II. Figure 13 thus shows that the query cost incurred for doing greedy age lookups or greedy distance lookups is superior to (i.e., less than) the query cost incurred by the other policies for quality that is also better.

IV. CONCLUSION

The following are the contributions of this paper:

- *Sensor network caching and lookup policies that improve data quality and query cost.* We measure the benefit

and cost of seven different caching and lookup policies as a function of the application quality requirements. We show that for some quality requirements (i.e., when delay drives data quality), policies that emulate cache hits by computing and returning approximate values for sensor data yield a simultaneous quality improvement and cost savings. This win-win is because when delay is sufficiently important, the benefit to both query cost and data quality achieved by using approximate values outweighs the negative impact on quality due to the approximation.

- *Form and magnitude of the cost vs. quality trade-off.* For our seven caching and lookup policies, five of these policies age and then delete cache entries uniformly based on an age threshold parameter, T. We observe that in many system configurations these five policies expose a linear cost vs. quality trade-off. When this linearity is present, we find that the underlying cost vs. accuracy and/or cost vs. delay functions are also linear.
- *Bounded cost for some caching and lookup policies.* For applications that require bounded resource consumption, we identify a class of policies for which the sensor field query rate can be bounded when servicing an arbitrary workload of user queries. Recall that the domain for our user queries is the set of discrete locations in the sensor field. This upper bound is a function of two variables: (1) the number of locations in each sensor field (these locations are also used to index the cache) and; (2) the age threshold parameter, T.
- *Impact of the manner in which the environment changes on query cost and data quality performance.* Our results characterize and quantify cost and quality performance for two different sensor fields, which each monitor environments with different characteristics. These results show that the form and magnitude of the cost and quality performance change, however, the performance trends generally remain the same. Specifically, the performance differences between policies change, but the policies that provide the best quality (and cost) performance in different system configurations are almost always the same.

ACKNOWLEDGMENTS

The authors would like to thank Eliot Moss and Weibo Gong for their feedback on earlier drafts of this paper. The researchers at the Intel Berkeley Research Lab also deserve thanks for sharing the sensor network traces that we used in this work.

REFERENCES

[1] A. Boulis, S. Ganeriwal, and M. B. Srivastava, "Aggregation in sensor networks: an energy-accuracy tradeoff," in *IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*. Piscataway, NJ, USA: IEEE Press, May 2003.

[2] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis, "Balancing energy efficiency and quality of aggregate data in sensor networks," *The VLDB Journal*, vol. 13, no. 4, pp. 384–403, 2004.

[3] S.-H. Son, M. Chiang, S. R. Kulkarni, and S. C. Schwartz, "The value of clustering in distributed estimation for sensor networks," in *Proceedings of the IEEE International Conference on Wireless Networks, Communications, and Mobile Computing (WirelessCom)*. Piscataway, NJ, USA: IEEE Press, June 2005.

[4] S. Tilak, N. B. Abu-Ghazaleh, and W. Heinzelman, "Infrastructure tradeoffs for sensor networks," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. New York, NY, USA: ACM Press, Sept. 2002, pp. 49–58.

[5] Y. Yu, B. Krishnamachari, and V. K. Prasanna, "Energy-latency tradeoffs for data gathering in wireless sensor networks," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*. New York, NY, USA: IEEE Communications Society, Mar. 2004.

[6] H. D. Schwetman, "CSIM 19: A powerful tool for building systems models," in *Proceedings of the Winter Simulation Conference*. New York, NY, USA: ACM Press, Dec. 2001.

[7] ——, "Introduction to process-oriented simulation and CSIM," in *Proceedings of the Winter Simulation Conference*. New York, NY, USA: ACM Press, Dec. 1990, pp. 154–157.

[8] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, and Y. Yao, "The Cougar project: a work-in-progress report," *ACM SIGMOD Record*, vol. 32, no. 4, pp. 53–59, Dec. 2003.

[9] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "The design of an acquisitional query processor for sensor networks," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM Press, 2003, pp. 491–502.

[10] ——, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on Database Systems*, vol. 30, no. 1, pp. 122–173, Mar. 2005.

[11] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan, "IrisNet: An architecture for a world-wide sensor web," *IEEE Pervasive Computing*, vol. 2, no. 4, pp. 22–33, Oct. 2003.

[12] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, Feb. 2003.

[13] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: a MAC protocol for event-driven wireless sensor networks," Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report 894, May 2003.

[14] C. Lu, B. M. Blum, T. F. Abdelzaher, J. A. Stankovic, and T. He, "RAP: a real-time communication architecture for large-scale wireless sensor networks," in *IEEE Real-Time and Embedded Technology and Applications Symposium*. Washington, DC, USA: IEEE Computer Society, Sept. 2002, pp. 55–66.

[15] C.-Y. Wan, S. B. Eisenman, and A. T. Campbell, "Coda: congestion detection and avoidance in sensor networks," in *ACM SenSys Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM Press, Nov. 2003, pp. 266–279.

[16] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *ACM SenSys Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM Press, Nov. 2003, pp. 1–13.

[17] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press, 1995.

[18] J. S. Bridle, "Probabilistic interpretation of feed-forward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing: Algorithms, Architecture and Applications*, vol. 6, 1990.

[19] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Francisco, California, USA: Morgan Kaufmann Publishers, 2000.

[20] C. Rodriguez, "A computational environment for data preprocessing in supervised classifications," Master's thesis, University of Puerto Rico, Mayaguez, July 2004.

[21] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *International Conference on Very Large Data Bases (VLDB)*. Toronto: Morgan Kaufmann, Aug. 2004, pp. 588–599.

[22] A. Jindal and K. Psounis, "Modeling spatially-correlated sensor network data," in *IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON)*. Piscataway, NJ, USA: IEEE Press, Oct. 2004, pp. 162–171.

[23] D. J. Yates, "Scalable data delivery for networked servers and wireless sensor networks," Ph.D. dissertation, Department of Computer Science, University of Massachusetts, Amherst, MA, Feb. 2006.