

# Amorphous Placement and Informed Diffusion for Timely Field Monitoring by Autonomous, Resource-Constrained, Mobile Sensors

HANY MORCOS      AZER BESTAVROS      IBRAHIM MATTA  
hmorcos@cs.bu.edu   best@cs.bu.edu   matta@cs.bu.edu  
Computer Science Department  
Boston University

October 4, 2006

## Abstract

Personal communication devices are increasingly equipped with sensors for passive monitoring of encounters and surroundings. We envision the emergence of services that enable a community of mobile users carrying such resource-limited devices to query such information at remote locations in the field in which they collectively roam. One approach to implement such a service is *directed placement and retrieval* (DPR), whereby readings/queries about a specific location are routed to a node responsible for that location. In a mobile, potentially sparse setting, where end-to-end paths are unavailable, DPR is not an attractive solution as it would require the use of delay-tolerant (flooding-based store-carry-forward) routing of both readings and queries, which is inappropriate for applications with data freshness constraints, and which is incompatible with stringent device power/memory constraints. Alternatively, we propose the use of *amorphous placement and retrieval* (APR), in which routing and field monitoring are integrated through the use of a cache management scheme coupled with an *informed* exchange of cached samples to diffuse sensory data throughout the network, in such a way that a query answer is likely to be found close to the query origin. We argue that knowledge of the distribution of query targets could be used effectively by an informed cache management policy to maximize the utility of collective storage of all devices. Using a simple analytical model, we show that the use of informed cache management is particularly important when the mobility model results in a non-uniform distribution of users over the field. We present results from extensive simulations which show that in sparsely-connected networks, APR is more cost-effective than DPR, that it provides extra resilience to node failure and packet losses, and that its use of informed cache management yields superior performance.

## 1 Introduction

**Motivation:** Advances in the manufacturing and miniaturization of sensors of various modalities are making it possible for such sensors to be embedded in mobile devices such as cellular phones, handheld computers, and automotive navigational systems.

Sensors are even expected to be embedded in future wearable computers to monitor vital signs [2, 20]. The communication capabilities of these devices open up the possibility of using a set of (possibly large number of) mobile devices in a given field as constituting a distributed repository of spatio-temporal sensory data. Thus, in this paper we consider *parasitic applications* that enable a community of mobile users carrying such devices to form ad-hoc overlay networks to query remote locations in the field in which they collectively roam – *e.g.*, allowing a spectator in a baseball game to query the number of cell-phones (which is an estimate of the number of people) at a concession stand, or allowing a traveller to query the availability and strength of public wireless networks at various airport locations. We describe our target applications as *parasitic* to delineate them from the *primary* applications of the mobile communication device. While it is conceivable to assume that such devices may have plenty of memory and (renewable) power in support of their primary functions, it is not acceptable to assume that such resources could be tapped to support the parasitic field monitoring applications we envision. Rather, it is only prudent to assume that the resources available to such applications are quite constrained – *e.g.*, the application is limited to using a small memory attached to the sensor. In this paper, and through efficient management of this limited memory, we show that the *mobility* of a set of sparsely deployed nodes could be leveraged to improve the recall rates for locally issued queries, posed over the field in which the nodes are roaming.

**Directed versus Amorphous Placement:** An important question here is related to the placement and storage of spatio-temporal samples – specifically, should each sensor node be assigned a spatiotemporal subspace for which it is responsible, or should the responsibility of the entire spatio-temporal space be shared across all nodes? We use the term “directed placement and retrieval” (DPR) to refer to the former of these approaches and the term “amorphous placement and retrieval” (APR) to refer to the latter.

DPR-like approaches have been proposed and evaluated in a number of studies in P2P networks [29, 24] as well as in sensor network (SN) settings [3, 22, 25], where they are termed as Data Centric Storage (DCS) approaches [27]. DPR simplifies query processing significantly, since a well-defined “home” for a spatio-temporal subspace makes it straightforward to route future queries over that space. In our context, using DPR, once a sample is obtained by a mobile node, storage of this sample requires its transport to the node (or locale of nodes) responsible for the spatio-temporal subspace to which this sample belongs. This could be done using any number of multi-hop ad-hoc or delay-tolerant network (DTN) routing techniques [13, 14, 21]. In a mobile, potentially sparse setting, where end-to-end paths are unavailable, DPR is *not* an attractive solution as it requires the use of flooding-based store-carry-forward routing of *both* readings and queries, which is inappropriate for “delay intolerant” field monitoring due to freshness requirements imposed on query results, and which is incompatible with the stringent constraints imposed on the use of device power and memory.<sup>1</sup>

Alternatively, in this paper, we propose the use of APR, whereby a reading is not associated with a locale where it must be stored, but rather such a reading could be stored in any one of the (and even replicated across multiple) mobile caches in the system. To improve the local view of nodes, upon meeting a new neighbor, nodes

---

<sup>1</sup>Even if memory/power are not constrained, the use of flooding would result in extensive network load and increased data dissemination delays due to (or in order to avoid) collisions, with negative implications on timely delivery of data with freshness constraints.

exchange a small number of samples. This exchange “diffuses” samples from different spots in the field to nodes that may have never visited these spots. Thus, caches are *pro-actively* managed so as to capture a local view of the field, which, as best as possible, matches the preference of users for query targets. A query issued locally at a node could then be answered directly by accessing the local cache, or at worst, by accessing the aggregate view in the caches of a small number of direct neighbors.

**Informed Cache Management:** In our setting (Section 2), each query targets a specific physical location — *e.g.*, what MAC addresses were seen at location  $X$ . Since having a sample precisely at location  $X$  is very unlikely, the query may allow for some spatial tolerance (or imprecision) by specifying a maximal distance between the observer and  $X$ .<sup>2</sup> Given the nature of the application at hand, one would expect that query targets will follow some distributional characteristics that reflect the interest in various locations in the field – an interest that may not be correlated with mobility preferences of users. For example, it may well be the case that queries preferentially target the edges of the field, whereas the mobility model of users results in a highly-skewed preference for locations near the center of the field.

The local cache at a node in the system could be seen as caching a spatiotemporal “sample” of the sensor field. By a spatiotemporal sample, we mean that each entry in a cache corresponds to a sensory data with spatial and temporal coordinates that identify both the physical location and time at which the sample was taken. Clearly, the limited cache in an embedded sensor node must be managed in a manner that maximizes its utility (Section 4). For example, upon the generation of a fresh sensory reading, a sensor node must decide whether to store this new reading in its cache memory or not, and if it decides to do so, which existing cache entry this new reading should replace.

Not only does a node have to decide what samples to replace in its local cache upon arrival of new local sensor readings, but also due to mobility, a node may find readings stored in a neighboring cache (within direct communication range) valuable as they may be samples from locations that are not well represented in the local cache. Towards that end, we argue that knowledge of the distributional characteristics of the query targets could be used effectively by an informed cache management policy to maximize the utility of collective storage of all devices. In this paper, and using a simplified mobility model, we analyze APR’s ability to achieve (say) uniform field coverage using an informed and an uninformed cache management strategy (Section 5). We show that the use of informed cache management is particularly important when the mobility model results in a non-uniform distribution of users over the field. We outline an implementation of APR in Cougar [1] (Section 6), and we confirm APR’s premise by extensive simulations (Section 8).

## 2 Definitions and Problem Statement

**Basic Assumptions:** We assume that nodes move independently and autonomously. In particular, node mobility is not driven by the need to effectively sample the field—*i.e.*, the probability of visiting a location in the field is not correlated with the probability for that location to be a query target. We assume that nodes know their locations, either

---

<sup>2</sup>Additionally, we assume that sampled sensory values (and hence query answers) must meet some recency constraints, which imply that sensory values expire.

relative or absolute. We assume that storage devoted to our “parasitic” field monitoring service is limited, necessitating the use of a cache management strategy. We also assume that nodes have unique known ID’s.

**Data Sampling:** We assume that mobile nodes sample the field according to a Poisson process. Nodes collect spatio-temporal samples, *i.e.*, each collected sample is associated with an  $(x, y)$  coordinates along with a time-stamp to indicate both the sample’s location and “age”.

**Data Freshness:** In order to be useful, returned query results should not be “stale”. Thus, we assume that a well-defined mechanism exists via which nodes are able to discard obsolete samples (*e.g.*, a time-to-live (TTL) for each sample), or otherwise assign a marginal utility to keeping one sample versus another – *i.e.*, an aging mechanism. Clearly, choosing the right parameters for aging depends on the stationarity (or time-scale of change) of the target phenomenon sampled by the sensors.

**Query Origin and Target:** While roaming the field, users may become interested in querying (or reading) the state of a remote location in the field. Such queries are submitted through the *query origin* (the node associated with the inquirer’s device). The remote location that the user is interested in reading is the *query target*. Different applications may exhibit different distributions of query origins and query targets. While the distribution of query origins may reflect the mobility model of users,<sup>3</sup> the same cannot be said about the distribution of query targets. In particular, *a priori* knowledge of the distribution of query targets could be used to *improve* the performance of the system (*e.g.*, by allowing nodes to give different weights to caching entries based on the spatial coordinates of the entries) [19]. We start with the problem when nodes are equally interested in the entire field (*i.e.*, uniform distribution for query targets). Then, we show how to generalize the solution given for the uniform case to handle query distributions that are skewed, but symmetric (*e.g.*, more query targets in the center of the field).

**Query Precision:** One particularly important parameter of queries is the tolerable inaccuracy in the result. We assume that queries target a specific location in the field along with some desirable *precision* ( $\ell$ ), which constrains how far the samples used to answer the query could be from the query target. Introducing query precision allows the support of applications in which queries might target locations in the field where no readings were collected.

**Objective and Approaches:** Assuming that the distribution of query targets is uniform suggests that the goal of the system would be to achieve uniform field coverage. This goal can be achieved using the DPR and APR approaches motivated in the last section. In the next section, we develop a formal definition of what constitutes “uniform” field coverage. Our approach for that is to use *mutual distant sampling* of the field, *i.e.*, keep samples that are as far from each other as possible. Section 3 explains the concept of mutual distant sampling, whereas Section 4 explicates how this concept fits in our design to handle both cases when query targets are uniform distribution or skewed over the field.

---

<sup>3</sup>For example, a mobility model that results in higher concentration of users in a particular part of the field will result in a higher number of queries originating from that part of the field.

### 3 Background: Mutually Distant Sampling

Teng [30] discusses the NP-hard problem of mutually distant sampling over a metric space and provides an analysis of the performance of a greedy approximation thereof.

Let  $\Gamma = (D, \|\cdot\|)$  be a metric domain, where  $\|\cdot\|$  is a non-negative measure of distance over the domain. We assume that this distance measure satisfies the triangle inequality. Given a positive integer  $k$ , then  $k$ -sampling of the domain amounts to finding a set  $S$  such that  $|S| = k$ , and  $S$  maximizes the minimum distance of its points. The minimum distance of  $S$  is defined as follows:

$$\min(S) = \min_{i \neq j} \|s_i, s_j\| \quad (1)$$

*i.e.*,  $S$  maximizes the minimum mutual distance among its samples.

**Greedy Approximation:** Teng [30] proves that the greedy algorithm sketched below provides a 0.5-approximation to the problem, *i.e.*,  $\min_{i \neq j} \|s_i, s_j\| \geq 0.5 \times \min_{i \neq j} \|t_i, t_j\|$ , where  $s_i, s_j \in S$ ,  $t_i, t_j \in T$ , where  $T$  is the optimal solution, and  $S$  is the set returned by the greedy algorithm.

**Algorithm 1** *Input:*  $\Gamma = (D, \|\cdot\|)$ , an integer  $k \geq 2$ .

1. Start with a random point  $x \in D$ . Let  $S = \{x\}$ .
2. For  $j = 2$  to  $k$ , repeat the following:
  - 2.1) Select the point  $y \in D$  such that  $y$  maximizes the following function

$$\psi(S, y) = \min_{q \in S} \|q, y\|$$

$\psi(S, y)$  defines the distance between a set  $S$  and a point  $y$  using the measure  $\|\cdot\|$ , as the minimum distance between  $y$  and all points  $q \in S$ .

- 2.2) Set  $S = S \cup \{y\}$

3. Return  $S$ .

When  $D$  is a set of points  $\{p_1, p_2, \dots, p_n\} \in \mathbb{R}^d$ , the complexity of this greedy algorithm is  $O(k^2n)$ . For the special case in which  $n = k + 1$ , we can find the optimum solution in  $O(k^2)$  using the following algorithm.

**Algorithm 2** *Input:*  $D = \{p_1, p_2, \dots, p_n\} \in \mathbb{R}^d$ , an integer  $k$ , such that  $n = k + 1$ .

1. For  $i = 2$  to  $n$ : associate with each point  $p_i$  the value  $d_i = \psi(D \setminus \{p_i\}, p_i)$   
 $d_i$  is the distance between point  $p_i$  and its closet neighbor.
2. Select the two points  $p_x, p_y$  whose inter-distance is minimal in all  $d_i, \forall 1 \leq i \leq n$ .  
For  $p_x$ , calculate the distance  $D_x = \psi(D \setminus \{p_x, p_y\}, p_x)$ . For  $p_y$ , calculate the distance  $D_y = \psi(D \setminus \{p_x, p_y\}, p_y)$ .
3. If  $(D_x < D_y)$ , let  $S = D \setminus \{p_x\}$ , otherwise, let  $S = D \setminus \{p_y\}$ .
4. Return  $S$ .

## 4 Amorphous Placement and Retrieval (APR)

APR is a simple scalable algorithm that employs mobility, as opposed to multi-hop communication, whenever possible, to diffuse field samples from field locations to nodes that have never visited these locations. Hence, it improves the local view of each node of the whole field and enables nodes to answer more queries locally saving communication overhead. Avoiding multi-hop communication (as in ad-hoc routing techniques [13, 21]) makes APR efficient in terms of energy consumption and more robust in case of node failures or packet losses, moreover, it saves the overhead of operating an ad-hoc routing protocol. We also show that, interestingly, under limited node mobility, APR results in an informed multi-hop diffusion of field readings (akin to a selective delay-tolerant multi-hop forwarding of these readings).

APR has two main components: (1) sample diffusion, and (2) cache management. Both components work together to enable nodes to optimize the contents of their caches resulting in better matching between the distribution of query targets and locally/nearby cached samples.

### 4.1 Sample Diffusion

The set of samples that are locally cached by a node (*i.e.*, the node’s view of the whole field) is a subset of the set of samples this node collects while moving in the field. The latter set is totally defined by the mobility model of nodes, since nodes sample the field along their movement trajectory. However, the workload presented to any node (*i.e.*, targets of queries posed to this node) is independent of the node’s trajectory. Hence, we need a mechanism to “decouple” each node’s view of the field from its movement trajectory. This mechanism relies on sample diffusion, whereby upon encountering each other, nodes exchange a small number of samples. This amounts to diversifying the contents of each cache, allowing improved matching of the nodes’ local view of the whole field to the query distribution at both nodes.

More specifically, a node  $z$  declares its presence to its neighbors by broadcasting a short `Hello` packet every  $\alpha$  seconds. `Hello` packets contain a compact summary of the cache of  $z$  (using a Bloom filter). Upon receipt of such a packet, a neighbor  $y$  replies with an `Exchange` packet: a packet containing  $k$  of its samples that failed the Bloom filter test (*i.e.*, node  $z$  does not have similar samples and hence its local view of the field would improve by getting these samples). The `Exchange` packet, likewise, contains a compact summary of  $y$ ’s cache. Upon receiving an `Exchange` packet,  $z$  adds the received samples to its cache applying the QCCM cache management algorithm described below, if needed. Then it replies to  $y$  with a similar packet containing  $k$  of its own samples.

Some parameters decisively affect the performance of the sample diffusion process. The first parameter is  $\alpha$ , the rate of sample diffusion. Slow sample diffusion rates may not specifically help diversifying the contents of caches, resulting in poor performance. While, high diffusion rates may cost too much communication power. The other parameter is the diffusion size  $k$ . Too small of a value may not be enough to improve performance, while a very large value means more energy consumption. These parameters need to be carefully tuned to optimize the performance of APR.

These parameters need to be carefully tuned to optimize the performance of APR. We experiment with the effect of these parameters later in the Section 8.

## 4.2 Query-Cognizant Cache Management

Since queries presented to each node have a certain target distribution that is independent of the movement trajectory of this node, it is best to manage the node’s cache in a way that makes it store a representation that mirrors this distribution—*e.g.*, when query targets are uniformly distributed, the cache management should strive to cover the whole field as uniformly as possible. To achieve this goal we propose Query-Cognizant Cache Management (QCCM) policy. QCCM is based on maximizing the mutual distance between samples, as explained in Section 3. Whenever the cache is full and there are more than one sample to be added to the cache (due to sample diffusion (Subsection 4.1)), Algorithm 1 is applied to determine which set of samples should be retained in the cache. Each time the cache is full and there is one more sample to add (due to the acquisition of a new field sample), Algorithm 2 is applied to determine which sample is to be evicted (if any) in favor of the newly acquired samples.

Notice that, in case the query model doesn’t follow a uniform distribution over the field, we can always apply a linear transformation on samples to get the effect of “stretching” (or scaling) the field at the area of high interest. Such that applying the QCCM algorithm will not evict samples from the area of high interest in favor of another samples covering a less important area. The exact form of this transformation depends on the exact distribution of query targets over the field. Notice that for smooth and continuous distributions (*e.g.*, a bivariate normal distribution), one only needs to apply a simple transformation over the distances between samples before applying the mutually-distant sampling algorithms.

As a proof of concept, we consider the case when queries follow a bivariate normal distribution whose mean  $\mu$  is the center of the field (*i.e.*,  $\mu = (L/2, L/2)$ ). Variance of this distribution defines the skewness of queries. Small variance means that queries are mostly concentrated in a small area in the center of the field, while larger values of variance makes the query access pattern approaches uniform. The transformation we suggest here is adequate for all 2D distributions that are symmetric in all directions (*e.g.*, bivariate normal with a symmetric covariance matrix). The stretching transformation is given in Algorithm 3.

**The Stretching Algorithm:** To be applied on samples’ coordinates before feeding them to QCCM, when query targets follow a smooth symmetrical distribution over the field.

**Algorithm 3** *Input:* Two dimensional bivariate normal symmetric distribution  $Q$  with mean  $\mu$  and symmetric covariance matrix  $\Sigma$ , such that the diagonal elements of  $\Sigma$  equal  $\sigma^2$ . Field dimensions  $L \times L$ , and A set of samples  $S = \{s_1, s_2, \dots, s_m\}$  every sample  $s_i$  has locations  $(x_i, y_i)$ .

*Output:* A set of “mapped” samples  $S' = \{s'_1, s'_2, \dots, s'_m\}$ , where every sample  $s'_i$  is the image of sample  $s_i$ , and the mapped location is  $(x'_i, y'_i)$

1. Let  $h$  be the value of the probability density function of a uniform distribution over the field.  $h = 1/L^2$

2. For  $i = 1$  to  $m$ :
  - 1) Consider the section of distribution  $Q$  defined by the plane going through  $\mu$  and location of sample  $s_i$  which is  $(x_i, y_i)$ . Call this section  $Q'$ .  $Q'$  will be a 1-D normal distribution with mean =  $L/2$  and variance  $\sigma$ .
  - 2) Sum the probability density in  $Q'$  between the mean of  $Q'$  and  $(x_i, y_i)$ , call this amount  $p$ .
  - 3) Sample  $s_i$  will be moved along the line connecting it to  $\mu$ . Let  $d'$  denote the new distance between  $s'$  and  $\mu$ .
  - 4) To calculate  $d'$  divide the probability mass  $p$  by  $h$  (i.e.,  $d' = p/h$ ).
  - 5) To calculate the coordinates of the mapped sample  $s'$ , map  $(x_i, y_i)$  to the corresponding polar coordinates  $(d, \theta_i)$ , where the origin is taken to be  $\mu$ . Then the polar coordinates of  $s'_i$  are  $(d', \theta_i)$ .
3. Return  $S'$ .

Figure 1 shows steps 1.2 through 1.4 in Algorithm 3. It shows how to find the distance between  $d'$ , the image of  $d$  and the center of the distribution.

Notice that to find distribution  $Q'$  in step 2.1, we can take any section in  $Q$  passing through the mean  $\mu$ , since  $Q$  is assumed to be symmetric in all directions. Hence  $Q'$  could be calculated once off-line. Notice also, that,  $p$  in step 2.2 can be calculated using the cumulative distribution function (CDF) of  $Q'$ .

Algorithm 3 uses a linear transformation to adjust the distance between every sample and the mean of the distribution so that the mapped distribution of points matches, as much as possible, the uniform distribution. This transformation has a similar effect of “stretching” the bivariate normal distribution so that it matches the normal distribution. The net effect on samples locations is that: samples that fall in areas of high-demand (i.e., where the query distribution has high values) are moved further from the center of the distribution, that is, distances between points from such areas get *extended*. However, points from less-popular areas are moved closer to the center of the distribution, resulting in the distances between such points getting shrunk, making them more favorable for eviction by QCCM.

The mapped coordinates resulting from the Stretching Algorithm are fed to QCCM to find eviction candidates. Notice that the only prerequisite to applying the Stretching Algorithm is that the query distribution follows any symmetric distribution. It is not required that this distribution should be centered at the center of the field. To see the reason, consider, for example, the case when this distribution is centered closer to one of the corners (let’s denote this corner by  $g$ ). Then, mapping points that are between the center of the distribution and  $g$  would result in moving these points further from the center of the distribution (since they lie in high-demand area). This could move points “out of the field”, since they were originally close to the corner. However, the functionality of QCCM does not depend on the absolute coordinates of the points, rather it depends on the distances between points. Hence, if the Stretching Algorithm, ended up assigning coordinates that are out of the field bounds to some points, QCCM will still function correctly producing the sought result; evicting samples from areas of low-demand in favor of points from areas of high-demand.

In the remainder of this paper, unless otherwise noted, we will assume that query targets are uniform over the field.

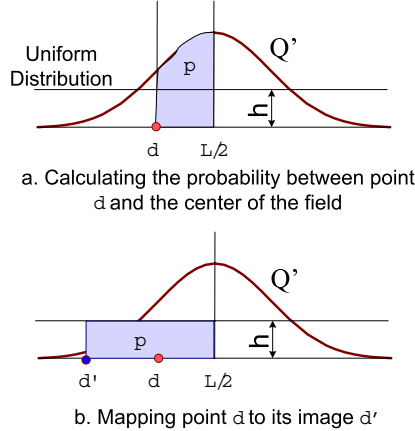


Figure 1: Application of Algorithm 3 (a) calculating the probability between point location  $d$  and the center of the distribution, (b) find the distance between  $d'$ , the image of  $d$ , and the center of the field.

## 5 Effect of Cache Management

In this section we develop a simple model to gain insight into how much the cache management of multiple mobile nodes affects their *collective* probability of success in answering queries. We assume that  $n$  mobile nodes roam in a two-dimensional periodic field (*i.e.*, torus) of size  $L \times L$ . Each node has a cache of size  $c$ , where  $L^2 \gg c$ . Nodes are given enough time  $T$  to sample the entire field<sup>4</sup>. Nodes answer queries uniformly distributed over the field and of precision  $\ell$ , where we use  $L_1$  (*i.e.*, Manhattan) distance. The mobile nodes move according to some mobility model, and they sample the field along their movement trajectory, applying a cache management algorithm whenever needed. We model any mobility model through a probability distribution  $p_{ij}, \forall (i, j) \in [L, L]$ , where  $p_{ij}$  is the steady-state probability of any node being in field location  $(i, j)$  under that mobility model. A “uniform” mobility model assigns the same probabilities to all field locations, while a “biased” model assigns different probabilities to different field locations (*e.g.*, a random waypoint mobility model results in a higher probability of being in the center of the field). Table 1 lists the parameters of the analysis. To be amenable to analysis, we assume that any collected sample stays fresh, and so a returned answer is always fresh. This assumption is reasonable if the rate of query/response is much larger than the rate of change in the sampled phenomenon. We relax this assumption in Section 8.

The goal of the model is to compare two cache management algorithms: QCCM, and random cache management (RCM) at steady state—we say that the system reached steady state when all nodes have sampled every location in the field. RCM randomly selects a sample to be evicted, whenever needed. Thus under RCM, we know that the set of samples retained in the cache of any node would follow the distribution

<sup>4</sup>Under a random walk, that time is  $T = O(L^2 \log L)$  according to [34]. This is an accurate estimation for  $L^2 \approx 25$ .

$L$	field side (square field $L \times L$ )
$c$	cache size
$\ell$	query precision
$n$	number of nodes in the field
$p_{ij}$	mobility model: steady state probability of being in location $(i, j)$

Table 1: Parameters of the analysis

induced by the mobility model,  $p_{ij}$ . On the other hand, QCCM decouples the set of samples retained in the cache from the underlying mobility model and tries to keep samples that provides optimum query answering. To focus on the efficiency of the cache management algorithm, we assume that nodes flood the field with their queries, so that cache management decisions done at one node affect the probability of success of queries issued at other nodes. We now introduce two lemmas to help us calculate coverage by each cache management algorithm, which, under the uniform query model assumption, is indicative of the query success ratio.

**Coverage of a single sample:** Assume a node keeps a sample  $e$  at location  $(x, y)$ , then the coverage of the field attained by keeping  $e$  is a function of  $\ell$ , the query precision. The following Lemma defines coverage of a single sample  $R(\ell)$ .

**Lemma 1** *Let  $\ell$  denote the query precision. Then, in a two-dimensional periodic field, and using the  $L_1$  distance measure, field coverage attained by keeping any sample (assuming no overlap with coverage from other samples) can be calculated by  $R(\ell) = \sum_{i=1}^{\ell} 4i + 1 = 2\ell(\ell + 1) + 1$*

*Proof:* It suffices to notice that on an  $L \times L$  torus, the number of neighboring locations at distance exactly  $\ell$  from any location equals exactly  $4\ell$ , and we add 1, to account for coverage of the field location where the sample lies. ■

**Optimal Inter-Sampling Spacing (ISS) in 2D torus:** We need to answer the question: how can we place  $c$  points on a torus of dimensions  $L \times L$ , such that the minimum mutual distance between any two points is maximized, and what would the optimum distance  $S_{opt}$  in this case. Let's assume for now that  $c$  is a square number, i.e.,  $c = s^2$  for some integer  $s < L$ , and  $L$  is a multiple of  $s$ . Then we can easily argue that placing the  $c$  points uniformly on the field maximizes their mutual minimum distance. In such a case, an optimal algorithm would be one that divides the torus into  $s \times s$  squares, then places a point in each square. Selecting the corresponding points in each square yields a minimum ISS of  $S_{opt} = L/s$ . The following lemma formalizes this fact.

**Lemma 2** *In an  $L \times L$  torus and given that  $c = s^2$ , if  $L > s$  and  $L$  is a multiple of  $s$ , then  $S_{opt} \geq \frac{L}{s}$ .*

**Performance of QCCM:** As we discussed above, at steady state, nodes would have sampled the entire field. Recall that QCCM decouples the cache content from the movement trajectory of the nodes. Then we assume that nodes are able to maximize inter-sample spacing, yielding  $ISS = S_{opt} = L/s$ . This is always true as long as the mobility model has nonzero probability of visiting all field locations. Since there are  $n$  nodes, we know that given any area  $A$  of size  $= S_{opt} \times S_{opt}$ ,  $A$  will host exactly

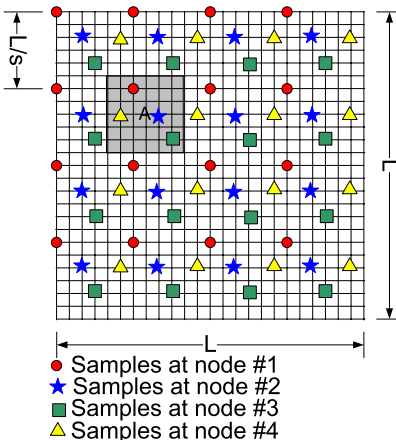


Figure 2: Idealized field coverage by four nodes applying QCCM. Notice that any area  $A = S_{opt} \times S_{opt}$  will have exactly one sample from each node.

one sample from each node, for a total of  $n$  samples in  $A$ . Coverage of  $A$ , in this case, corresponds to coverage of the whole field, since the coverage pattern in  $A$  is repeated over the rest of the field.

To simplify the analysis, we assume that nodes do not optimize their caches with respect to contents in their neighbors' caches (*i.e.*, nodes do not try to minimize the intersection of coverage achieved by samples in their caches and coverage of samples in their neighbors' caches). Under this assumption, it follows that  $A$  has  $n$  randomly placed samples. Figure 2 illustrates this setup. Now, consider any field location  $(l_{ij})$  in  $A$ , the probability ( $q = \Pr[l_{ij} \text{ covered}]$ ) of covering this location is proportional to the value of  $\ell$ , and can be calculated as  $q = \frac{R(\ell)}{S_{opt} \times S_{opt}}$ . This follows from the fact that coverage of any field location is related to coverage of one sample. For example, if  $\ell = 0$ ,  $l_{ij}$  has only one chance of being covered (*i.e.*, having a sample at  $l_{ij}$ ). If  $\ell = 1$ , then  $l_{ij}$  has five chances (having a sample at location  $l_{ij}$  itself, or having a sample at any of its four neighboring locations), and so on. Now we can view the attempt to cover any field location in  $A$  by the  $n$  samples, as  $n$  independent Bernoulli trials, each with probability of success  $q$ . Thus, the probability of covering any field location exactly  $x$  times (*i.e.*, probability  $l_{ij}$  will fall into the coverage area of  $x$  different samples) has a binomial distribution and is given by  $\Pr[B(n, q) = x]$ , where  $B(n, q)$  is the Binomial probability. By running a summation of the last quantity for  $x = 1 \dots n$ , we can obtain the probability of success under QCCM as:

$$Success_{QCCM} = \sum_{x=1}^n \binom{n}{x} q^x (1-q)^{n-x} = 1 - (1-q)^n \quad (2)$$

**Performance of RCM:** Under RCM, nodes sample the underlying mobility model, hence their cache content will match this distribution. Following the same lines of analysis as we did in QCCM, we have  $n$  nodes, each with cache  $c$ , for a total of  $n \times c$  samples in the field. For a given value of  $\ell$ , let's define  $\mathcal{N}_{ij}$  as the set of neighboring locations of  $l_{ij}$  within  $\ell$  distance units. The probability  $\omega_{ij}$  of covering a location  $l_{ij}$

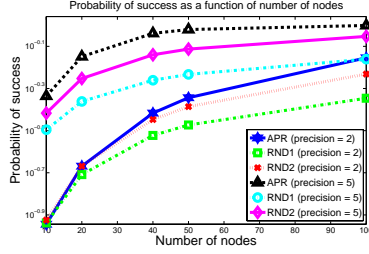


Figure 3: Effect of cache management algorithm on query success ratio for  $n$  mobile nodes. Notice RND2, RCM under mobility model 2 (Figure 4 right), is better than RND1, RCM under mobility model 1 (Figure 4 left), since the earlier is less skewed than the latter.

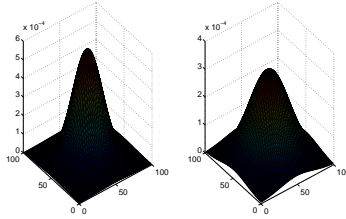


Figure 4:  $p_{ij}$  for the two mobility models used in Figure 3

can be calculated as:  $\omega_{ij} = p_{ij} + \sum_{l_{xy} \in \mathcal{N}_{ij}} p_{xy}$

Hence the probability of  $l_{ij}$  being covered exactly  $x$  times is given by:  $\Pr[B(n \times c, \omega_{ij}) = x]$ , and the expected number of locations that are covered exactly  $x$  times is given by:  $\sum_{0 \leq i, j \leq L} \Pr[B(nc, \omega_{ij}) = x]$ . Then, we can calculate coverage of the field by running a summation for all  $x = 1 \cdots n \times c$ , and the success probability is given by:

$$Success_{RCM} = \frac{1}{L^2} \sum_{x=1}^{nc} \binom{nc}{x} \left[ \sum_{0 \leq i, j \leq L} \omega_{ij}^x (1 - \omega_{ij})^{nc-x} \right] \quad (3)$$

Figure 3 plots Equations 2 and 3 for two different mobility models depicted in Figure 4. We have numerically confirmed that both mobility models have no  $i, j$  such that  $p(i, j) = 0$ , *i.e.*, nodes can sample the entire field under both models. It is clear that QCCM has a noticeable performance advantage over RCM, as it manages cache content based on the workload, decoupling it from the trajectory of motion of nodes.

## 6 Implementation Details

In this section, we sketch how an APR-like system may be implemented over Cougar, a publicly-available distributed multi-agent architecture [1]. Cougar includes components to support agent-to-agent messaging, naming, mobility, blackboards, external

user interface, and additional “pluggable” capabilities. Plugins are added software components which are loaded into agents to define their behavior. The Cougaar Component Model allows developers to configure Cougaar to match both their domain and system requirements / constraints. Cougaar is Java-based which enables it to run on any platform that supports Java.

To instantiate an application within the Cougaar architecture, one has to define plugins, message transport system (MTS), blackboard to define interaction between agents, and a Cougaar community to enable interaction between different agents (see figure 5).

We define an APR agent within the Cougaar architecture using the following plugins:

- (1) *Cache management plugin*: responsible for managing the cache space, adding samples and finding samples to diffuse when meeting other neighbors.
- (2) *PreXmt plugin*: interacts with the message transport system (MTS) to send outgoing messages to neighbors.
- (3) *PostRecv plugin*: interacts with the MTS to intercept incoming messages. It also decides which plugin should be called to process the message depending on its type.
- (4) *AppAPR plugin*: responsible for interacting with the user, getting their queries and answers thereof.
- (5) *Sampling plugin*: responsible for setting schedule to sample the environment (this may be specified by the application user). It is also responsible for interacting with the sensor hardware.
- (6) *Location plugin*: responsible for keeping track of the current agent location (whether using a GPS or a localization algorithm).
- (7) *Query Processing plugin*: responsible for answering queries. It interacts with the cache management plugin to search the local cache. It also interacts with the PreXmt and PostRcv plugins to query neighbors, if needed.
- (8) *Sample diffusion plugin*: responsible for handling sample diffusion when meeting neighbors. This plugin interacts with the cache management plugin to get samples to diffuse to other neighbors and to store samples from neighbors.

The Cougaar Message transport system (MTS) enables communication between different agents. It supports a number of loadable transport protocols. The MTS we choose is an asynchronous light-weight UDP protocol with retry mechanism and support for correct in-order delivery.

The Cougaar blackboard enables communication between different plugins in the same agent. It functions via traditional publish/subscribe semantics. Examples of relations between plugins in the APR agent include a publish `getLocation()` service by the location plugin, to which the sampling plugin subscribes. Another example is the `SendMessage()` service published by PreXmt plugin, to which query processing and sample diffusion plugins subscribe.

APR agents would need to employ a Cougaar Community that enables communication between different agents performing the same task. A Cougaar community is a combination of a membership list service with a mechanism for community-based broadcast messaging.

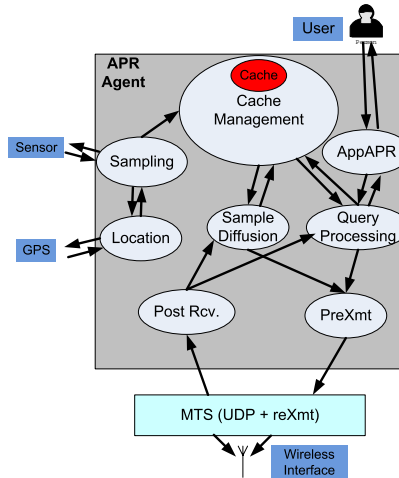


Figure 5: Sketch of APR agent in the Cougar system.

## 7 Directed Placement and Retrieval (DPR)

A radically different approach to “amorphous” placement is planned or directed placement. In this approach the system plans the storage location of every group of samples. The storage location is independent of the location of sensors collecting these samples, hence, this approach mandates transporting samples from the collecting sensor(s) to the storage sensor(s). We call the latter the *home* of the sample. DPR has two main questions to resolve: (1) how to plan sample placement, and (2) how to transport samples from the collecting sensors to the home sensor. Hashing is a widely used technique to answer questions like the first. In systems like [22] and [25], it was proposed to hash samples (based on the sample name) to some location in the field. Nodes closest to that location are considered the home nodes for these samples. To account for mobility, sample replication has been employed to maintain the semantics of the approach (i.e., hashing any sample  $e$  to get a field location, sensors closest to a hashed location are the home sensors for  $e$ ). Queries are likewise hashed using the same function to get the location where answers to the query should be found. To answer the second question, geographic routing techniques (e.g., GPSR [14]) were employed. Assuming nodes are aware of their own location and that of their neighbors, GPSR can be used to route packets to the node closest to a given location in the field. It is clear that, unlike APR, DPR-like approaches depend on multi-hop communication, which consumes much energy. In this paper, we use a slightly different version from the one described above. Instead of hashing samples and queries to field locations, we hash them to node ID’s. It is worth noting that, DPR-like algorithms are not originally designed to handle mobile nodes. However, to allow for fair comparison between APR and DPR, we assume that, under DPR, any two nodes  $a$  and  $b$  route packets (samples, queries, and query answers packets) between them on the *optimum* route found by applying Dijkstra’s algorithm. Dijkstra’s algorithm requires instant knowledge of the whole network topology — a piece of information that many realistic systems would lack. Therefore, results reported in this paper should be viewed as providing an upper-bound on any realistic implementation of DPR algorithms. The hashing of samples is based on the

sample location. More specifically, we divide the field into *Responsibility Regions* (RR for short), each region is assigned to a node. All samples collected by any node at the RR of node  $z$  are forwarded to  $z$ .  $z$  manages its cache such that, it keeps samples collected only from its RR. Queries are likewise hashed, based on the query target to get the ID of the home node. Queries are forwarded to their home node, and answers are routed back to inquirer. Having in mind that, nodes only keep samples from their respective RR, the cache management technique used to manage these samples does not have a huge impact on performance. We experimented with both RCM and QCCM, and results were very close. The reason is that the area of RR is usually much smaller than that of the field, that the effect of the cache management is not really noticeable on performance.

## 8 Performance Evaluation

We evaluated the two approaches we identified for storage and retrieval of sensory data in mobile sensor networks using simulations. Next we give details of basic setup.

**Simulation Model and Setup:** we conducted a set of detailed packet-level simulation experiments, in which we used identical mobility and sampling scenarios for the various approaches. Mobility scenarios for our experiments were generated off line using different mobility models, including the corrected version of the Random Waypoint mobility model [18], the Random Direction model [23] and the Boundless Simulation Area model [8]. Since results of all mobility models are similar, we only report results for the corrected Random Waypoint model. In our simulations, we set the minimum and maximum speed of motion to 0.1 m/sec, and 20 m/sec, respectively.

The sampling process follows a Poisson process with exponential inter-arrival time of two seconds, whereby a sample at time  $t$  constitutes the sensed value of the field at the current location of the node. We report results of simulating 100 mobile nodes moving in a field of  $1400\text{m} \times 1400\text{m}$ , where distance is measured in normal Euclidean distance. The simulation runs for 5,000 seconds. In the following figures, every point is the average of a 20 simulation runs, with 95% confidence intervals shown. Notice confidence intervals are extremely small in most cases.

**Performance Metrics:** the first metric we use is query success ratio (QSR), which is a ratio between the number of successfully answered queries to the number of all queries. To measure efficiency in terms of consumed energy, we compute the number of successfully answered queries per unit energy (Success Per Energy SPE, for short). The goal is to achieve a high query success ratio consuming the least amount of energy. We use an energy model based on the model presented in [11] (Equations 1 and 2).

**Effect of APR Parameters:** in Section 4, we alluded to the importance of two parameters in APR, specifically, the rate of sample diffusion  $\alpha$ , and the size of the diffusion  $k$ . The following two experiments show the effect of these parameters on the performance of APR.

**Effect of Exchange size  $k$ :** intuitively, we expect that the larger the exchange size, the higher the query success rate, and the more consumed energy. Figure 6(left) shows the effect of  $k$  on QSR. It is clear that increasing  $k$  improves the success of APR. Improvement of more than 10% can be gained using larger exchange sizes. However,

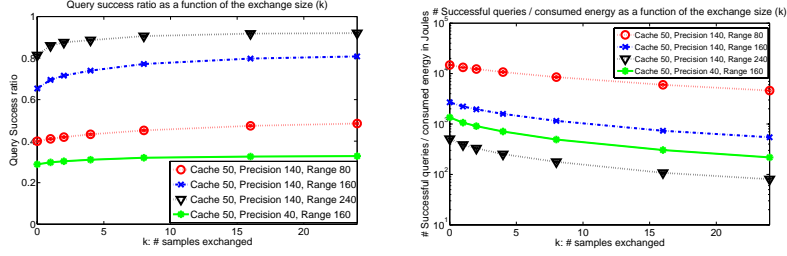


Figure 6: Effect of sample exchange size: Query success ratio (left) and Query success ratio per unit of energy (right).

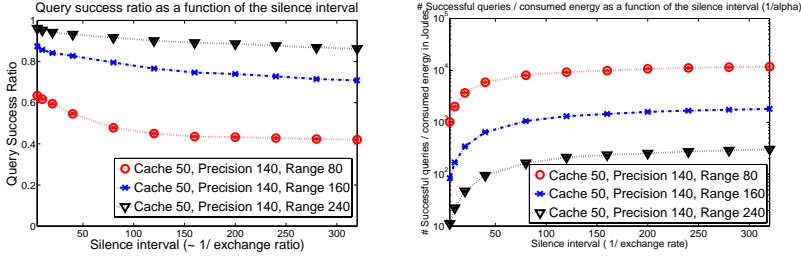


Figure 7: Effect of exchange rate: Query success ratio (left) and Query success ratio per unit of energy (right).

increasing  $k$  for queries with tight precision constraints (*i.e.*, small value of  $\ell$ ) does not help the performance much. The reason for this will be explained later when we discuss the effect of  $\ell$  on APR. Figure 6(right) shows the effect of  $k$  on the SPE. As expected, increasing the exchange size consumes more energy which negatively affects performance. As apparent from the figures, there is no optimum setting for  $k$ , rather selecting a specific value of  $k$  is a compromise between QSR and SPE. In the rest of the experiments we use  $k = 4$  to benefit from the sample diffusion mechanism without consuming much energy in the process.

**Effect of Exchange rate  $\alpha$ :** performing sample diffusion more often should lead to better coverage of the field, leading to higher QSR. Figures 7(left) and 7(right) show the effect of the silene interval ( $=1/\alpha$ ) on performance; smaller values of  $\alpha$  invigorate more frequent sample diffusion which results in higher QSR. However, exchanging more packets consumes more energy which decreases efficiency (in terms of SPE). Again, in this case there is no clear optimum value for  $\alpha$ . In later experiments we use  $\alpha = 0.005 \text{ sec}^{-1}$  (*i.e.*, maximum silence interval of 200 seconds), to minimize the energy consumed in sample diffusion, and to avoid packet collisions due to storms of Hello and Exchange packets. The overall conclusion of these two experiments is that, parameterizing APR is platform-specific, *i.e.*, when deployed on energy-sensitive devices, slower sample diffusion would be an efficient strategy. While when deployed on energy-unconstrained devices, taking advantage of the high QSR presented by higher rate of sample diffusion could be the preferred strategy.

**Effect of APR Mechanisms:** APR has two main mechanisms: sample diffusion, and

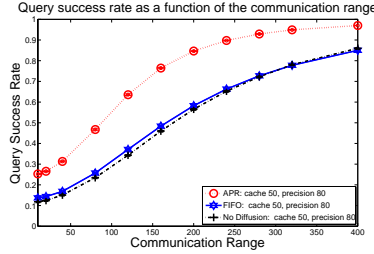


Figure 8: Effect of APR mechanisms on QSR.

**QCCM.** In this subsection, we quantify the effect of each component on APR’s performance. Towards this end, we compare three different versions of APR: 1) Normal APR with both sample diffusion and QCCM, 2) APR with QCCM but no sample diffusion, and 3) APR with sample diffusion, and FIFO cache management. Figure 8 shows QSR of the three versions as a function of the communication range  $r$ . It is clear that sample diffusion coupled with QCCM achieves the highest QSR. There is a clear difference in performance between APR and FIFO cache management validating our analytical findings in Section 5. On the other hand, disabling the sample diffusion mechanisms hinders the performance of APR.

**APR vs. DPR:** In this subsection we compare amorphous placement to directed placement. To that end, we vary the following parameters: communication range  $r$ , query precision  $\ell$ , cache size  $c$ , TTL, and packet loss probability (PLP). We also study the effect of varying the distribution of queries to a non-uniform distribution, and finally we quantify the effect of mobility (or lack thereof) on both protocols. Unless otherwise noted, the default values are:  $r = 160\text{m}$ ,  $\ell = 140\text{m}$ ,  $c = 50$ , TTL = 200 secs, and PLP = 0.

**Communication Range:** communication range  $r$  defines the level of connectivity in the network. We argued that DPR would achieve high QSR only when the network is very-well connected, while APR is able to achieve better QSR in less connected networks. Validating our intuition, Figure 9(left) shows query success ratio for APR and DPR using different values for query precision. It is clear that APR outperforms DPR for networks with smaller communication range, while the roles are reversed when we increase the communication range only for queries have tight precision requirements. Notice that, for queries with lax precision requirements, APR is much more cost-effective even in well-connected networks.

To visualize the impact of network connectivity on QSR of DPR, we also plot the probability of having a connected network as a function of the communication range. This curve is based on the network connectivity model presented in [5]. It is clear that DPR’s performance peaks only when the network is well connected. Increasing the value of  $\ell$  (*i.e.*, making precision requirement less strict) helps APR outperform DPR over a wider region of communication ranges. We discuss this effect in more detail in a following experiment. As for SPE, Figure 9(right) shows that, for shorter communication ranges DPR achieves better performance at higher energy consumption level than APR. As the communication range increases, DPR consumes much more energy compared to APR rendering it inefficient in terms of SPE. This is mainly because, unlike

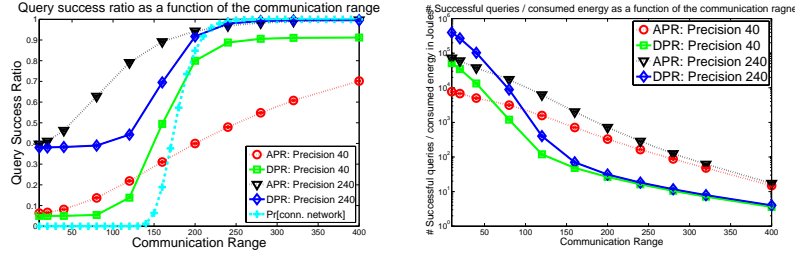


Figure 9: Effect of communication range: Query success ratio (left) and Query success ratio per unit of energy (right).

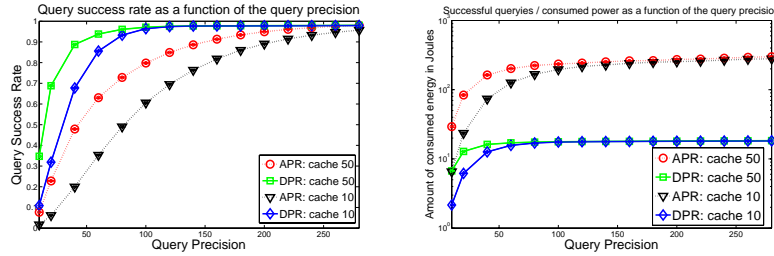


Figure 10: Effect of query precision: Query success ratio (left) and Query success ratio per unit of energy (right).

APR, DPR depends on multi-hop communication which consumes much energy. This experiment hints that using very short communication ranges, APR delivers higher performance, but at a higher cost. Increasing the communication range makes APR more efficient in terms of QSR and SPE compared to DPR. When the network is highly connected, DPR delivers better performance in terms of QSR, only for queries with tight precision requirements. For those with loose precision requirements, APR is the best choice for almost all communication ranges.

**Effect of Query Precision:** given a query target, query precision is the maximum distance we allow between the query target and any sample that can be used to answer this query. Assuming uniform distribution of queries, APR aims to give all nodes a uniform coverage of the entire field. Since nodes' caches are limited, the supported precision under APR will be unavoidably limited. On the other hand, DPR gives each node a very detailed view of a specific region in the field, suggesting that it should be able to handle queries with tighter precision requirements. Figure 10 shows QSR and SPE as a function of the precision. DPR excels in tight precision requirements (and good network connectivity) as measured by QSR, but at a higher energy consumption. As the precision requirement is relaxed, APR catches up and eventually outperforms DPR with much more efficient performance in terms of SPE. Notice the improvement in APR's performance as we increase the cache size; the larger the cache size, the higher the query success ratio at a given precision requirement.

**Cache Size:** Increasing the storage capacity of nodes helps APR cover the field with a better precision, while it does not affect DPR much, since adding more cache space

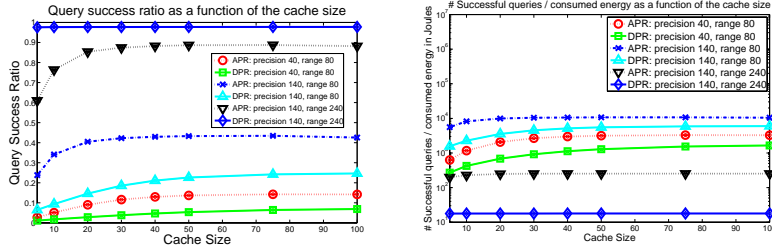


Figure 11: Effect of cache size: Query success ratio (left) and Query success ratio per unit of energy (right).

does not change the level of network connectivity, the dominant factor in DPR’s performance. Figure 11 shows the performance of APR vs. DPR as a function of the cache size. As we noticed, the cache size barely helps DPR’s QSR, while it has a more noticeable performance on that of APR. Beyond a certain cache size, APR’s performance reaches a steady state plateau and remains constant even if we increase the cache size. There are multiple factors to this behavior. First, the TTL of samples = 200 secs, and the sampling rate is 0.5 sample/sec (*i.e.*, 1 sample every 2 seconds), suggesting that cache sizes larger than 100 are not particularly useful. Second, one of the determinant factor in APR performance is the sample diffusion process, whose parameters: the exchange rate  $\alpha$ , and size  $k$  have noticeable effect on the performance of APR. Adding more cache, does not help the sample diffusion process hence the performance of APR reach a steady state. On the other hand, as we have already noticed, relaxing the precision constraint improves the performance of APR, while increasing the communication range boosts that of DPR. In terms of energy consumption, APR is more efficient in all cases.

**Sample Freshness (TTL):** APR depends on mobility and one-hop sample exchange to diffuse samples throughout the field, while DPR uses multi-hop communication to achieve the same effect. Since mobility is slower than multi-hop communication, DPR is expected to beat APR for data types with small TTL. However, a larger TTL allows enough time for the sample diffusion process in APR to function properly, resulting in much improved performance. Figure 12 shows the performance of APR vs. DPR as a function of the TTL of samples. For samples with short TTL, DPR delivers better performance than APR in terms of QSR. While for samples with longer TTL, APR outperforms DPR. For the same value of TTL, increasing cache size helps APR but not DPR, and increasing the communication range helps both, but its effect is more spoken on DPR. For all values of TTL, APR is more efficient in terms of energy consumption.

**Packet Loss Probability (PLP):** Figure 13 shows the performance of APR vs. DPR as a function of packet loss probability (PLP). PLP effect is more pronounced on DPR. The reason is that, DPR depends on multi-hop communication in sample storage, query forwarding, and query response forwarding, which makes it more vulnerable to packet losses. APR, on the other hand, depends on one-hop communication which makes it more resilient to packet losses. Notice that the difference between APR and DPR in query success ratio at loss probability = 1 is attributed to the cache management algorithm. APR applies QCCM over all samples collected all over the field, unlike DPR which applies QCCM only over samples collected at its own RR.

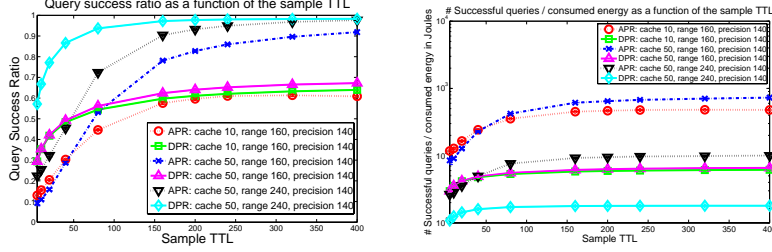


Figure 12: Effect of sample TTL: Query success ratio (left) and Query success ratio per unit of energy (right).

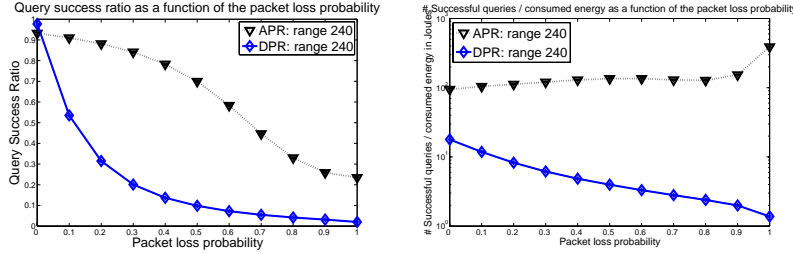


Figure 13: Effect of packet loss: Query success ratio (left) and Query success ratio per unit of energy (right).

**Performance in Static Networks:** One might expect that in static networks, APR’s performance will deteriorate significantly compared to DPR. In this section, we show that, counter-intuitively, lack of mobility does not impact the general behavior of APR’s performance significantly.

In mobile networks, nodes get multiple chances of getting in contact with different neighbors allowing them better sample diffusion and thus an improved view of the entire field. In case of static networks, APR depends, indirectly, on delay-tolerant multi-hop dissemination to achieve this effect. To see why this is the case, consider a node ( $z$ ) at location  $(x_z, y_z)$  in a completely static network. Due to its immobility, all samples cached by  $z$  will be from location  $(x_z, y_z)$ . This will be true, until  $z$  starts sample diffusion process with its neighbors, at which point,  $z$  will cache samples gathered at locations of its direct neighbors. As the sample diffusion process continues, and QCCM is applied,  $z$  will eventually cache samples gathered at neighbors of its direct neighbors, and so on. This effect goes on until  $z$  gets a uniform view of the entire field. The combination of QCCM and informed sample diffusion help to diversify the cache contents of all nodes improving the performance of the entire system. Mobility only speeds up this process, especially when the network is not well connected.

The repeated diffusion of samples to nodes farther from the collecting nodes is one form of delay-tolerant multi-hop communication. However, in this case, unlike DPR, nodes on the way get a chance to cache such samples themselves. Figure 14 shows the performance of APR and DPR in a static network as a function of the query precision. The relative trend in the performance of APR and DPR is not significantly changed (*i.e.*,

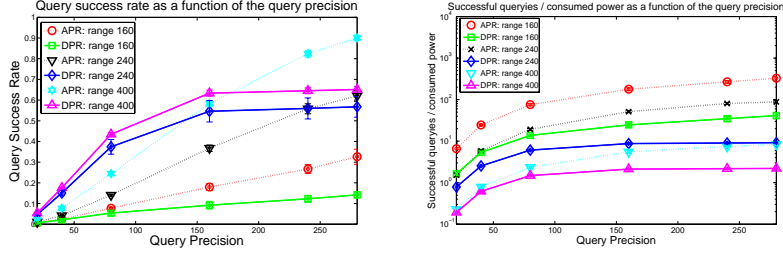


Figure 14: Effect of query precision in a static network: Query success ratio (left) and Query success ratio per unit of energy (right).

relaxing the precision constraint improves APR’s performance). This accentuates the effectiveness of APR’s mechanisms in delivering high performance even in networks with no/limited mobility. Regarding energy efficiency, Figure 14 (right) shows that APR is always more efficient than DPR.

It is worth pointing out that, the effect of network partitioning is more pronounced when there is lack of mobility. In APR, mobility helps nodes that are temporarily isolated to come in contact with neighbors and exchange valuable samples, which improves the field view at these nodes. When there is no/limited mobility, and partitioned network, disconnected nodes have no such chance and hence their performance deteriorates. This effect is more magnified under DPR, since having persistent network partitions harms the performance of the *entire* system (due to partitioning the field into RR’s and assigning an RR to each node), as opposed to harming the performance of only the group of disconnected nodes, under APR. Another weakness in DPR is that, since some of the RR will not have sensors reside in them, queries about these RR will be always missed. Since APR does not depend on the idea of RR, but rather searches for the sample closest to the query target, the performance of APR for the same queries is decidedly better. The probability of this scenario happening increases as we relax the precision constraint and increase the communication range (see Figure 14).

**Non-uniform Query Distribution:** In Section 3, we described a solution when the query targets follows a non-uniform symmetric distribution over the field. As a proof of concept, we show results when assuming queries follow a bivariate normal distribution whose mean  $\mu$  is the center of the field (*i.e.*,  $\mu = (L/2, L/2)$ ) and a symmetric covariance matrix. Figure 15 (left) shows the effect of applying The Stretching Algorithm on a set of points sampled according to a bivariate normal distribution with  $\mu = (700, 700)$ , and a  $2 \times 2$  covariance matrix  $\Sigma$  that has  $\sigma^2$  on the diagonal entries and 0.6 at the off-diagonal entries, and  $\sigma^2 = 17000$ . It shows the original locations of the samples (marked by solid circles) and their mapped images (marked by squares). It is clear that the distribution of the original samples is skewed (all samples are concentrated around the mean), while the distribution of the mapped samples approaches a uniform distribution.

Figure 15 (center and right) shows the results of APR without applying the stretching algorithm, APR with the stretching algorithm and DPR when the query distribution is a bivariate normal distribution with mean  $\mu$  that is the center of the field, and symmetric covariance matrix  $\Sigma$  with diagonal elements =  $\sigma^2$ . The values of  $\sigma^2$  are depicted on

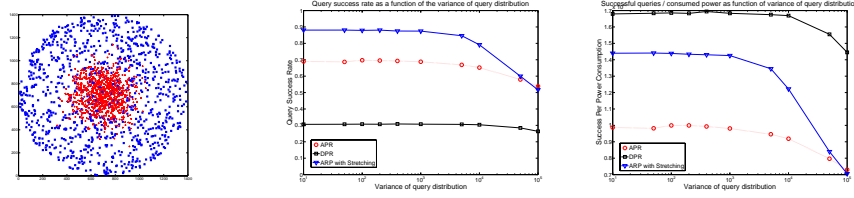


Figure 15: Effects of non-uniform query distribution. Output of the stretching algorithm (left), QSR (Center), success per consumed power unit(Right).

the x-axis. For skewed access patterns, the stretching algorithm succeeds in mapping sample locations to match a uniform distribution yielding superior performance. When the variance grows such that the query distribution approaches a uniform distribution, Performance of APR with stretching matches APR with no stretching.

**Summary of Findings:** We conclude this section with a summary of findings from all of our experiments. In this paper, we have shown that:

- Communication range is the main determinant of DPR’s performance: a loosely connected network renders DPR dysfunctional. In contrast, APR features higher resilience to network disconnectivity.
- In well-connected networks, queries with tighter precision constraints are better handled by DPR than APR. Relaxing precision constraints improves APR’s performance. In loosely-connected networks, APR is better than DPR, even for queries with tight precision constraints.
- Unlike DPR, APR is able to take advantage of increased cache sizes in all settings.
- In well-connected networks, when the monitored field values have tight freshness (TTL) constraints, DPR beats APR in handling queries with stringent precision constraints. APR’s performance improves as we increase the value of TTL. In loosely-connected networks, the performance of APR dominates that of DPR, irrespective of freshness (TTL) constraints.
- APR features much higher resilience to packet losses (and node failures) compared to DPR.
- APR’s performance is not significantly affected by lack of mobility. In fact, when the network is not well connected, lack of mobility negatively impacts the performance of DPR much more than that of APR.
- Applying a mapping (a linear transformation) to sample locations before feeding them to QCCM, enables APR to deliver superior performance when the query distribution is non-uniform over the field.
- APR is more energy efficient than DPR in almost all situations.

## 9 Related Work

There have been extensive research on data management and query resolution in sensor networks. Applications where sensors are mobile and produce large-size samples (e.g., cameras) make these problems more challenging. Due to space limitations, we restrict our attention to only a representative sample of related research, which we broadly categorize based on whether the network or the users (sinks) are mobile.

**Static/mobile network, static sinks:** Data Centric Routing (DCR) and Data Centric Storage (DCS) fall into this category. DCR, such as Directed Diffusion [12], employs flooding. The overhead of flooding is amortized assuming long-running queries. A sink floods its query/interest, and targeted sensors respond. In our APR scheme, each node is able to answer queries locally, or at worst, using limited-scope flooding, since each node actively collects a view of the entire field that matches the spatial distribution of the query targets.

DCS [27] attempts to avoid flooding altogether, hence is suitable for one-shot queries. DCS employs hashing to associate a data item with a specific location in the field. A geographic routing protocol, such as GPSR [14], is used to transport a query/answer for a data item. We compared APR with DPR, which is a DCS approach.

Mobility challenges the design of both DCR and DCS—it continually changes the topology underneath the routing protocol. Other proposals, such as data mules [26], smart tags [4] and mobile relays [32], employ mobile elements as relays among static nodes. These schemes target delay-tolerant applications, which is not the focus of this paper. Another delay-tolerant scheme was proposed by Small *et al.* [28] for a whale monitoring application.

**Static network, mobile sinks:** Both TTDD [35] and SEAD [16] fall into this category. They target long-running queries from mobile users. Essentially, these schemes can be thought of as a hierarchical extension to Directed Diffusion, whereby the effect of sink mobility is localized.

**Mobile network, mobile sinks:** The work by Zaho *et al.* [37] and Lee *et al.* [17] fall into this category. The first employs powerful message ferries to act as relays. In the latter proposal, each node keeps track of its recent contacts, along with their sensed events, and employs last-encounter routing to locate a target node. In a similar setting of delay-tolerant applications, Wang *et al.* [33] employs history-based forwarding and buffer management.

Our proposed APR protocol also fall into this category, albeit its suitability for delay-sensitive applications. APR does not require external mobile elements, such as ferries or data mules. And APR takes a different, proactive approach to improve query performance. The key idea in APR is that it decouples the negative effects of uncontrolled mobility on query performance, by making cache management cognizant of the query profile. Thus, queries can be readily answered from the field view (samples) stored in the local cache or very few neighboring caches.

Another related field, is data management in ad hoc networks [36, 9]. The main difference between these efforts and ours is that: usually, in ad hoc networks, the set of data objects is a limited set with a known source for every object, which is not the case in sensor networks, since the field locations that can be sampled are too many, and

any node can sample any field location. More over, correlation between different data objects are usually ignored. Hara *et al.* Consider this correlation in [10]. However, the correlation structure they consider is random. In our case, the correlation between samples is manifested in utilizing samples to answer queries targeting close-by field locations. Hence, the correlation is not random and has a physical interpretation.

Finally, sample diffusion idea used by APR resembles gossiping and randomized rumor routing [15, 6, 31]. In these efforts multi-hop routing of delay-tolerant data is avoided and mobility is deployed instead. However, as we have shown, APR is flexible enough to resort to delay-tolerant multi-hop forwarding when the need arises. The sample diffusion process also borrows ideas from cache summary [7] by Fan *et al.* to maximize its gain.

## 10 Conclusion

In this paper, we presented a proactive approach, APR, that amorphaously places and diffuses sensor data collected by autonomously mobile nodes, allowing nodes (and node neighborhoods) to compile an integrated view of the monitored field of interest, in anticipation of freshness-constrained and precision-constrained queries thereof. A salient feature of APR is that it enables the management of the nodes' cache content in such a way so as to match the distribution of query targets, regardless of the distribution of the locations that are collectively visited (and sensed). Given a uniform distribution of queries over the space, we demonstrated, by analysis and extensive simulations, how query performance improves under an informed (query-aware) diffusion of sensory samples that maximizes the minimum distance between samples in a node's cache. Our current work is focused on the development of a general transformation that allows APR to handle arbitrary distributions of query targets – and not only the symmetric distributions we considered in this work.

## References

- [1] Cougar.<http://cougar.cougar.org/software/latest/doc/roadmap.html>.
- [2] Lifeguard, wearable wireless physiological monitor. <http://lifeguard.stanford.edu/>.
- [3] M. Ali and Z. A. Uzmi. CSN: A network protocol for serving dynamic queries in large-scale wireless sensor networks. In *CNSR'04*, pages 165–174, Fredericton, N.B, Canada, May 2004.
- [4] A. Beaufour, M. Leopold, and P. Bonnet. Smart-tag based data dissemination. In *WSNA '02*, pages 68–77, New York, NY, USA, 2002. ACM Press.
- [5] C. Bettstetter. On the connectivity of wireless multihop networks with homogeneous and inhomogeneous range assignment. In *IEEE Vehicular Technology Conf. (VTC)*, Vancouver, BC, Canada, September 2002.
- [6] R. R. Choudhury. Brownian gossip: Exploiting node mobility for diffusing information in wireless networks. In *Workshop on Stochasticity in Distributed Systems (StoDis)*, December 2005.

- [7] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: a scalable wide-area web cache sharing protocol. *IEEE/ACM Trans. Netw.*, 8(3):281–293, 2000.
- [8] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proc. of the IEEE Int. Conf. on Universal Personal Communications (ICUPC)*, pages 562–565, October 1997.
- [9] T. Hara. Effective replica allocation in ad hoc networks for improving data accessibility. In *Infocom '01*, pages 1568–1576, 2001.
- [10] T. Hara, N. Murakami, and S. Nishio. Replica allocation for correlated data items in ad hoc sensor networks. *SIGMOD Rec.*, 33(1):38–43, 2004.
- [11] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 8*, page 8020, Washington, DC, USA, 2000. IEEE Computer Society.
- [12] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *MobiCom '00*, pages 56–67, 2000.
- [13] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: the dynamic source routing protocol for multihop wireless ad hoc networks. pages 139–172, 2001.
- [14] B. Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00*, pages 243–254, New York, NY, USA, 2000. ACM Press.
- [15] A.-M. Kermarrec, L. Massouli, and A. J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(3):248–258, 2003.
- [16] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon. Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In *SenSys '03*, pages 193–204, New York, NY, USA, 2003. ACM Press.
- [17] U. Lee, E. Magistretti, B. Zhou, M. Gerla, P. Bellavista, and A. Corradi. Efficient data harvesting in mobile sensor platforms. In *PERCOMW '06*, page 352, Washington, DC, USA, 2006. IEEE Computer Society.
- [18] G. Lin, G. Noubir, and R. Rajamaram. Mobility models for ad-hoc network simulation. In *INFOCOM*, 2004.
- [19] C. Lu, G. Xing, O. Chipara, C.-L. Fok, and S. Bhattacharya. A spatiotemporal query service for mobile users in sensor networks. In *ICDCS '05*, pages 381–390, Washington, DC, USA, 2005. IEEE Computer Society.
- [20] P. Lukowicz, U. Anliker, J. Ward, G. Trster, E. Hirt, , and C. Neufelt. Amon: A wearable medical computer for high risk patients. In *ISWC, the 6th International Symposium on Wearable Computers*, pages 133–134, October 2002.
- [21] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. 2003.
- [22] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker. Ght: a geographic hash table for data-centric storage. In *WSNA '02*, pages 78–87, New York, NY, USA, 2002. ACM Press.
- [23] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser. An analysis of the optimum node density for ad hoc mobile networks. In *ICC*, pages 857–861, Helsinki, Finland, June 2001.
- [24] R. S., F. P., H. M., K. R., and S. S. A scalable content-addressable network. In *SIGCOMM '01*, pages 161–172, New York, NY, USA, 2001. ACM Press.
- [25] K. Seada and A. Helmy. Refereed poster: Rendezvous regions: A scalable architecture for service provisioning in large-scale mobile ad hoc networks. In *ACM SIGCOMM*, 2003.

- [26] R. Shah, S. Roy, S. Jain, and W. Brunette. Data mules: Modeling a three-tier architecture for sparse sensor networks. In *IEEE SNPA Workshop*, May 2003.
- [27] S. Shenker, S. Ratnasamy, B. Karp, R. Govindan, and D. Estrin. Data-centric storage in sensornets. *SIGCOMM Comput. Commun. Rev.*, 33(1):137–142, 2003.
- [28] T. Small and Z. J. Haas. The shared wireless infostation model: a new ad hoc networking paradigm (or where there is a whale, there is a way). In *MobiHoc '03*, pages 233–244, New York, NY, USA, 2003. ACM Press.
- [29] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160, New York, NY, USA, 2001. ACM Press.
- [30] S.-H. Teng. Low energy and mutually distant sampling. *J. Algorithms*, 30(1):52–67, Jan 1999.
- [31] S. Voulgaris, M. Jelasity, and M. van Steen. A robust and scalable peer-to-peer gossiping protocol. In *2nd Int'l Workshop Agents and Peer-to-Peer Computing, LNCS 2872*, Springer, 2003.
- [32] W. Wang, V. Srinivasan, and K.-C. Chua. Using mobile relays to prolong the lifetime of wireless sensor networks. In *MobiCom '05*, pages 270–283, New York, NY, USA, 2005. ACM Press.
- [33] Y. Wang and H. Wu. Dft-msn: The delay/fault-tolerant mobile sensor network for pervasive information gathering. In *INFOCOM '06*, Barcelona, Spain, April 2006.
- [34] G. H. Weiss. *Aspects and Applications of the Random Walk*. Elsevier Science B.V., North-Holland, 1994.
- [35] F. Ye, H. Luo, J. Cheng, S. Lu, and L. Zhang. A two-tier data dissemination model for large-scale wireless sensor networks. In *MobiCom '02*, pages 148–159, New York, NY, USA, 2002. ACM Press.
- [36] L. Yin and G. Cao. Supporting cooperative caching in ad hoc networks. In *Proceedings of the 23rd Conference of the IEEE Communications Society (Infocom), March 7 - 11, 2004, Hong Kong, Hong Kong*, March 2004. IEEE Infocom.
- [37] W. Zhao, M. Ammar, and E. Zegura. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04*, pages 187–198, New York, NY, USA, 2004. ACM Press.