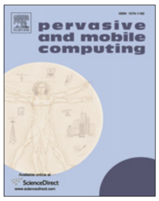
Pervasive and Mobile Computing 4 (2008) 851–870

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



Data quality and query cost in pervasive sensing systems[☆]

David J. Yates[a,*], Erich M. Nahum[b], James F. Kurose[c], Prashant Shenoy[c]

[a] Computer Information Systems Department, Bentley University, Waltham, MA 02452, USA

[b] IBM T.J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532, USA

[c] Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA

ARTICLE INFO

Article history:
Received 7 February 2008
Received in revised form 1 September 2008
Accepted 2 September 2008
Available online 19 September 2008

Keywords:
Pervasive sensing systems
Wireless sensor networks
Query processing
End-to-end delay
Data accuracy
Energy efficiency
Caching
Replication

ABSTRACT

This research is motivated by large-scale pervasive sensing applications. 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 saving. 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 single sensor field query. Finally, we demonstrate that our results are robust to the manner in which the environment being monitored changes using models for two different sensing systems.

1. Introduction

Applications for pervasive sensing systems vary in scale from monitoring and controlling microscopic manufacturing equipment, to implementing an earthquake early warning system for a country like Japan. There are many performance metrics of interest in sensing systems for such applications. We focus on two that are common to the vast majority of sensing applications:

(1) The *accuracy* of the data acquired by the application from the sensor networks; 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 pervasive sensing 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 sensing 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 [2,11,22–24,26,28].

☆ This work was supported in part by the Bentley Laboratory for Intelligent System Software and the National Science Foundation under grant EEC-0313747.
* Corresponding author. Tel.: +1 781 891 2735.
E-mail address: dyates@bentley.edu (D.J. Yates).

1574-1192/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2008.09.003

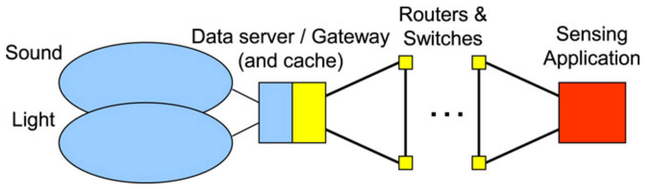


Fig. 1. Sensing system deployment.

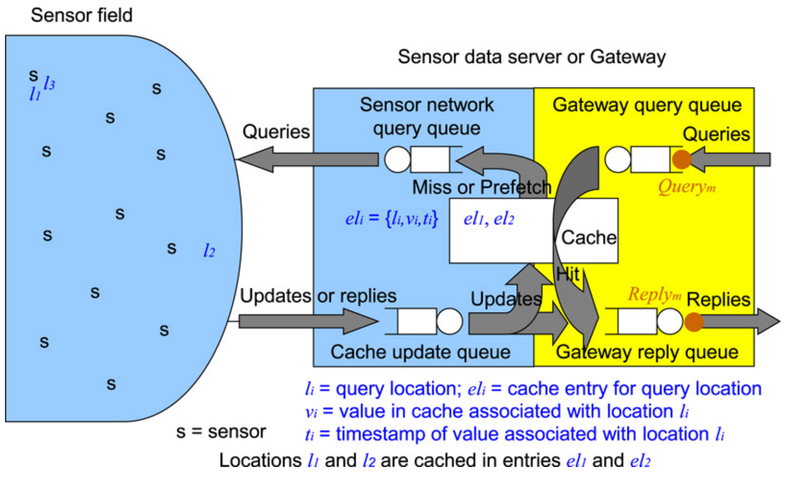

Sensor field
Sensor data server or Gateway
Sensor network query queue
Gateway query queue
Queries
Queries
Query_m
Miss or Prefetch
el_i = {l_i, v_i, t_i}
el_1, el_2
Cache
Hit
Updates or replies
Updates
Reply_m
Replies
Cache update queue
Gateway reply queue
s = sensor
l_i = query location; el_i = cache entry for query location
v_i = value in cache associated with location l_i
t_i = timestamp of value associated with location l_i
Locations l_1 and l_2 are cached in entries el_1 and el_2

Fig. 2. Sensor network data server or gateway with a cache.

To perform our research, we construct a model for a pervasive sensing system. We then develop novel policies for caching sensor network data values in sensor field gateway servers, 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 sensing system model to assess the impact of several factors on data quality and query cost 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 [20,21].

Almost all sensing system 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 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 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 [5,16]. However, the schema could also be modified while the system is running (e.g., as in IrisNet [7]). The database model might also expose a low-level interface to the sensing application. Directed Diffusion [12] does this by allowing applications to process attributes or attribute-value pairs directly. DSN [4], Tenet [8], Kairos [9], and Regiment [17] do this by introducing task-oriented or declarative programming languages for applications to acquire and process sensor network data.

1.1. Data acquisition and caching in pervasive sensing systems

Consider the impact of adding a cache to the data server or gateway in Fig. 1. Fig. 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 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 Fig. 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.

2. Cost and quality in sensing systems

2.1. 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 sensing 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 cache miss based on a lookup at the location specified in each user query. If consecutive misses occur in the cache for the same location, this policy sends redundant queries into the sensor field. When a reply is received its value is loaded into the cache, 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. This is also true for the last two policies: *greedy distance lookups* and *median-of*-3 *lookups*.
- **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,

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

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

2.2. Sensing system data quality and query cost

We normalize sensor network data quality in order to compare quality measurements from different sensing systems, 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

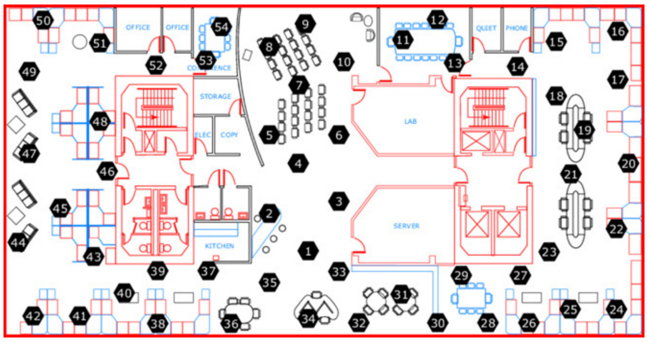


Fig. 3. Sensor field at intel berkeley research lab.

importance of delay when compared with value deviation. The expression that defines quality, denoted Q_n, is:

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

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

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

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

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, $0 \le Q_n \le 1$. This type of normalization has been used by others in neural networks; data mining for pattern recognition; and data classification [1,3,10,19].

We use two different sensing system 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 [6].

2.3. *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. Sensors always communicate with their closest base station at a cost that incorporates free-space energy loss for each transmission [18]. Thus, the properties of each one-way communication to and from location l are as follows:

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

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,

$$\text{Delay}_l = q\, r_{b'} \mid \max(\text{Delay}_l) = 1 \text{ s} \tag{6}$$

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. Thus, the minimum cost to query a location in the sensor field is normalized to 2 units (1 for the query + 1 for the reply), and the maximum delay to query a location in the sensor field is 2 s (not including queuing delay). Finally, each base station is connected to the sensor field with an access link with a capacity of 25 queries/s.

2.4. *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 Fig. 3, which is taken from [6]. Each entry in the trace is from a Mica2Dot sensor, which senses humidity, temperature, light, and

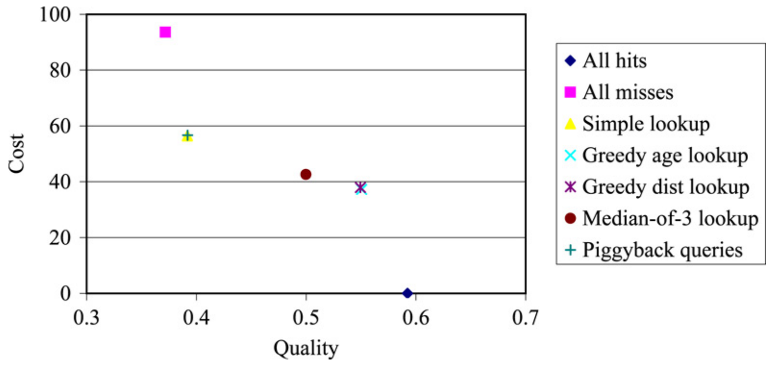


Fig. 4. Cost vs. quality for $A = 0.1$ and correlated changes over 1000 locations.

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 an average of about once every 1.33 s. 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. A value of 1 Lux corresponds to moonlight, 400 Lux to a bright office, and 100,000 Lux to full sunlight.

Four base stations are placed at the corners of the floor plan shown in Fig. 3. As before, sensors always communicate with their closest base station. We further assume that the cost and delay of each one-way communication are given by Eqs. (5) and (6), respectively.

2.5. Query workload model

We use a query workload model that is well suited for pervasive sensing 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, 13,25]. On the other hand, other researchers assume that queries either have exclusively periodic interarrival times [15,16] or random (usually exponential) interarrival times [5,29]. 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 [12,13,25]. Specifically, our query workload is characterized by two parameters:

- $\tau =$ the period of the polling component of the query workload ($\tau > 0$); and
- $\lambda =$ 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/s is used as the rate parameter to generate queries with exponentially distributed interarrival times with mean $1/\lambda$. The parameter τ is set to $111.1\overline{1}$ s so that the arrival rate for polling queries is 9 queries/s. When $\lambda = 81$ and $\tau = 111.1\overline{1}$, the aggregate arrival rate for queries is $81 + 9 = 90$ queries/s. Since the total capacity of the sensor field access links is $4 \times 25 = 100$ queries/s, 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, λ and τ are fixed for the results described in Section 3: $\lambda = 0.81$ queries/s and $\tau = 600$ s. This makes the average arrival rate for queries two orders of magnitude less than query rate for simulated changes, namely 0.9 queries/s. The total capacity of the sensor field access links is $4 \times 0.25 = 1$ query/s. Thus, the average link utilization is also 0.90 for "all miss" runs. These parameters are varied in Sections 4 and 5 as we examine performance trends in our two different sensing system models.

3. 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

$$\mu_{l,t+1} = \frac{1}{3}\mu + \frac{1}{3}\mu_{l,t} + \frac{1}{3}\mu_{N(l),t}. \tag{7}$$

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 $\mu - 6\sigma/\mu + 6\sigma$. 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

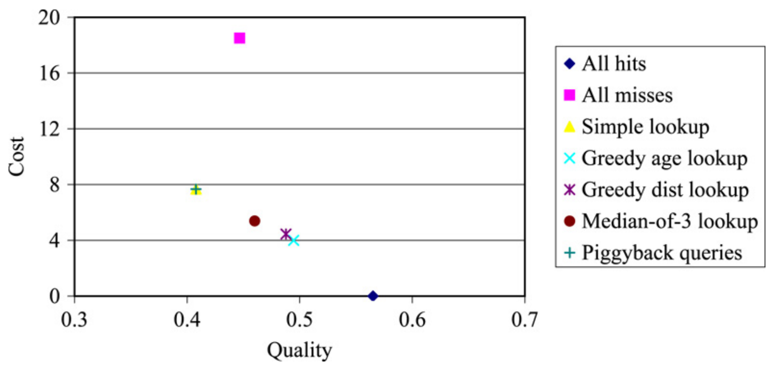


Fig. 5. Cost vs. quality for $A = 0.1$ and trace-driven changes over 54 locations.

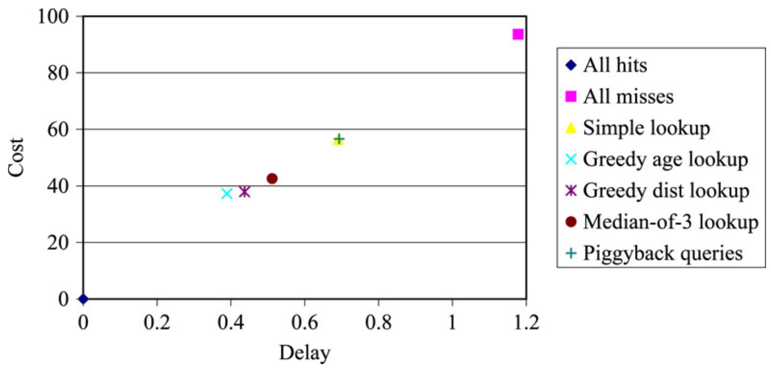


Fig. 6. Cost vs. delay for $A = 0.1$ and correlated changes over 1000 locations.

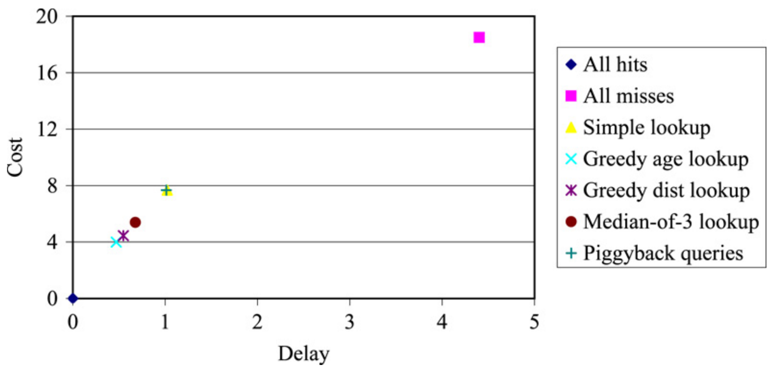


Fig. 7. Cost vs. delay for $A = 0.1$ and trace-driven changes over 54 locations.

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 [14].

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 [27]. The odd-numbered figures, Figs. 5, 7 and 9, show results for light intensity in Lux measured over time in the Intel Berkeley lab data set. These results are for $T = 90$ s, and 0.9 queries/s. Note that because of Eq. (1), the maximum sensor field query rate, $\max(R_f)$, is reduced to $54/90 = 0.6$ queries/s. The even-numbered figures, Figs. 4, 6 and 8, 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, $T = 8.8\overline{8}$ s and the average user query rate is 90 queries/s. Because of Eq. (1), the maximum sensor field query rate $\max(R_f) = 1000/T = 112.5$ queries/s.

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 Figs. 4 through 11.

1. *There is a cost vs. quality trade-off for some data quality requirements but not others.* For example, consider the results shown in Figs. 4 and 5. Fig. 5 shows cost versus quality for all seven caching and lookup policies, where $A = 0.1$ and the

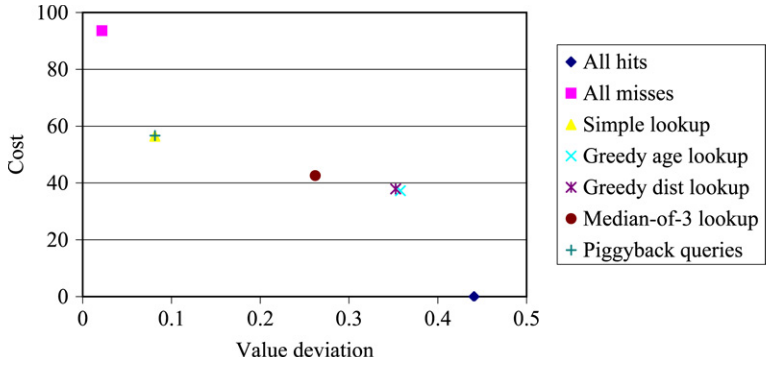


Fig. 8. Cost vs. value deviation for $A = 0.1$ and correlated changes over 1000 locations.

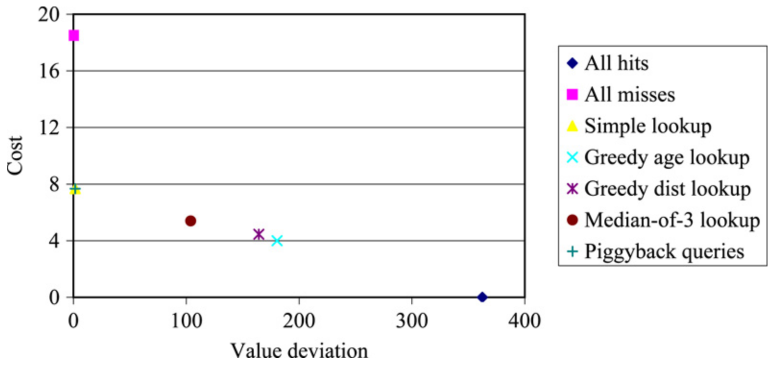


Fig. 9. Cost vs. value deviation for $A = 0.1$ and trace-driven changes over 54 locations.

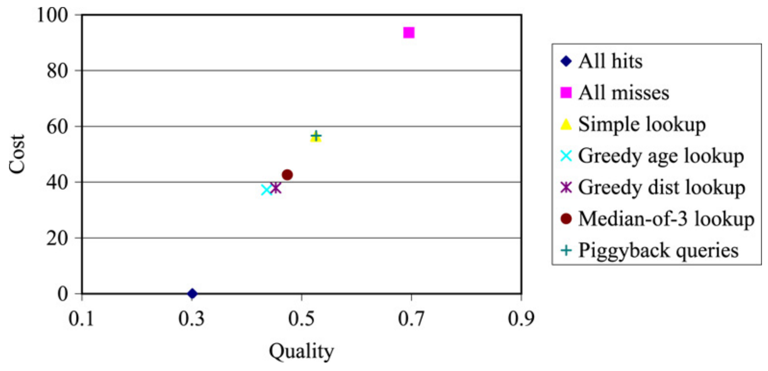


Fig. 10. Cost vs. quality for $A = 0.9$ and correlated changes over 1000 locations.

values at each location are changed according to the lab trace [6]. 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. Fig. 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 Fig. 5 are less than in Fig. 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 Figs. 4 and 5. This worse "all misses" quality is due entirely to an increase in the normalized delay term in the right hand side of Eq. (2). This can be verified by comparing the relative differences in delays between the policies, shown on the horizontal axes in Figs. 6 and 7.

We now examine system configurations for which delay is the more important component of quality in more detail. Figs. 10 and 11 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

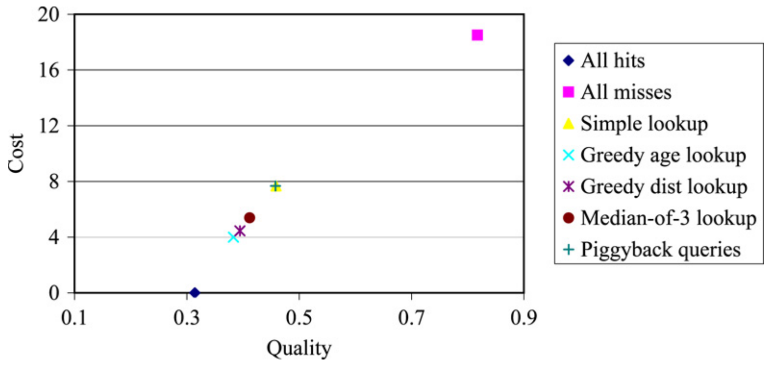


Fig. 11. Cost vs. quality for $A = 0.9$ and trace-driven changes over 54 locations.

Table 1
Hit ratios, costs, and delays for $T = 8.8\overline{8}$ s, 90 queries/s, 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 2
Hit ratios, costs, and delays for $T = 90$ s, 0.9 queries/s, 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

and lookup policies have the best cost performance and the best quality performance for both models of changing the environment in Figs. 10 and 11. 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 Figs. 4 through 11 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 1 shows the cache hit ratio for each of the (cost, delay) points in Fig. 6. Similarly, Table 2 shows the hit ratio for each of the (cost, delay) points in Fig. 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. Figs. 8 and 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 in the dispersion of the value deviations in Fig. 9 when compared with those in Fig. 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 Figs. 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

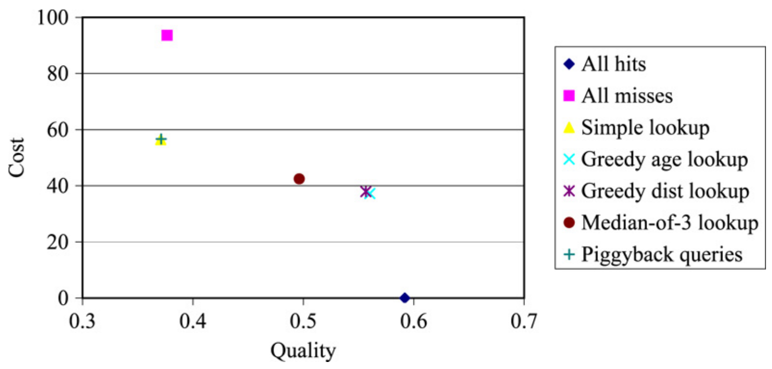


Fig. 12. Cost vs. quality for $A = 0.1$ and 9 of 1000 correlated changes/s.

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, Figs. 10 and 11 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.

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, value deviation performance is significantly better when using precise lookups and queries, as shown in Fig. 9. Fig. 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 2. Fig. 11 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. It is helpful to summarize the cost and quality performance results presented above as follows:

- When value deviation is more important to quality than delay, there is a linear cost vs. quality trade-off. We obtain the best cost performance by implementing policies that approximate sensor values by using cached values from nearby locations. The best quality performance is achieved by policies that always query and cache the sensor field location specified in the user query.
- When delay is more important than value deviation, policies that approximate values using cached values from nearby locations provide the best cost performance as well as the best quality performance.
- These results hold for both simulated changes to the environment and trace-driven changes to the environment.

4. Performance trends when value deviation is most important

The results in the previous section provide a thorough understanding of cost and quality performance in pervasive sensing systems for two models of how the environment changes. We next investigate performance trends that emerge as the query rate increases or decreases relative to the rate at which the environment changes.

Because our simulator fully models an environment with correlated changes specified by Eq. (7), we explicitly vary the rate at which the environment changes. To do this we increase the number of locations changed per second from 9 to 90 to 900 while maintaining a query workload with an average rate of 90 queries/s. The Intel Berkeley lab traces specify how the environment changes for our trace-driven experiments. In this case, we vary the relative rate at which the environment changes by decreasing the average query rate from 90 to 9 to 0.9 queries/s.

We begin this investigation by considering cost and quality performance when value deviation is more important in determining data quality than system end-to-end delay. Results for these simulations appear in this section. In the following section, we explore sensor network applications for which system delay is more important than value deviation.

The same high-level conclusions presented in the last section hold true when the relative rate at which the environment changes is increased by two orders of magnitude. However, some of the underlying results differ either qualitatively or quantitatively. Figs. 12 through 17 show these results for correlated changes to the environment. Figs. 18 through 23 show the corresponding results for trace-driven changes to the environment. If value deviation is more important than delay,

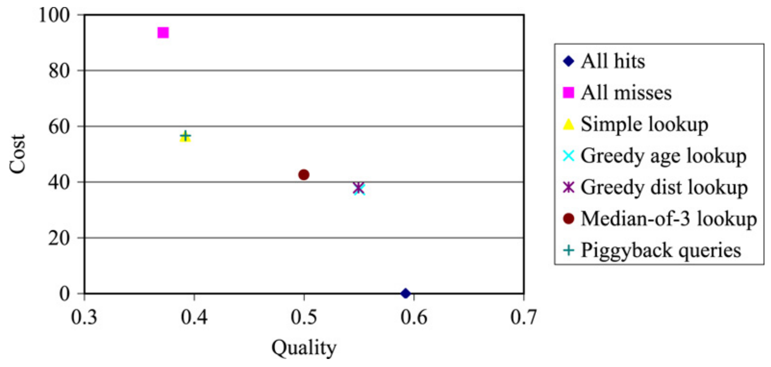


Fig. 13. Cost vs. quality for $A = 0.1$ and 90 of 1000 correlated changes/s.

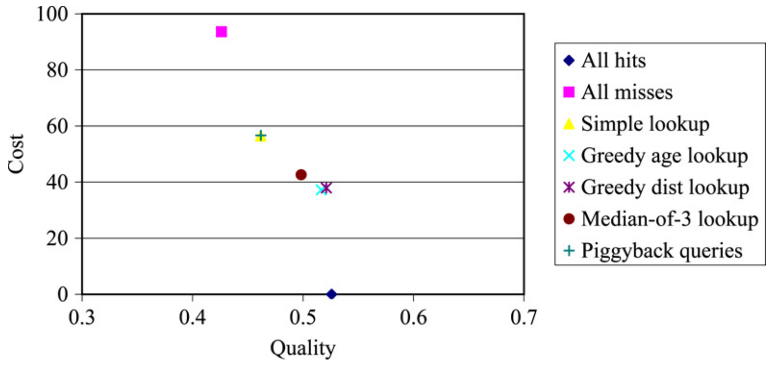


Fig. 14. Cost vs. quality for $A = 0.1$ and 900 of 1000 correlated changes/s.

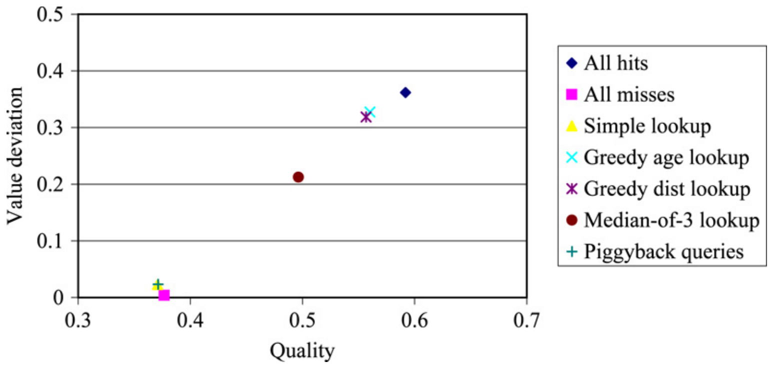


Fig. 15. Value deviation vs. quality for $A = 0.1$ and 9 of 1000 correlated changes/s.

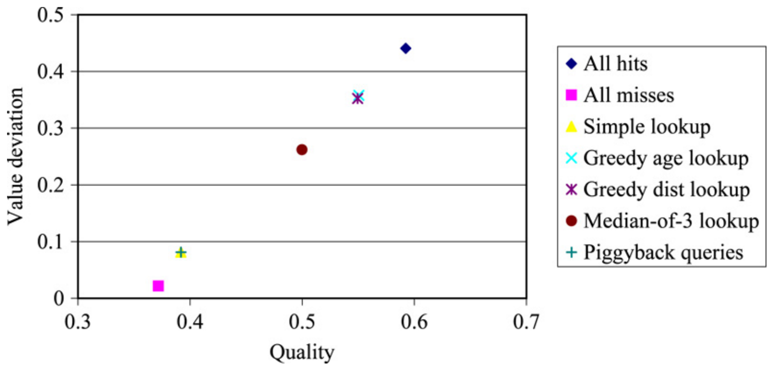


Fig. 16. Value deviation vs. quality for $A = 0.1$ and 90 of 1000 correlated changes/s.

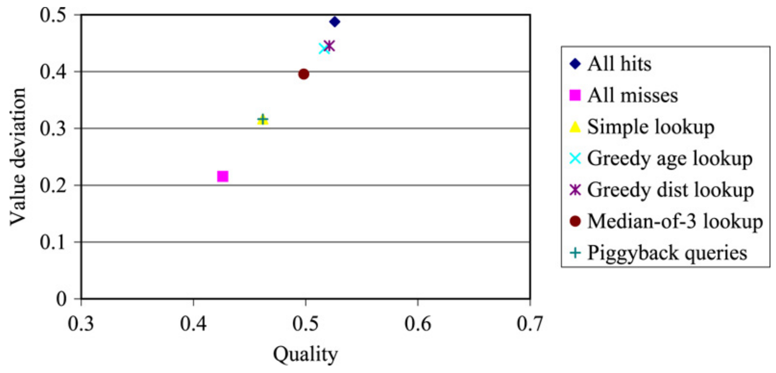


Fig. 17. Value deviation vs. quality for $A = 0.1$ and 900 of 1000 correlated changes/s.

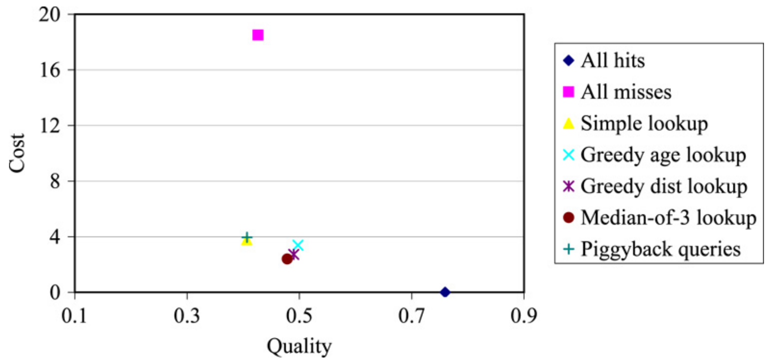


Fig. 18. Cost vs. quality for $A = 0.1$, 90 queries/s and trace-driven changes over 54 locations.

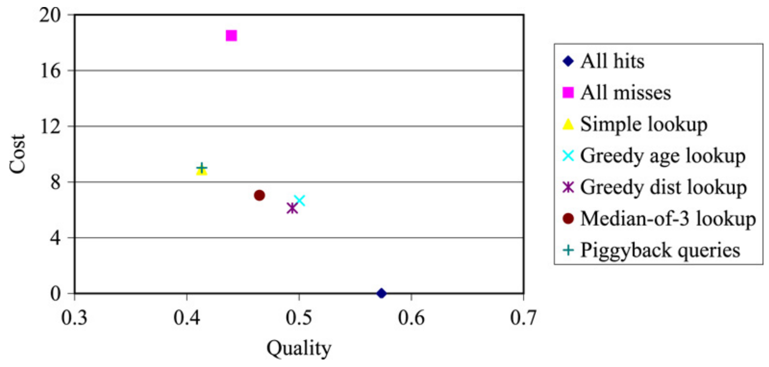


Fig. 19. Cost vs. quality for $A = 0.1$, 9 queries/s and trace-driven changes over 54 locations.

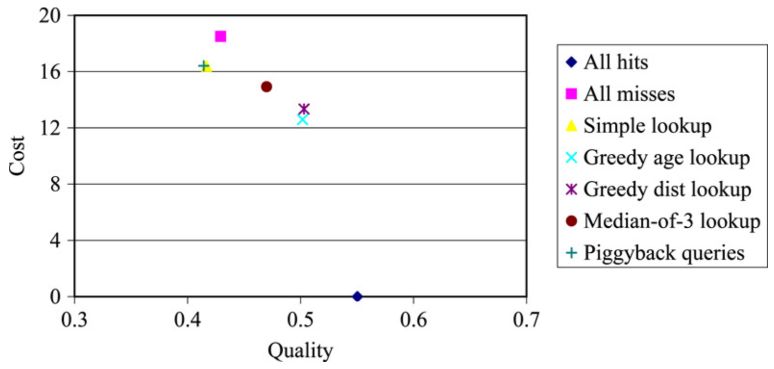


Fig. 20. Cost vs. quality for $A = 0.1$, 0.9 queries/s and trace-driven changes over 54 locations.

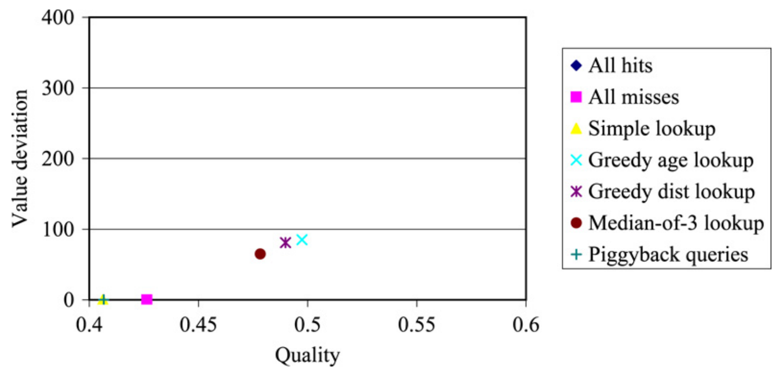


Fig. 21. Value deviation vs. quality for $A = 0.1$, 90 queries/s and trace-driven changes over 54 locations.

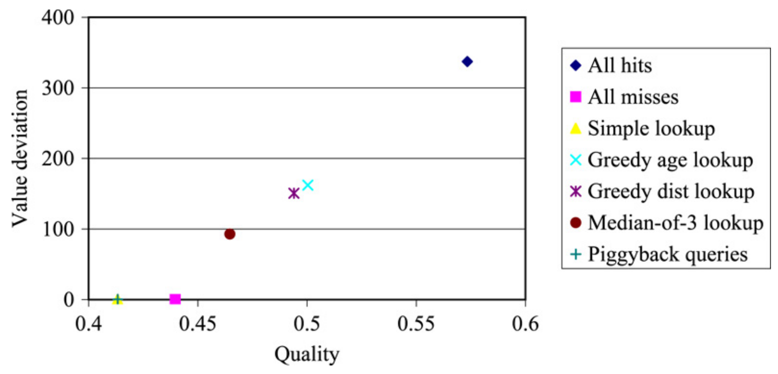


Fig. 22. Value deviation vs. quality for $A = 0.1$, 9 queries/s and trace-driven changes over 54 locations.

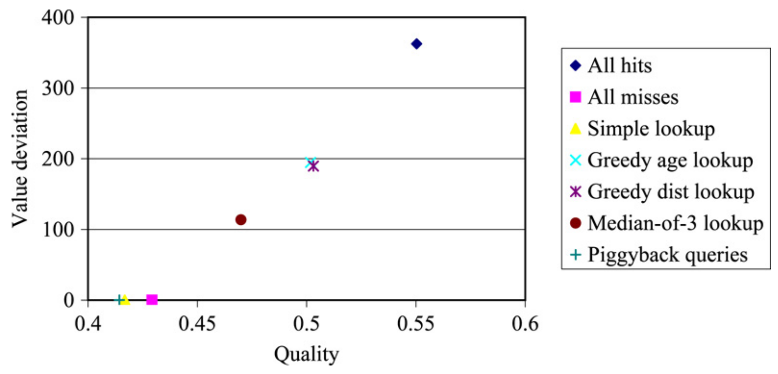


Fig. 23. Value deviation vs. quality for $A = 0.1$, 0.9 queries/s and trace-driven changes over 54 locations.

simple lookups and piggybacked queries again provide the best performance when quality is more important than cost. This can be seen in Figs. 12–14. If cost is at a premium compared with quality, greedy age lookups or greedy distance lookups trade worse quality for lower cost. In fact, these policies have the most favorable cost performance, other than the "all hits" case. These figures also show that our results are robust with respect to how rapidly our correlated environment changes. For example, the trade-off between cost and quality is linear in Figs. 12–14. Excluding the "all hits" case again, the range of quality values decreases as the rate at which the environment changes increases. In spite of these differences in quality performance, the relative cost and quality performance of most of the policies remain the same. Figs. 15–17 confirm a strong positive correlation between value deviation and quality when $A = 0.1$. Examining these figures in increasing order illustrates how value deviation increases as the rate at which the environment changes increases.

For trace-driven changes to the environment, simple lookups and piggybacked queries also provide the best performance when quality is more important than cost. This can be seen in Figs. 18–20. The trade-off between cost and quality is also linear in these figures. Although the costs increase from Fig. 18 to Fig. 19 to Fig. 20, the range of normalized quality is approximately the same (again, excluding the "all hits" case).

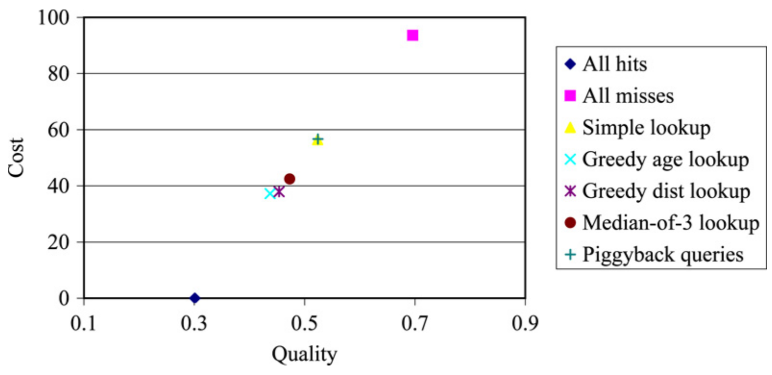


Fig. 24. Cost vs. quality for $A = 0.9$ and 9 of 1000 correlated changes/s.

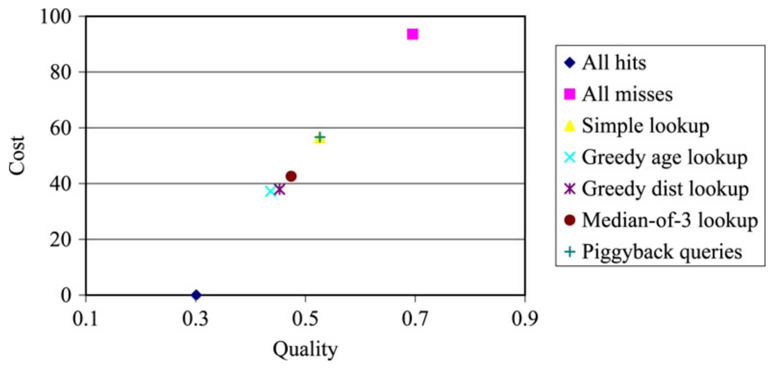


Fig. 25. Cost vs. quality for $A = 0.9$ and 90 of 1000 correlated changes/s.

Figs. 21–23 confirm a positive correlation between value deviation and quality when $A = 0.1$ and changes to the environment are driven by our trace data. Examining these figures in increasing order illustrates how value deviation increases as the average query rate (and thus the cache hit ratio) decreases.

5. Performance trends when end-to-end delay is most important

In the previous section we presented performance results that are important to sensor network applications for which high accuracy (i.e., low value deviation) is the most important factor in the quality of their data. In this section we consider how performance varies as the query rate increases or decreases relative to the rate at which the environment changes when system end-to-end delay is most important. For correlated changes to the environment, we again vary the rate at which the environment changes from 9 to 90 to 900 locations/s while maintaining a query workload with an average rate of 90 queries/s. For the Intel Berkeley lab traces, we again vary the relative rate at which the environment changes by decreasing the average query rate from 90 to 9 to 0.9 queries/s.

We examine system configurations for which delay is a more important component of quality than value deviation. Figs. 24 through 29 show such configurations for correlated changes to the environment. For correlated changes to the environment, the cost remains constant for policies as the rate of change for sensor values in the environment increases from 9 locations/s to 900 locations/s. At the same time, the quality performance for all policies is very similar, but not identical. Figs. 30 through 32 show configurations for trace-driven changes to the environment. In these system configurations, the cost increases as the average query rate and the cache hit ratio both decrease.

A value of $A = 0.9$ is used in Figs. 24 through 32. This choice of A makes system delay significantly more important to quality than value deviation. This is demonstrated, for example, by Figs. 27–29. These figures all show a strong positive and linear correlation between delay and quality for correlated changes to the environment. Examining these figures also confirms that the delays for each policy are constant across the three system configurations, and furthermore are identical to those in Table 1, since the query workloads are the same.

For correlated changes to the environment, Figs. 24–26 show that performing greedy age lookups or greedy distance lookups yields the best cost performance, and the best quality performance, excluding the "all hits" case. These figures also show that our results are robust with respect to how rapidly our correlated sensor field values change, and that the relative performance differences among our policies remain the same. Similar results hold for trace-driven changes to the environment. These can be seen in Figs. 30–32. However, it appears that as higher cache hit ratios (e.g., in Fig. 30) cause a

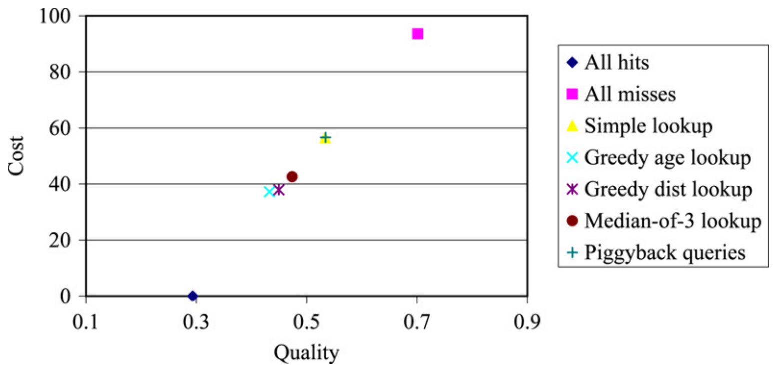


Fig. 26. Cost vs. quality for $A = 0.9$ and 900 of 1000 correlated changes/s.

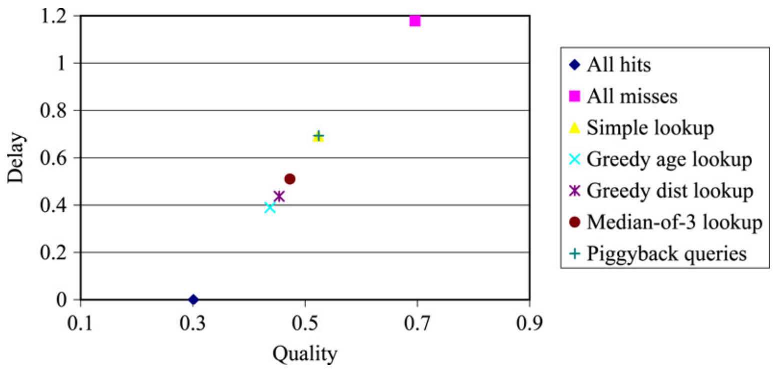


Fig. 27. Delay vs. quality for $A = 0.9$ and 9 of 1000 correlated changes/s.

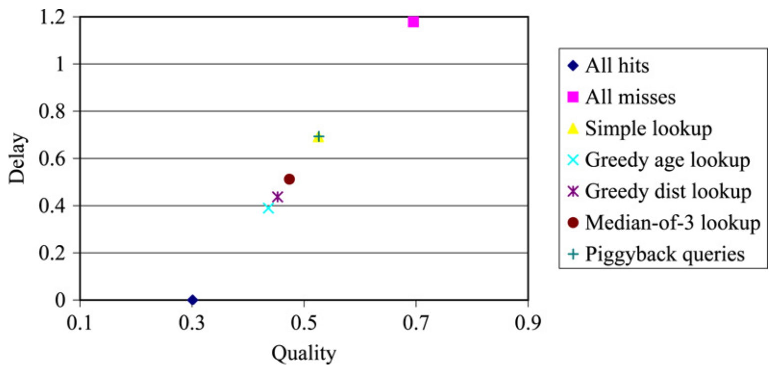


Fig. 28. Delay vs. quality for $A = 0.9$ and 90 of 1000 correlated changes/s.

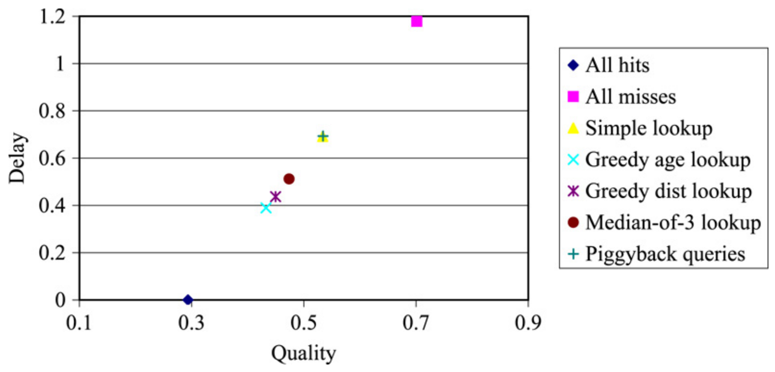


Fig. 29. Delay vs. quality for $A = 0.9$ and 900 of 1000 correlated changes/s.

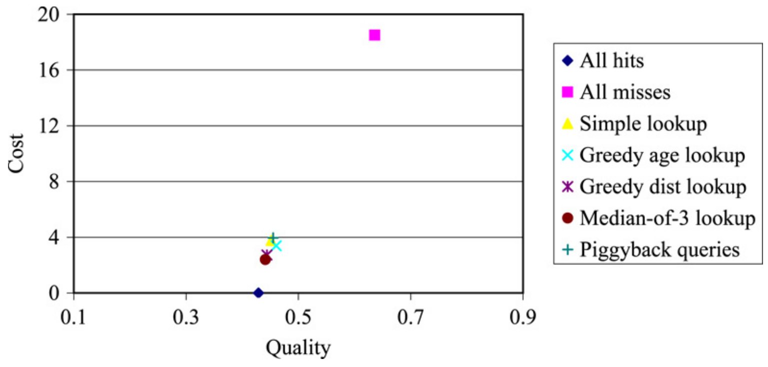


Fig. 30. Cost vs. quality for $A = 0.9$, 90 queries/s and trace-driven changes over 54 locations.

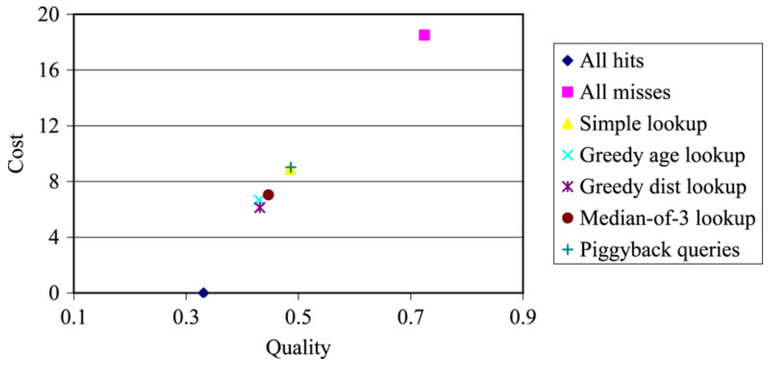


Fig. 31. Cost vs. quality for $A = 0.9$, 9 queries/s and trace-driven changes over 54 locations.

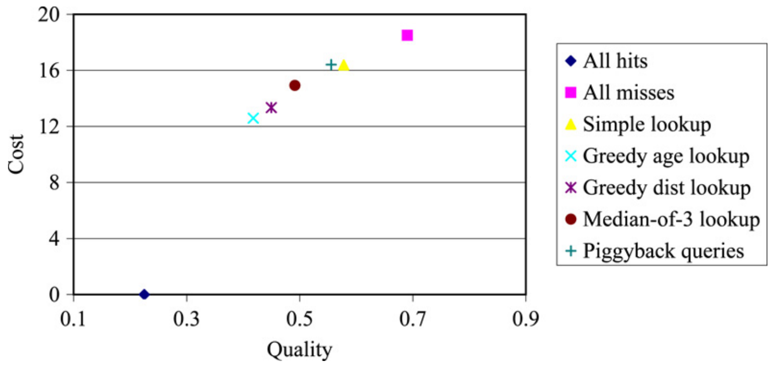


Fig. 32. Cost vs. quality for $A = 0.9$, 0.9 queries/s and trace-driven changes over 54 locations.

performance convergence among the caching policies, the median-of-3 policy exhibits better cost and quality performance than doing greedy lookups.

We have seen that for correlated changes to the environment, one or more of the greedy policies provide the best cost and quality performance among the realistic policies. For trace-driven changes to the environment, greedy policies (including the median-of-3 policy) also provide the best cost performance and the best quality performance. This can be seen in Figs. 30–32, in which there is no trade-off between cost and quality. As costs increase from Fig. 30 to Fig. 31 to Fig. 32, the range of quality values also increases. However, these quality values remain in the continuum between the “All hits” and “All misses” quality values.

6. Performance impact of cache entry age threshold (T)

In practice, picking a good value for T depends on how rapidly the environment being monitored might change, and the utility of a cached entry as it approaches being T seconds old. If T is too small, it is easy to imagine that the cache might not

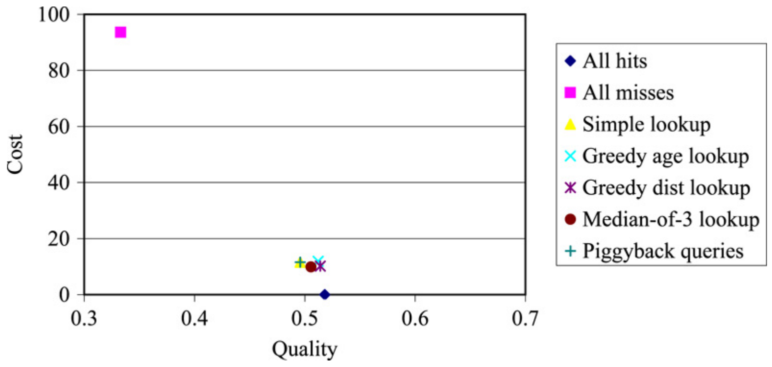


Fig. 33. Cost vs. quality for $A = 0.1$, $T = 88.8\overline{8}$ s, and correlated changes over 1000 nodes.

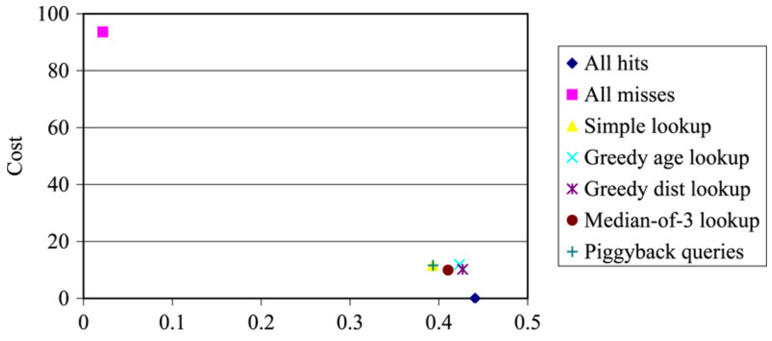


Fig. 34. Cost vs. value deviation for $A = 0.1$, $T = 88.8\overline{8}$ s, and correlated changes over 1000 nodes.

be effective. If T is too large, then the cache may be helpful in terms of saving cost, but may be harmful if value deviation increases as a function of elapsed time.

In this section we present results for values of T that are both larger and smaller than the values we have used thus far. This allows us to explore two additional operating regions for pervasive sensing systems. In the first operating region, the cache hit ratio is increased and the sensor field query rate is bounded at a rate that is much less than average query arrival rate from the sensing applications. In the second region, the cache hit ratio is significantly decreased such that for all interesting caching and lookup policies, less than 10% of queries yield a cache hit.

6.1. *Larger values of T*

Since energy is typically a critical resource in wireless sensor networks, we wanted to understand the effects of decreasing resource consumption by reducing the volume of sensor field query traffic. We accomplished this by increasing the value of the age threshold parameter associated with cache entries (our value of T), and thus boosting the cache hit ratio for our age-based policies.

For simple lookups, piggybacked queries, the median-of-3 policy, and both greedy policies, an age threshold of $T = 88.8\overline{8}$ s was used for correlated changes to the environment. This value is an order of magnitude larger than the value of T used for our earlier results. For this larger environment (with 1000 locations), the sensor field is queried an average of 90 times per second and changed at a rate of 90 locations/s. For this value of T, all of these policies except simple lookups have an upper bound on the sensor field query rate of $\max(R_f) = |\mathbf{N}|/88.8\overline{8} = 11.25$ queries/s. This bound is guaranteed by Eq. (1) and implies that the average query rate (90 queries/s) is eight times greater than the peak sensor field query rate. Our results for this larger value of T appear in Figs. 33 through 35 and Table 3. The most noticeable difference between the results in Figs. 33 through 35 when compared with our earlier results is that the performance differences between the five age-based policies are quite small. This similar performance of our age-based policies is most pronounced for performance metrics that are independent of A (e.g., cost, delay, and hit ratio). For example, Table 3 shows cost and delay performance for all non-trivial policies (from simple lookups through piggybacked queries) that is almost the same for correlated changes. Note also that corresponding hit ratios for these policies are also virtually identical. In fact, the hit ratio for all of these policies falls between 88% and 90% in this table. These hit ratios are much greater than our earlier results in which the hit ratio ranges were 40%–62% in Table 1. Even the characteristic cost vs. value deviation graph, Fig. 34, shows much less difference in value deviation than our earlier results. In all of our results for larger values of T, the much higher hit ratios (88% or greater) make the cost performance of our policies fairly close to the case where the hit ratio is 100% (i.e., the “all hits” case). Furthermore, the quality performance is close to the quality performance of the all hits case regardless of whether value deviation or delay

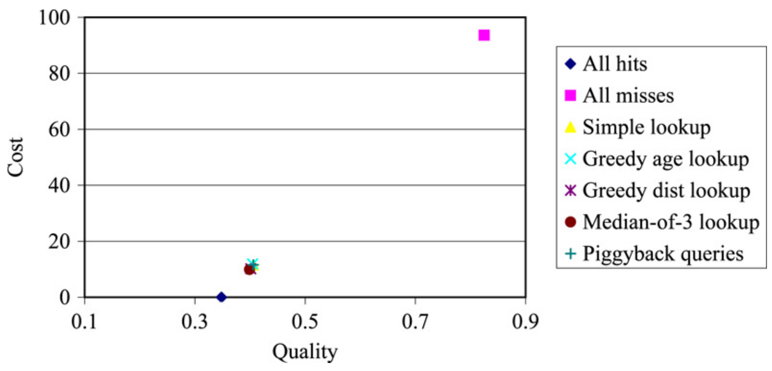


Fig. 35. Cost vs. quality for $A = 0.9$, $T = 88.\overline{88}$ s, and correlated changes over 1000 nodes.

Table 3
Hit ratios, costs, and delays for $T = 88.\overline{88}$ s, 90 queries/s, and correlated changes over 1000 locations

Policies	Hit ratio	Cost	Delay
Simple lookup	0.88	12	0.14
Greedy age lookup	0.88	12	0.13
Greedy distance lookup	0.89	10	0.13
Median-of-3 lookup	0.90	10	0.12
Piggyback queries	0.88	12	0.14

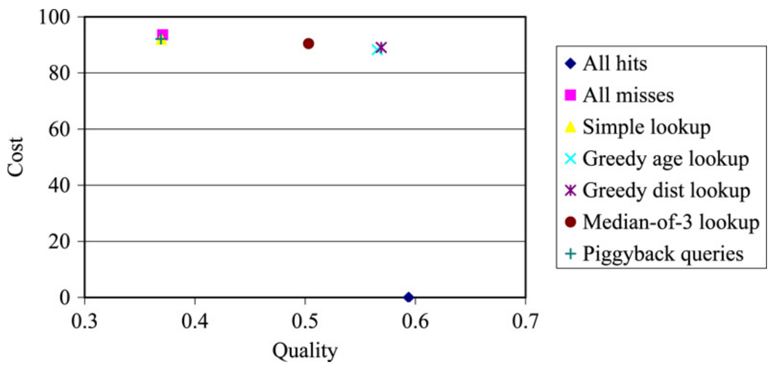


Fig. 36. Cost vs. quality for $A = 0.1$, $T = 0.\overline{88}$ s, and correlated changes over 1000 nodes.

is deemed more important to the sensing applications. Fig. 33 shows this converged quality performance when $A = 0.1$, and Fig. 35 shows analogous results when $A = 0.9$. We were surprised that in spite of a significant change in our cache entry aging parameter, some of our important earlier results still hold. These results can be summarized as follows:

- When $A = 0.9$, system delay drives quality, and there is no trade-off between cost and quality. Thus for correlated changes, Fig. 35 exhibits the same trends for cost and quality as the corresponding delays in Table 3.
- When $A = 0.1$, value deviation in the sensing system causes a trade-off between quality and cost. Specifically, Fig. 33 shows the same trends as its value deviation counterpart, Fig. 34.

6.2. Smaller values of T

In Section 6.1 we examined results for high cache hit ratios, which were induced by increasing the age threshold, T. We also wanted to study system configurations with low cache hit ratios. To obtain results for these configurations, we decreased our values of T by an order of magnitude when compared with the values of T used to produce our earlier results (see Section 3). For our five age-based policies, an age parameter of $T = 0.\overline{88}$ s was used with correlated changes to the environment. Note also that this value is two orders of magnitude smaller than the value of T used in Section 6.1, making the upper bound on the sensor field query rate 100 times larger. For the four applicable policies, $\max(R_f) = |\mathbf{N}|/0.\overline{88} = 1125$ queries/s. This bound is quite loose in that it is more than 12 times the average query rate of 90 queries/s. Results for this smaller value of T appear in this section in Figs. 36 through 38 and Table 4.

The most interesting results in Figs. 36 through 38 are the negative results. Fig. 36 shows that a cache can, for some caching and lookup policies, provide only marginal benefit when compared to having no cache (illustrated by the “All misses”

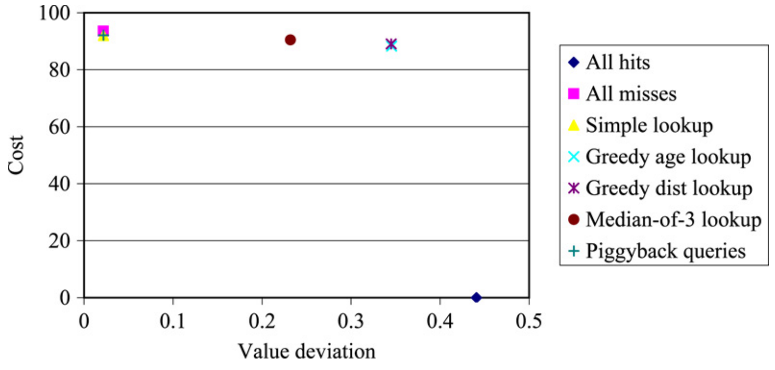


Fig. 37. Cost vs. value deviation for $A = 0.1$, $T = 0.8\overline{8}$ s, and correlated changes over 1000 nodes.

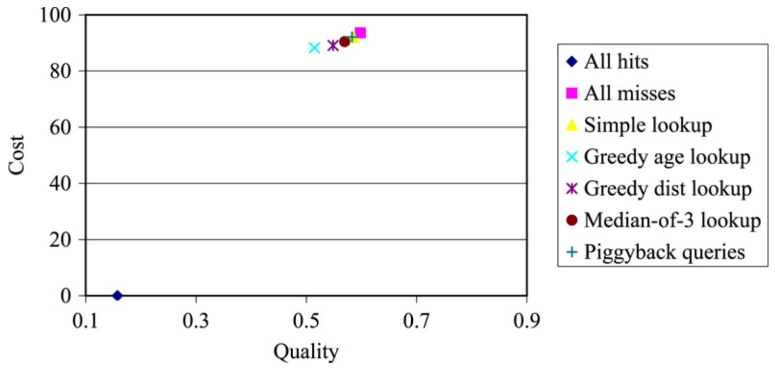


Fig. 38. Cost vs. quality for $A = 0.9$, $T = 0.8\overline{8}$ s, and correlated changes over 1000 nodes.

Table 4
Hit ratios, costs, and delays for $T = 0.8\overline{8}$ s, 90 queries/s, and correlated changes over 1000 locations

Policies	Hit ratio	Cost	Delay
Simple lookup	0.02	92	1.15
Greedy age lookup	0.08	88	0.93
Greedy distance lookup	0.06	89	1.01
Median-of-3 lookup	0.05	90	1.07
Piggyback queries	0.02	92	1.14

results). Furthermore, by using an inappropriate caching and lookup policy, the presence of a cache for sensor network data can even hurt quality performance. For example, Fig. 36 shows that the greedy lookup policies yield a cost savings of about 4% while incurring a significant penalty in quality. What renders these policies perhaps not worthwhile (because of their worse quality performance) is the significant value deviation that is introduced by a cache that is almost ineffective. These value deviations are shown in Fig. 37, in which the greedy policies and the median-of-3 lookup policy show significantly worse value deviation for a negligible or small amount of cost savings.

There are also interesting positive results shown by these figures. The cost vs. quality trade-off favors using simple lookups or piggybacked queries when value deviation drives quality (e.g., when $A = 0.1$). For correlated changes, Fig. 36 shows that using simple lookups or piggybacked queries provides about the same quality performance as all misses, while yielding a small cost savings of 1%–2%. The cost savings and quality performance are both more compelling for trace-driven changes. For this smaller environment (with 54 nodes), the cost savings increases to 12%–15% while quality also improves by about 5%. These results are not shown here, but are discussed further in [27]. Now consider the case where delay is more important than value deviation in Eq. (2). For example, when $A = 0.9$, Fig. 38 shows that the cost and quality performance of the greedy policies and the median-of-3 lookup policy is better than the performance of not using a cache. Specifically, a cost saving of up to 5% is obtained with a simultaneous improvement in quality of up to 20%. The characteristic cost and delay performance shown in Table 4 are highly correlated. Furthermore, since the cache hit ratio of the age-based policies is between 2% and 8% in this table, the quality provided by these policies is clustered toward the “All misses” point near the top of Fig. 38. However, Fig. 38 also shows that the greedy policies again provide the best performance in terms of both cost and quality. When the cache hit ratios are low, we have seen that the presence of a cache provides greater performance benefits when delay drives quality. Specifically when $A = 0.9$, we observed cost savings of up to 5% and quality improvements of up to 20%. In contrast, Figs. 36 and 37 make a weaker case for using a cache when its hit ratio is low (8% or less for all of our age-based policies, as shown in Table 4). In this section, we studied the impact of manipulating the cache hit ratio by varying the cache age threshold parameter, T:

- We achieve a significant cost saving (up to 90% in our configurations) by increasing T, and thus increasing the cache hit ratio.
- We still achieve a small cost saving (from 2% to 5%) when T is small, but greater than zero.
- In both of these cases, and for both of our sensor field models, these cost savings occur with a simultaneous quality improvement when delay is more important to quality than value deviation [27]. However, when value deviation is more important to quality, we observe quality that is sometimes worse (by up to 50% in our configurations), and sometimes better (by up to 5%).

7. 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 saving. 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. When this linearity is not present, the performance differences between our policies in terms of both cost and quality, can be small. When this is true, we observe that the cache hit ratios for our policies are close in value.
- *Bounded cost for some caching and lookup policies.* For sensing 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 sensing systems, which each monitor environments with different characteristics. These results show that while the form and magnitude of the cost and quality performance change, 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 sensing system configurations are almost always the same.
- *Effect of the age threshold parameter (T) for cache entries on performance.* For the caching policies that we propose and evaluate, the cache hit ratio for a given query workload can be increased by increasing T. The converse is also true. We determine how cost and quality performance are impacted as T is changed by two orders of magnitude. We also compare these results with "all misses" and "all hits" baseline policies. We achieve a significant cost saving (up to 90% in our configurations) by increasing T, and thus increasing the cache hit ratio. We still achieve a small cost saving (from 2% to 5%) when T is small, but greater than zero. When T is too small and value deviation is most important to quality, we observe quality that is sometimes worse, and sometimes better.

Acknowledgments

The authors would like to thank Eliot Moss, Weibo Gong, Benyuan Liu, and David Martin 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] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[2] A. Boulis, S. Ganeriwal, M.B. Srivastava, Aggregation in sensor networks: An energy-accuracy tradeoff, in: IEEE Workshop on Sensor Network Protocols and Applications, SNPA, 2003.

[3] J.S. Bridle, Probabilistic interpretation of feed-forward classification network outputs, with relationships to statistical pattern recognition, Neurocomputing: Algorithms, Architecture and Applications 6.
[4] D. Chu, L. Popa, A. Tavakoli, J.M. Hellerstein, P. Levis, S. Shenker, I. Stoica, The design and implementation of a declarative sensor network system, in: ACM Conference on Embedded Networked Sensor Systems, SenSys, 2007.
[5] A. Demers, J. Gehrke, R. Rajaraman, N. Trigoni, Y. Yao, The Cougar project: A work-in-progress report, ACM SIGMOD Record 32 (4) (2003) 53–59.
[6] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, W. Hong, Model-driven data acquisition in sensor networks, in: International Conference on Very Large Data Bases, VLDB, Morgan Kaufmann, Toronto, 2004.
[7] P.B. Gibbons, B. Karp, Y. Ke, S. Nath, S. Seshan, IrisNet: An architecture for a world-wide sensor web, IEEE Pervasive Computing 2 (4) (2003) 22–33.
[8] O. Gnawali, B. Greenstein, K.-Y. Jang, A. Joki, J. Paek, M. Vieira, D. Estrin, R. Govindan, E. Kohler, The Tenet architecture for tiered sensor networks, in: ACM Conference on Embedded Networked Sensor Systems, SenSys, 2006.
[9] R. Gummadi, O. Gnawali, R. Govindan, Macro-programming wireless sensor networks using Kairos, in: International Conference on Distributed Computing in Sensor Systems, DCOSS, 2005.
[10] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2000.
[11] W. Hu, A. Misra, R. Shorey, CAPS: Energy-efficient processing of continuous aggregate queries in sensor networks, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2006.
[12] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE/ACM Transactions on Networking 11 (1) (2003) 2–16.
[13] K. Jamieson, H. Balakrishnan, Y.C. Tay, Sift: A MAC protocol for event-driven wireless sensor networks, Technical Report 894, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA, USA, May 2003.
[14] A. Jindal, K. Psounis, Modeling spatially correlated data in sensor networks, ACM Transactions on Sensor Networks 2 (4) (2006) 466–499.
[15] C. Lu, B.M. Blum, T.F. Abdelzaher, J.A. Stankovic, T. He, RAP: A real-time communication architecture for large-scale wireless sensor networks, in: IEEE Real-Time and Embedded Technology and Applications Symposium, 2002.
[16] S. Madden, M.J. Franklin, J.M. Hellerstein, W. Hong, TinyDB: An acquisitional query processing system for sensor networks, ACM Transactions on Database Systems 30 (1) (2005) 122–173.
[17] R. Newton, G. Morrisett, M. Welsh, The Regiment macroprogramming system, in: International Conference on Information Processing in Sensor Networks, IPSN, 2007.
[18] V. Raghunathan, C. Schurgers, S. Park, M.B. Srivastava, Energy-aware wireless microsensor networks, IEEE Signal Processing Magazine 19 (2) (2002) 40–50.
[19] C. Rodriguez, A computational environment for data preprocessing in supervised classifications, Master's Thesis, University of Puerto Rico, Mayaguez, July 2004.
[20] H.D. Schwetman, Introduction to process-oriented simulation and CSIM, in: Proceedings of the ACM Winter Simulation Conference, 1990.
[21] H.D. Schwetman, CSIM 19: A powerful tool for building systems models, in: Proceedings of the ACM Winter Simulation Conference, 2001.
[22] M.A. Sharaf, J. Beaver, A. Labrinidis, P.K. Chrysanthis, Balancing energy efficiency and quality of aggregate data in sensor networks, The VLDB Journal 13 (4) (2004) 384–403.
[23] S.-H. Son, M. Chiang, S.R. Kulkarni, 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, 2005.
[24] S. Tilak, N.B. Abu-Ghazaleh, W. Heinzelman, Infrastructure tradeoffs for sensor networks, in: Proceedings of First ACM International Workshop on Wireless Sensor Networks & Applications, 2002.
[25] C.-Y. Wan, S.B. Eisenman, A.T. Campbell, Coda: Congestion detection and avoidance in sensor networks, in: ACM Conference on Embedded Networked Sensor Systems, SenSys, 2003.
[26] D.J. Yates, E. Nahum, J. Kurose, P. Shenoy, Data quality and query cost in pervasive sensing systems, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2008.
[27] D.J. Yates, Scalable data delivery for networked servers and wireless sensor networks, Ph.D. Thesis, Department of Computer Science, University of MA, Amherst, MA, February 2006.
[28] Y. Yu, B. Krishnamachari, V.K. Prasanna, Energy-latency tradeoffs for data gathering in wireless sensor networks, in: Proceedings of the Conference on Computer Communications, IEEE Infocom, 2004.
[29] J. Zhao, R. Govindan, Understanding packet delivery performance in dense wireless sensor networks, in: ACM Conference on Embedded Networked Sensor Systems, SenSys, 2003.